Electronic Transactions on Numerical Analysis.
Volume 45, pp. 107–132, 2016.

ISSN 1068–9613.

ETNA
Kent State University
http://etna.math.kent.edu

LOW-RANK SOLVERS FOR FRACTIONAL DIFFERENTIAL EQUATIONS*

TOBIAS BREITEN†, VALERIA SIMONCINI‡, AND MARTIN STOLL§

Abstract. Many problems in science and technology can be cast using differential equations with both fractional time and spatial derivatives. To accurately simulate natural phenomena using this technology, fine spatial and temporal discretizations are required, leading to large-scale linear systems or matrix equations, especially whenever more than one space dimension is considered. The discretization of fractional differential equations typically involves dense matrices with a Toeplitz structure in the constant coefficient case. We combine the fast evaluation of Toeplitz matrices and their circulant preconditioners with state-of-the-art linear matrix equation methods to efficiently solve these problems, both in terms of CPU time and memory requirements. Additionally, we illustrate how these techniques can be adapted when variable coefficients are present. Numerical experiments on typical differential problems with fractional derivatives in both space and time showing the effectiveness of the approaches are reported.

Key words. fractional calculus, fast solvers, Sylvester equations, preconditioning, low-rank methods, tensor equations

AMS subject classifications. 65F08, 65F10, 65F50, 92E20, 93C20

1. Introduction.

The study of integrals and derivatives of arbitrary order, so-called fractional order, is an old topic in mathematics going back to Euler and Leibniz (see [19] for historical notes). Despite its long history in mathematics it was not until recently that this topic has gained mainstream interest outside the mathematical community. This surging interest is mainly due to the inadequateness of traditional models to describe many real world phenomena. The well-known anomalous diffusion process is one typical such example [37]. Other applications of fractional calculus are viscoelasticity - for example using the Kelvin-Voigt fractional derivative model [28, 63], electrical circuits [25, 47], electro-analytical chemistry [58], or image processing [48].

With the increase of problems using fractional differential equations there is corresponding interest in the development and study of accurate, fast, and reliable numerical methods that allow their solution. There are various formulations for the fractional derivative mainly divided into derivatives of Caputo or Riemann-Liouville type (see definitions in Section 2). So far, much of the numerical analysis focused on ways to discretize these equations using either tailored finite difference [35, 45] or finite element [16, 34] methods. Of particular importance is the preconditioning of the linear system, which can be understood as additionally employing an approximation of the discretized operator. For some types of fractional differential equations, preconditioning has recently been considered (see [32, 38]). Another approach that has recently been studied by Burrage et al. considers approximations to matrix functions to solve the discretized system (see [5] for details). In this paper we focus on the solution of the discretized equations when a finite difference approximation is used. Our work here is motivated by some recent results in [46], where the discretization via finite differences is considered in a purely algebraic framework.

*Received July 1, 2015. Accepted January 28, 2016. Published online on April 22, 2016. Recommended by Daniel Kressner. Most of this work was completed while the first author was with the Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg.

†Institute for Mathematics and Scientific Computing, Heinrichstr. 36/III, University of Graz, Austria (tobias.breiten@uni-graz.at).

‡Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, 40127 Bologna, Italy, and IMATI-CNR, Pavia (valeria.simoncini@unibo.it).

§Numerical Linear Algebra for Dynamical Systems, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Germany (stollm@mpi-magdeburg.mpg.de).

ETNA
Kent State University
http://etna.math.kent.edu

Our aim is to exploit a novel linear algebra formulation of the discretized differential equations, which allows us to efficiently and reliably solve the resulting very large algebraic equations. The new framework captures the intrinsic matrix structure in each space and time directions typical of the problem; the corresponding *matrix* or tensor equations are solved with recently developed fast iterative methods, which determine accurate low-rank approximations in the solution manifold.

We therefore structure the paper as follows. Section 2 provides some background on both Caputo and Riemann-Liouville fractional derivatives. In Section 3 we introduce different problems, all of fractional order. The problems are either space-, time- or space-time-fractional. Additionally, we introduce variable and constant coefficients. We start with a problem revealing the basic matrix structure that is obtained when tailored finite difference methods are applied. Later, a more complicated setup will lead to not only having to solve a simple structured linear system, but rather a (linear) matrix Sylvester equation. An even further structured equation is obtained when we consider a two-dimensional (in space) setup. We also consider a problem that combines two-dimensional spatial derivatives of fractional order with a time-fractional derivative, which in turn leads to a tensor structured equation. In Section 4 we turn to constructing fast solvers for the previously obtained matrix equations. We therefore start by introducing circulant approximations to the Toeplitz matrices, which are obtained as the discretization of an instance of a fractional derivative. These circulant matrices are important as preconditioners for matrices of Toeplitz type, but can also be used with the tailored matrix equation solvers presented in Section 4.1, either of direct or preconditioned form. Section 4.3 provides some of the eigenvalue analysis needed to show that our methods perform robustly within the given parameters. This is then followed by a tensor-valued solver in Section 4.4. The numerical results given in Section 5 illustrate the competitiveness of our approaches.

Throughout the manuscript the following notation will be used. MATLAB notation will be used whenever possible; for a matrix $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_n] \in \mathbb{R}^{m\times n}$, $\text{vec}(\mathbf{U}) = [\mathbf{u}_1^T, \dots, \mathbf{u}_n^T]^T \in \mathbb{R}^{nm}$ denotes the vector obtained by stacking all columns of $\mathbf{U}$ one after the other; $\mathbf{A} \otimes \mathbf{B}$ denotes the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$ and $\mathbf{U}^H$ is the complex conjugate transpose of $\mathbf{U}$.

2. Fractional calculus and Grünwald formulae.

2.1. The fractional derivative.

In fractional calculus there are competing concepts for the definition of fractional derivatives. The Caputo and the Riemann-Liouville fractional derivatives [45] are both used here and we use this section to briefly recall their definition.

Consider a function $f = f(t)$ defined on an interval $[a, b]$. Assuming that f is sufficiently often continuously differentiable, the Caputo derivative of real order α with $(n-1 < \alpha \le n)$ is defined as (see, e.g., [46, Formula (3)], [6, Section 2.3])

$$ {}_a^C D_t^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(s)ds}{(t-s)^{\alpha-n+1}}, $$

where $\Gamma(x)$ is the gamma function. Following the discussion in [46], the Caputo derivative is frequently used for the derivative with respect to time. We also define the Riemann-Liouville derivative (see, e.g., [46, Formulas (6-7)]): assuming that f is integrable for $t > a$, a left-sided fractional derivative of real order α with $(n-1 < \alpha \le n)$ is defined as

$$ {}_a^{RL} D_t^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \left(\frac{d}{dt}\right)^n \int_a^t \frac{f(s)ds}{(t-s)^{\alpha-n+1}}, \qquad a < t < b. $$

ETNA
Kent State University
http://etna.math.kent.edu

Analogously, a right-sided Riemann-Liouville fractional derivative is given by

$$ {}_t^{RL}D_b^\alpha f(t) = \frac{(-1)^n}{\Gamma(n-\alpha)} \left(\frac{d}{dt}\right)^n \int_t^b \frac{f(s)ds}{(s-t)^{\alpha-n+1}}, \qquad a < t < b. $$

If one is further interested in computing the symmetric Riesz derivative of order α, one can simply perform the half-sum of the left and right-sided Riemann-Liouville derivatives (see, e.g., [46, Formula (5)]), that is,

$$ \frac{d^\alpha f(t)}{d\,|t|^\alpha} = {}_tD_R^\alpha f(t) = \frac{1}{2}\left({}_a^{RL}D_t^\alpha f(t) + {}_t^{RL}D_b^\alpha f(t)\right). \tag{2.1} $$

Here a connection to both the left-sided and the right-sided derivatives is made[1]. In the remainder of this paper we want to illustrate that fractional differential equations using the formulations together with certain discretization approaches lead to similar structures at the discrete level. Our goal is to give guidelines and offer numerical schemes for the efficient and accurate evaluation of problems of various form. Detailed introductions to fractional differential equations can be found in [45, 51].

2.2. Numerical approximation. The discretization of fractional derivatives is often done by finite difference schemes of Grünwald-Letnikov type. Assume that, for a one dimensional problem, the spatial domain is given by $x \in [a, b]$. We here follow [35] for the introduction of the basic methodology. According to [35], the following holds

$$ {}_a^{RL}D_x^\alpha f(x,t) = \lim_{M\to\infty} \frac{1}{h^\alpha} \sum_{k=0}^{M} g_{\alpha,k} f(x-kh,t), $$

for the left-sided derivative (see (3) in [35] or (7.10) in [45]), where $h = \frac{x-a}{M}$, and $g_{\alpha,k}$ is given by

$$ g_{\alpha,k} = \frac{\Gamma(k-\alpha)}{\Gamma(-\alpha)\Gamma(k+1)} = (-1)^k \binom{\alpha}{k}. $$

For the efficient computation of the coefficients $g_{\alpha,k}$ one can use the following recurrence relation[2] ([45])

$$ g_{\alpha,0} = 1, \quad g_{\alpha,k} = \left(1 - \frac{\alpha+1}{k}\right) g_{\alpha,k-1}. $$

This finite difference approximation is of first order [35]. One can analogously obtain an expression for the right-sided derivative via

$$ {}_x^{RL}D_b^\alpha f(x,t) = \lim_{M\to\infty} \frac{1}{h^\alpha} \sum_{k=0}^{M} g_{\alpha,k} f(x+kh,t), $$

where $h = \frac{b-x}{M}$.

Alternatively, a shifted Grünwald-Letnikov formula is introduced in [35]. This discretization shows advantages regarding the stability when the fractional derivative is part of an

[1] In this work we are not discussing which of these derivatives is the most suitable for the description of a natural phenomenon.

[2] In MATLAB, this can be efficiently computed using $\mathtt{y} = \mathtt{cumprod}([1, 1-((\alpha+1)./(1:\mathtt{n}))])$, where n is the number of coefficients.

ETNA
Kent State University
http://etna.math.kent.edu

unsteady differential equation. The basis of the discretized operator used later is the following shifted expression

$$ {}_a^{RL}D_x^{\alpha} f(x,t) = \lim_{M\to\infty} \frac{1}{h^{\alpha}} \sum_{k=0}^{M} g_{\alpha,k} f(x-(k-1)h,t) $$

introduced in [35, Theorem 2.7].

3. Some model problems and discretizations. In what follows, we introduce some prototypes of fractional differential equations. We start with space fractional problems with constant coefficients. These will then be expanded by an additional time fractional derivative. Similarities and differences in the non-constant coefficient case are pointed out. We emphasize that the subsequent examples are neither the only relevant formulations nor the only possible ways to discretize each problem. Instead, we show that classical discretization techniques typically lead to dense but highly structured problems which, when considered as matrix equations, allow for an efficient numerical treatment.

3.1. Space fractional problems. Consider the one dimensional fractional diffusion equation

$$ \begin{aligned} \frac{du(x,t)}{dt} - {}_xD_R^{\beta} u(x,t) &= f(x,t), \quad && (x,t)\in(0,1)\times(0,T], \\ u(0,t) = u(1,t) &= 0, && t\in[0,T], \\ u(x,0) &= 0, && x\in[0,1], \end{aligned} \tag{3.1} $$

with spatial differentiation parameter $\beta \in (1,2)$. Equation (3.1) is discretized in time by an implicit Euler scheme of step size τ to give

$$ \frac{u^{n+1}-u^n}{\tau} - {}_xD_R^{\beta} u^{n+1} = f^{n+1}, \tag{3.2} $$

where $u^{n+1} := u(x,t_{n+1})$, and $f^{n+1} := f(x,t_{n+1})$ denote the values of $u(x,t)$ and $f(x,t)$ at time $t_{n+1} = (n+1)\tau$. According to (2.1), the Riesz derivative is defined as the weighted average of the left and right-sided Riemann-Liouville derivative. Hence, let us first consider the one-sided analogue of (3.2), i.e.,

$$ \frac{u^{n+1}-u^n}{\tau} - {}_a^{RL}D_x^{\beta} u^{n+1} = f^{n+1}. $$

The stable discretization of this problem is introduced in [35, Formula (18)], where using $u_i^{n+1} := u(x_i,t_{n+1})$, the derivative is approximated via

$$ {}_a^{RL}D_x^{\beta} u_i^{n+1} \approx \frac{1}{h_x^{\beta}} \sum_{k=0}^{i+1} g_{\beta,k} u_{i-k+1}^{n+1}. $$

Here, $h_x = \frac{b-a}{n_x+1}$, and n_x is the number of interior points in space. Incorporating the homogeneous Dirichlet boundary conditions, an approximation to (3.2) in matrix form is given

ETNA
Kent State University
http://etna.math.kent.edu

by

$$\frac{1}{\tau}\begin{bmatrix} u_1^{n+1}-u_1^n \\ u_2^{n+1}-u_2^n \\ \vdots \\ \vdots \\ u_{n_x}^{n+1}-u_{n_x}^n \end{bmatrix} = \underbrace{h_x^{-\beta}\begin{bmatrix} g_{\beta,1} & g_{\beta,0} & 0 & \dots & 0 \\ g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & \ddots & \vdots \\ \vdots & \ddots & \ddots & g_{\beta,0} & 0 \\ g_{\beta,n_x-1} & \ddots & \ddots & g_{\beta,1} & g_{\beta,0} \\ g_{\beta,n_x} & g_{\beta,n_x-1} & \dots & g_{\beta,2} & g_{\beta,1} \end{bmatrix}}_{\mathbf{T}_\beta^{n_x}} \underbrace{\begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{n_x}^{n+1} \end{bmatrix}}_{\mathbf{u}^{n+1}} + \underbrace{\begin{bmatrix} f_1^{n+1} \\ f_2^{n+1} \\ \vdots \\ \vdots \\ f_{n_x}^{n+1} \end{bmatrix}}_{\mathbf{f}^{n+1}}.$$

We now approximate the symmetric Riesz derivative as the weighted sum of the left- and right-sided Riemann-Liouville fractional derivative (see also [46, formula (28)]), to obtain the differentiation matrix

$$\mathbf{L}_\beta^{n_x} = \frac{1}{2}\left(\mathbf{T}_\beta^{n_x} + (\mathbf{T}_\beta^{n_x})^T\right).$$

While we simply take the weighted average of the discretized operators, the justification of the weighted two-sided average is laid in [36, equation (16)], where all c_i are constant at $\frac{1}{2}$ and all f_i are zero. Using this notation it is easy to see that the implicit Euler method for solving (3.2) requires the solution of the algebraic linear system

$$\left(\mathbf{I}^{n_x} - \tau\mathbf{L}_\beta^{n_x}\right)\mathbf{u}^{n+1} = \mathbf{u}^n + \tau\mathbf{f}^{n+1} \tag{3.3}$$

at every time-step. We discuss the efficient solution of this system in Section 4. We here focus on the shifted version of the Grünwald formulae, but we want to remark that all techniques presented in this paper are also applicable when the unshifted version is used.

Next, for $\Omega = (a_x, b_x) \times (a_y, b_y)$, consider the two-dimensional version of (3.1)

$$\begin{aligned} \frac{du(x,y,t)}{dt} - {}_xD_R^{\beta_1}u(x,y,t) - {}_yD_R^{\beta_2}u(x,y,t) &= f(x,y,t), && (x,y,t) \in \Omega \times (0,T], \\ u(x,y,t) &= 0, && (x,y,t) \in \Gamma \times [0,T], \\ u(x,y,0) &= 0, && (x,y) \in \Omega, \end{aligned} \tag{3.4}$$

where $\beta_1, \beta_2 \in (1,2)$ and Γ denotes the boundary of Ω. Using again the shifted Grünwald finite difference for the spatial derivatives, gives in the x-direction

$${}_xD_R^{\beta_1}u(x,y,t) = \frac{1}{\Gamma(-\beta_1)}\lim_{n_x\to\infty}\frac{1}{h_x^{\beta_1}}\sum_{k=0}^{M_x}\frac{\Gamma(k-\beta_1)}{\Gamma(k+1)}u(x-(k-1)h_x,y,t),$$

and then in the y-direction

$${}_yD_R^{\beta_2}u(x,y,t) = \frac{1}{\Gamma(-\beta_2)}\lim_{n_y\to\infty}\frac{1}{h_y^{\beta_2}}\sum_{k=0}^{M_y}\frac{\Gamma(k-\beta_2)}{\Gamma(k+1)}u(x,y-(k-1)h_y,t).$$

With the previously defined weights, and employing an implicit Euler method in time, we obtain the following equation

$$\frac{1}{\tau}\left(\mathbf{u}^{n+1}-\mathbf{u}^n\right) = \underbrace{\left(\mathbf{I}^{n_y}\otimes\mathbf{L}_{\beta_1}^{n_x} + \mathbf{L}_{\beta_2}^{n_y}\otimes\mathbf{I}^{n_x}\right)}_{\mathbf{L}_{\beta_1,\beta_2}^{n_x n_y}}\mathbf{u}^{n+1} + \mathbf{f}^{n+1}.$$

ETNA
Kent State University
http://etna.math.kent.edu

Note that for the numerical approximation we have used $h_x = \frac{b_x - a_x}{n_x+1}$, $h_y = \frac{b_y - a_y}{n_y+1}$ with $n_x + 2$ and $n_y + 2$ degrees of freedom in both spatial dimensions. Again, the boundary degrees of freedom have been eliminated. To proceed with the implicit time stepping, one now has to solve the large linear system of equations

$$\left(\mathbf{I}^{n_x n_y} - \tau \mathbf{L}_{\beta_1,\beta_2}^{n_x n_y}\right) \mathbf{u}^{n+1} = \mathbf{u}^n + \tau \mathbf{f}^{n+1}$$

at each time step t_n. Due to the structure of $\mathbf{L}_{\beta_1,\beta_2}^{n_x n_y}$ we have

$$\mathbf{I}^{n_x n_y} - \tau \mathbf{L}_{\beta_1,\beta_2}^{n_x n_y} = \mathbf{I}^{n_y} \otimes \left(\frac{1}{2}\mathbf{I}^{n_x} - \tau \mathbf{L}_{\beta_1}^{n_x}\right) + \left(\frac{1}{2}\mathbf{I}^{n_y} - \tau \mathbf{L}_{\beta_2}^{n_y}\right) \otimes \mathbf{I}^{n_x}.$$

In other words, the vector $\mathbf{u}^{n+1} \in \mathbb{R}^{n_x n_y}$ is the solution of a Kronecker structured linear system. On the other hand, interpreting $\mathbf{u}^{n+1}, \mathbf{u}^n, \mathbf{f}^{n+1}$ as vectorizations of matrices, i.e.,

$$\mathbf{u}^{n+1} = \mathbf{vec}(\mathbf{U}^{n+1}), \;\; \mathbf{u}^n = \mathbf{vec}(\mathbf{U}^n), \;\; \mathbf{f}^{n+1} = \mathbf{vec}(\mathbf{F}^{n+1}), \;\; \mathbf{U}^{n+1}, \mathbf{U}^n, \mathbf{F}^{n+1} \in \mathbb{R}^{n_x \times n_y},$$

this can be rewritten as a Sylvester equation of the form

$$\left(\frac{1}{2}\mathbf{I}^{n_x} - \tau \mathbf{L}_{\beta_1}^{n_x}\right) \mathbf{U}^{n+1} + \mathbf{U}^{n+1} \left(\frac{1}{2}\mathbf{I}^{n_y} - \tau \mathbf{L}_{\beta_2}^{n_y}\right)^T = \mathbf{U}^n + \tau \mathbf{F}^{n+1}. \tag{3.5}$$

Note that $\mathbf{L}_{\beta_1}^{n_x}$ and $\mathbf{L}_{\beta_2}^{n_y}$ are both symmetric but, in general, different. In Section 4 we shall exploit this key connection to efficiently determine a numerical approximation to $\mathbf{U}^{n+1}$, and thus to $\mathbf{u}^{n+1}$.

3.2. Time fractional problems. We now assume that the time derivative is of fractional order as well. Hence, in a one-dimensional space, let us consider

$$\begin{aligned} {}_0^C D_t^\alpha u(x,t) - {}_x D_R^\beta u(x,t) &= f(x,t), & (x,t) &\in (0,1) \times (0,T], \\ u(0,t) = u(1,t) &= 0, & t &\in [0,T], \\ u(x,0) &= 0, & x &\in [0,1], \end{aligned} \tag{3.6}$$

where $\alpha \in (0,1)$ and $\beta \in (1,2)$.

We again approximate the Riesz derivative as the weighted sum of the left- and right-sided Riemann-Liouville fractional derivatives used before. The Caputo time-fractional derivative ${}_0^C D_t^\alpha u(x,t)$ is then approximated using the unshifted Grünwald-Letnikov approximation to give

$$\mathbf{T}_\alpha^{n_t+1} \begin{bmatrix} u(x,t_0) \\ u(x,t_1) \\ \vdots \\ u(x,t_{n_t}) \end{bmatrix} - \mathbf{I}^{n_t+1} \begin{bmatrix} {}_x D_R^\beta u(x,t_0) \\ {}_x D_R^\beta u(x,t_1) \\ \vdots \\ {}_x D_R^\beta u(x,t_{n_t}) \end{bmatrix} = \mathbf{I}^{n_t+1} \begin{bmatrix} f(x,t_0) \\ f(x,t_1) \\ \vdots \\ f(x,t_{n_t}) \end{bmatrix},$$

where

$$\mathbf{T}_\alpha^{n_t+1} := \tau^{-\alpha} \begin{bmatrix} g_{\alpha,0} & 0 & \dots & \dots & 0 \\ g_{\alpha,1} & g_{\alpha,0} & \ddots & & \vdots \\ \ddots & \ddots & \ddots & \ddots & \vdots \\ \ddots & \ddots & \ddots & g_{\alpha,0} & 0 \\ g_{\alpha,n_t} & \dots & \dots & g_{\alpha,1} & g_{\alpha,0} \end{bmatrix} = \left[\begin{array}{c|c} \tau^{-\alpha} g_{\alpha,0} & \mathbf{0}^T \\ \hline \vdots & \mathbf{T}_\alpha^{n_t} \\ \tau^{-\alpha} g_{\alpha,n_t} & \end{array}\right].$$

ETNA
Kent State University
http://etna.math.kent.edu

Here, $\mathbf{T}_\alpha^{n_t+1}$ represents the discrete Caputo derivative. In case of a non-zero initial condition $u(x,t_0)=u_0$ we get

$$\mathbf{T}_\alpha^{n_t}\begin{bmatrix} u(x,t_1)\\ u(x,t_2)\\ \vdots\\ u(x,t_{n_t})\end{bmatrix} - \mathbf{I}^{n_t}\begin{bmatrix} {}_xD_R^\beta u(x,t_1)\\ {}_xD_R^\beta u(x,t_2)\\ \vdots\\ {}_xD_R^\beta u(x,t_{n_t})\end{bmatrix} = \begin{bmatrix} f(x,t_1)\\ f(x,t_2)\\ \vdots\\ f(x,t_{n_t})\end{bmatrix} - \begin{bmatrix} g_{\alpha,1}\\ g_{\alpha,2}\\ \vdots\\ g_{\alpha,n_t}\end{bmatrix} u(x,t_0).$$

If we now discretize in space, this leads to the following algebraic linear system in Kronecker form

$$\left((\mathbf{T}_\alpha^{n_t}\otimes\mathbf{I}^{n_x}) - (\mathbf{I}^{n_t}\otimes\mathbf{L}_\beta^{n_x})\right)\mathbf{u} = \tilde{\mathbf{f}}, \tag{3.7}$$

where $\mathbf{u}=[\mathbf{u}_1^T,\mathbf{u}_2^T,\dots,\mathbf{u}_{n_t}^T]^T\in\mathbb{R}^{n_xn_t}$. Similarly, we define $\tilde{\mathbf{f}}=[\mathbf{f}_1^T,\mathbf{f}_2^T,\dots,\mathbf{f}_{n_t}^T]^T$. Introducing the matrices $\mathbf{U}=[\mathbf{u}_1,\mathbf{u}_2,\dots,\mathbf{u}_{n_t}]$ and $\mathbf{F}=[\mathbf{f}_1,\dots,\mathbf{f}_{n_t}]$, we can rewrite (3.7) as

$$\mathbf{U}(\mathbf{T}_\alpha^{n_t})^T - \mathbf{L}_\beta^{n_x}\mathbf{U} = \mathbf{F}, \tag{3.8}$$

which shows that $\mathbf{U}$ is the solution to a Sylvester matrix equation (see, e.g., [31]), with $\mathbf{T}_\alpha^{n_t}$ lower triangular and $\mathbf{L}_\beta^{n_x}$ a dense symmetric matrix.

Finally, if we have fractional derivatives in both a two-dimensional space and time, we consider

$$\begin{aligned} {}_0^CD_t^\alpha u(x,y,t) - {}_xD_R^{\beta_1}u(x,y,t) - {}_yD_R^{\beta_2}u(x,y,t) &= f(x,y,t), && (x,y,t)\in\Omega\times(0,T],\\ u(x,y,t) &= 0, && (x,y,t)\in\Gamma\times[0,T],\\ u(x,y,0) &= 0, && (x,y)\in\Omega. \end{aligned}$$

Following the same steps as before we obtain a space-time discretization written as the following algebraic linear system

$$\left(\mathbf{T}_\alpha^{n_t}\otimes\mathbf{I}^{n_xn_y} - \mathbf{I}^{n_t}\otimes\mathbf{L}_{\beta_1,\beta_2}^{n_xn_y}\right)\mathbf{u} = \tilde{\mathbf{f}}. \tag{3.9}$$

The space-time coefficient matrix now has a double tensor structure, making the numerical solution of the associated equation at each time step much more complex than in the previous cases. An effective solution strategy resorts to recently developed algorithms that use approximate tensor computations; see Section 4.4.

3.3. Variable coefficients. In this section we consider a more general FDE, which involves separable variable coefficients

$$\begin{aligned} \frac{du(x,y,t)}{dt} &= p_+(x,y)\,{}^{\mathrm{RL}}_0D_x^{\beta_1}u(x,y,t) + p_-(x,y)\,{}^{\mathrm{RL}}_xD_1^{\beta_1}u(x,y,t)\\ &\quad + q_+(x,y)\,{}^{\mathrm{RL}}_0D_y^{\beta_2}u(x,y,t)\\ &\quad + q_-(x,y)\,{}^{\mathrm{RL}}_yD_1^{\beta_2}u(x,y,t) + f(x,y,t), \qquad (x,y,t)\in\Omega\times(0,T],\\ u(x,y,t) &= 0, \qquad (x,y,t)\in\Gamma\times[0,T],\\ u(x,y,0) &= 0, \qquad (x,y)\in\Omega. \end{aligned}$$

To simplify the presentation, we provide the details only for the two-dimensional space-fractional example, since the other cases can be obtained in a similar manner. We thus consider the case $p_+(x,y)=p_{+,1}(x)p_{+,2}(y)$, and similarly for the other coefficients. The left and right

ETNA
Kent State University
http://etna.math.kent.edu

sided Riemann-Liouville derivatives enter the equation with different coefficients, therefore the block

$$\left(p_+ \, {}^{\mathrm{RL}}_{0}D_x^{\beta_1} + p_- \, {}^{\mathrm{RL}}_{x}D_1^{\beta_1} + q_+ \, {}^{\mathrm{RL}}_{0}D_y^{\beta_2} + q_- \, {}^{\mathrm{RL}}_{y}D_1^{\beta_2}\right) u(x,y,t)$$

becomes

$$\left(\mathbf{P}_{+,1}\mathbf{T}_{\beta_1}\mathbf{U}\mathbf{P}_{+,2} + \mathbf{P}_{-,1}\mathbf{T}_{\beta_1}^T\mathbf{U}\mathbf{P}_{-,2} + \mathbf{Q}_{+,1}\mathbf{U}\mathbf{T}_{\beta_2}\mathbf{Q}_{+,2} + \mathbf{Q}_{-,1}\mathbf{U}\mathbf{T}_{\beta_2}^T\mathbf{Q}_{-,2}\right)$$

or, in Kronecker notation,

$$\left(\mathbf{P}_{+,2}\otimes\mathbf{P}_{+,1}\mathbf{T}_{\beta_1} + \mathbf{P}_{-,2}\otimes\mathbf{P}_{-,1}\mathbf{T}_{\beta_1}^T + \mathbf{Q}_{+,2}\mathbf{T}_{\beta_2}^T\otimes\mathbf{Q}_{+,1} + \mathbf{Q}_{-,2}\mathbf{T}_{\beta_2}\otimes\mathbf{Q}_{-,1}\right)\mathbf{vec}(\mathbf{U}).$$

Note that the matrices $\mathbf{P}$ and $\mathbf{Q}$ are diagonal matrices representing the finite difference evaluations of the coefficients of the FDE. A simpler case occurs when the coefficients only depend on one spatial variable; for instance, the following data was used in [61]:

$$\left({}^{\mathrm{C}}_{0}D_t^{\alpha} - p_+ \, {}^{\mathrm{RL}}_{0}D_x^{\beta_1} - p_- \, {}^{\mathrm{RL}}_{x}D_1^{\beta_1} - q_+ \, {}^{\mathrm{RL}}_{0}D_y^{\beta_2} - q_- \, {}^{\mathrm{RL}}_{y}D_1^{\beta_2}\right) u(x,y,t) = 0,$$

with

$$p_+ = \Gamma(1.2)x^{\beta_1},\ p_- = \Gamma(1.2)(2-x)^{\beta_1},\ q_+ = \Gamma(1.2)y^{\beta_2},\ q_- = \Gamma(1.2)(2-y)^{\beta_2}.$$

After the Grünwald-Letnikov discretization, the system matrix $\mathbf{A}$ reads

$$\mathbf{A} = \mathbf{T}_{\alpha}^{n_t}\otimes\mathbf{I}^{n_2n_1} - \mathbf{I}^{n_t}\otimes\mathbf{I}^{n_2}\otimes\left(\mathbf{P}_+\mathbf{T}_{\beta_1} + \mathbf{P}_-\mathbf{T}_{\beta_1}^{\top}\right) - \mathbf{I}^{n_t}\otimes\left(\mathbf{Q}_+\mathbf{T}_{\beta_2} + \mathbf{Q}_-\mathbf{T}_{\beta_2}^{\top}\right)\otimes\mathbf{I}^{n_1},$$

where $\mathbf{T}_\beta$ is the one-sided derivative, $\mathbf{P}_+, \mathbf{P}_-, \mathbf{Q}_+, \mathbf{Q}_-$ are diagonal matrices with the grid values of p_+, p_-, q_+, q_-, respectively. Now, using

$$\hat{\mathbf{L}}_{\beta_1}^{n_x,P} = \mathbf{P}_+\mathbf{T}_{\beta_1} + \mathbf{P}_-\mathbf{T}_{\beta_1}^{\top},$$

we obtain the following form of the two-dimensional problem

$$\mathbf{A} = \mathbf{T}_{\alpha}\otimes\mathbf{I}^{n_1n_2} - \mathbf{I}^{n_t}\otimes\hat{\mathbf{L}}_{\beta_1}^{n_x,P}\otimes\mathbf{I}^{n_2} - \mathbf{I}^{n_t}\otimes\mathbf{I}^{n_1}\otimes\hat{\mathbf{L}}_{\beta_2}^{n_x,Q}.$$

In the more general case when the coefficients are sum of separable terms, that is,

$$p_+(x,y) = \sum_{j=1}^{r} p_{+,j}(x)p_{+,j}(y),$$

then the equation can be rewritten accordingly. For instance, assuming for simplicity that all coefficients are of order r, we obtain

$$\sum_{j=1}^{r}\left(\mathbf{P}_{+,2}^{(j)}\otimes\mathbf{P}_{+,1}^{(j)}\mathbf{T}_{\beta_1} + \mathbf{P}_{-,2}^{(j)}\otimes\mathbf{P}_{-,1}^{(j)}\mathbf{T}_{\beta_1}^T + \mathbf{Q}_{+,2}^{(j)}\mathbf{T}_{\beta_2}^T\otimes\mathbf{Q}_{+,1}^{(j)} + \mathbf{Q}_{-,2}^{(j)}\mathbf{T}_{\beta_2}\otimes\mathbf{Q}_{-,1}^{(j)}\right).$$

Our methodology can be applied to this case as well. An even more general form is obtained if all coefficients are of different form, and hence results in affine decompositions of different orders. For reasons of exposition we will not discuss this case further.

The structure of the linear systems in the variable coefficient case is very similar to the constant coefficient case, but also shows crucial differences. While the matrix $\mathbf{L}_{\beta_1}^{n_x}$ is a symmetric Toeplitz matrix, the matrix $\mathbf{L}_{\beta_1}^{n_x,P}$ is not symmetric and also without Toeplitz structure. Often we obtain a diagonal times a Toeplitz matrix, which will be exploited in our solution strategy.

ETNA
Kent State University
http://etna.math.kent.edu

4. Matrix equations solvers. This section is devoted to the introduction of the methodology that allows us to efficiently solve the problems presented in Section 3. We start our discussion by considering the constant coefficient case, and then we adapt the matrix structure to the (separable) variable coefficient case. We shall work with methods that exploit the presence of matrices and tensors as much as possible, so as to reduce the complexity of the problem. Moreover, we try to use the special "Toeplitz" structure whenever possible, either in the coefficient matrix or in its approximations. This is followed by a discussion of a tensor-based approach for the time-fractional problem with two space dimensions based on the highly popular TT toolbox [40, 41].

As a general remark for all solvers we are going to survey, we mention that they are all related to Krylov subspaces. Given a matrix $\mathcal{A}$ and a vector $\mathbf{r}_0$, the Krylov subspace of size l is defined as

$$\mathcal{K}_l(\mathcal{A}, \mathbf{r}_0) = \text{span}\left\{\mathbf{r}_0, \mathcal{A}\mathbf{r}_0, \ldots, \mathcal{A}^{l-1}\mathbf{r}_0\right\}.$$

As l increases, the space dimension grows, and the spaces are nested, namely $\mathcal{K}_l(\mathcal{A}, \mathbf{r}_0) \subseteq \mathcal{K}_{l+1}(\mathcal{A}, \mathbf{r}_0)$. In the following we shall also consider wide forms of generalizations of this original definition, from the use of matrices in place of $\mathbf{r}_0$, to the use of sequences of shifted and inverted matrices instead of $\mathcal{A}$.

4.1. Numerical solution of the Sylvester equation. The numerical solution of linear matrix equations of the form

$$\mathbf{AU} + \mathbf{UB} = \mathbf{G}, \tag{4.1}$$

with $\mathbf{A} \in \mathbb{R}^{n_A \times n_A}$, $\mathbf{B} \in \mathbb{R}^{n_B \times n_B}$, and $\mathbf{G} \in \mathbb{R}^{n_A \times n_B}$ arises in a large variety of applications; we refer the reader to [54] for a detailed description. A unique solution $\mathbf{U} \in \mathbb{R}^{n_A \times n_B}$ is ensured if $\mathbf{A}$ and $-\mathbf{B}$ have disjoint spectra. Robust numerical procedures for solving (4.1) when $\mathbf{A}$ and $\mathbf{B}$ have modest dimensions - up to a few hundreds - have been widely tested, and the Bartels-Stewart algorithm has emerged as the method of choice [3]. The method relies on a full Schur decomposition of the two matrices, and then on a backward substitution of the transformed problem.

We mention here another method that can be used when either $\mathbf{A}$ or $\mathbf{B}$ are small and already in triangular form, the way the equation (3.8) is in (3.6), when n_t is small. Partition $\mathbf{U}$ and $\mathbf{G}$ as $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{n_B}]$ and $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_{n_B}]$. Explicit inspection shows that if $\mathbf{B}$ is upper triangular, then $\mathbf{u}_1 \in \mathbb{R}^{n_A}$ can be obtained by solving the shifted system $(\mathbf{A}+\mathbf{B}_{11}\mathbf{I})\mathbf{u}_1 = \mathbf{g}_1$. All subsequent columns of $\mathbf{U}$ may be obtained with a backward substitution as

$$(\mathbf{A}+\mathbf{B}_{ii}\mathbf{I})\mathbf{u}_i = \mathbf{g}_i - \sum_{k=1}^{i-1} \mathbf{u}_k \mathbf{B}_{ki}, \quad i = 2, ..., n_B.$$

Each of these systems may be solved by CG equipped with a circulant preconditioner, as for (3.1). This strategy is appealing when n_B is small.

Generically, however, we consider the case where the coefficient matrices $\mathbf{A}$ and $\mathbf{B}$ are both extremely large, typically dense, making full spectral decompositions and backward solves prohibitively expensive both in terms of computational and memory requirements. On the other hand, $\mathbf{G}$ usually has much lower rank than the problem dimension, which makes low-rank approximation procedures more appealing. Note that the matrix $\mathbf{G}$ in our case reflects the right-hand side of the FDE and, as this often has a certain smoothness, we perform a truncated singular value decomposition to obtain a low-rank representation $\mathbf{G}$. Based on this

ETNA
Kent State University
http://etna.math.kent.edu

low-rank property, efficient methods seek an approximation $\widetilde{\mathbf{U}} = \mathbf{V}\mathbf{Y}\mathbf{W}^T \approx \mathbf{U}$, for matrices $\mathbf{V} \in \mathbb{R}^{n_A \times n_V}, \mathbf{Y} \in \mathbb{R}^{n_V \times n_W}, \mathbf{W} \in \mathbb{R}^{n_B \times n_W}$, where $n_V, n_W \ll n_A, n_B$. Among these approaches are Krylov subspace projection and ADI methods, the latter being a particular Krylov subspace method [54]. We consider the following general projection methods for (4.1): given two approximation spaces Range($\mathbf{V}$) and Range($\mathbf{W}$), an approximation $\widetilde{\mathbf{U}} = \mathbf{V}\mathbf{Y}\mathbf{W}^T$ is determined by requiring that the residual matrix $\mathbf{R} := \mathbf{A}\widetilde{\mathbf{U}} + \widetilde{\mathbf{U}}\mathbf{B} - \mathbf{G}$ satisfies[3]

$$\mathbf{V}^T\mathbf{R}\mathbf{W} = 0.$$

Assuming that both $\mathbf{V}$ and $\mathbf{W}$ have orthogonal columns, and using $\widetilde{\mathbf{U}} = \mathbf{V}\mathbf{Y}\mathbf{W}^T$, the condition above gives the reduced matrix equation

$$(\mathbf{V}^T\mathbf{A}\mathbf{V})\mathbf{Y} + \mathbf{Y}(\mathbf{W}^T\mathbf{B}\mathbf{W}) - \mathbf{V}^T\mathbf{G}\mathbf{W} = 0;$$

for $\mathbf{V}^T\mathbf{A}\mathbf{V}$ and $\mathbf{W}^T\mathbf{B}\mathbf{W}$ of small size, this equation can be efficiently solved by the Bartels-Stewart method, giving the solution $\mathbf{Y}$. Different choices of Range($\mathbf{V}$) and Range($\mathbf{W}$) lead to different approximate solutions. The quality of such an approximation depends on whether certain spectral properties of the coefficient matrices $\mathbf{A}$ and $\mathbf{B}$ are well represented in the two approximation spaces. Among the most successful choices are rational Krylov subspaces: letting $\mathbf{G} = \mathbf{G}_1\mathbf{G}_2^T$, rational Krylov subspaces generate the two spaces

$$\text{Range}(\mathbf{V}) = \text{Range}([\mathbf{G}_1, (\mathbf{A} - \sigma_1\mathbf{I})^{-1}\mathbf{G}_1, (\mathbf{A} - \sigma_2\mathbf{I})^{-1}\mathbf{G}_1, \ldots])$$

and

$$\text{Range}(\mathbf{W}) = \text{Range}([\mathbf{G}_2, (\mathbf{B}^T - \eta_1\mathbf{I})^{-1}\mathbf{G}_2, (\mathbf{B}^T - \eta_2\mathbf{I})^{-1}\mathbf{G}_2, \ldots]),$$

for specifically selected shifts σ_i, η_i, $i = 1, 2, \ldots$. Note that a shift σ of multiplicity k can be used, as long as terms with powers $(\mathbf{A} - \sigma\mathbf{I})^{-j}$, $j = 1, \ldots, k$ are included in the basis. In our numerical experience we found the choice $\sigma_i, \eta_i \in \{0, \infty\}$ particularly effective: for Range($\mathbf{V}$) this choice corresponds to an approximation space generated by powers of $\mathbf{A}$ and $\mathbf{A}^{-1}$ and it was first proposed under the name of *Extended Krylov subspace* [14]. In [53] it was shown for $\mathbf{B} = \mathbf{A}^T$ that such a space can be generated progressively as

$$\mathbb{EK}(\mathbf{A}, \mathbf{G}_1) = \text{Range}([\mathbf{G}_1, \mathbf{A}^{-1}\mathbf{G}_1, \mathbf{A}\mathbf{G}_1, \mathbf{A}^{-2}\mathbf{G}_1, \mathbf{A}^2\mathbf{G}_1, \mathbf{A}^{-3}\mathbf{G}_1, \ldots]),$$

and expanded until the approximate solution $\widetilde{\mathbf{U}}$ is sufficiently accurate. Note that in a standard implementation that sequentially generates $\mathbb{EK}(\mathbf{A}, \mathbf{G}_1)$, two "blocks" of new vectors are added at each iteration, one block multiplied by $\mathbf{A}$, and one "multiplied" by $\mathbf{A}^{-1}$. The block size depends on the number of columns in $\mathbf{G}_1$, although as the iteration proceeds this number could decrease, in case rank deficiency occurs. The implementation of the resulting projection method with $\mathbb{EK}(\mathbf{A}, \mathbf{G}_1)$ as approximation space was called KPIK in [53] for the Lyapunov matrix equation, that is (4.1) with $\mathbf{B} = \mathbf{A}^T$ and $\mathbf{G}_1 = \mathbf{G}_2$.

The procedure in [53] can be easily adapted to the case of general $\mathbf{B}$, so that also the space $\mathbb{EK}(\mathbf{B}^T, \mathbf{G}_2)$ is constructed [54]. For consistency we shall also call KPIK our implementation for the Sylvester equation. A MATLAB function implementing this algorithm is available at `www.dm.unibo.it/~simoncin/software.html`.

The effectiveness of the procedure can be measured in terms of both the dimension of the approximation space needed to achieve the required accuracy, as well as computational

[3] It can be shown that this requirement corresponds to an orthogonality (*Galerkin*) condition of the residual vector for the equation in Kronecker form, with respect to the space spanned by Range($\mathbf{W} \otimes \mathbf{V}$) [54].

ETNA
Kent State University
http://etna.math.kent.edu

time. The first issue, that is convergence of the approximate to the exact solution, was recently analyzed in [27] for KPIK applied to the Lyapunov equation, and in [4] as a particular case of rational Krylov space methods applied to the Sylvester equation; see Section 4.3. Regarding computational complexity, since each step of KPIK requires the solution of linear systems with coefficient matrices $\mathbf{A}$ and $\mathbf{B}$, an exact complexity analysis is highly problem dependent. Nevertheless, it is clear that there is a direct relation between the problem dimensions n_A, n_B and the computational time. In particular, for system dimensions as given in Section 5, a direct solution based on an LU decomposition of the system matrices is hardly possible. To alleviate this computation, in our implementation the inner systems with $\mathbf{A}$ and $\mathbf{B}$ are solved *inexactly*, that is by means of a preconditioned iterative method, with a sufficiently high accuracy so as to roughly maintain the KPIK rate of convergence expected with the exact (to machine precision) application of $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$. In our numerical experiments we shall call iKPIK the inexact version of the method. At this point we need to distinguish between the constant and the variable coefficient case. In the constant coefficient case we will inexactly solve for the Toeplitz matrices by using a circulant preconditioned CG method (see Section 4.2). In the case of the variable coefficient we can use the preconditioned GMRES approach that uses the local finite difference matrix for the preconditioner; see (4.3) in Section 4.2. Note that matrix vector products with Toeplitz matrices can be handled very efficiently; we recall this fact in the next section.

Finally, we observe that our stopping criterion for the whole procedure is based on the residual norm. In accordance with other low-rank approximation methods, the Frobenius norm of the residual matrix, namely $\|\mathbf{R}\|^2 = \sum_{i,j} \mathbf{R}_{ij}^2$, can be computed without explicitly storing the whole residual matrix $\mathbf{R}$. Indeed, using $\mathbf{G}_1 = \mathbf{V}\boldsymbol{\gamma}_1$ and $\mathbf{G}_2 = \mathbf{W}\boldsymbol{\gamma}_2$, we have

$$\mathbf{R} = [\mathbf{AV},\, \mathbf{V}] \begin{bmatrix} 0 & \mathbf{Y} \\ \mathbf{Y} & -\boldsymbol{\gamma}_1\boldsymbol{\gamma}_2^T \end{bmatrix} [\mathbf{B}^T\mathbf{W},\, \mathbf{W}]^T = \mathbf{Q}_1\boldsymbol{\rho}_1 \begin{bmatrix} 0 & \mathbf{Y} \\ \mathbf{Y} & -\boldsymbol{\gamma}_1\boldsymbol{\gamma}_2^T \end{bmatrix} \boldsymbol{\rho}_2^T\mathbf{Q}_2^T,$$

where the two skinny QR factorizations $[\mathbf{AV},\, \mathbf{V}] = \mathbf{Q}_1\boldsymbol{\rho}_1$ and $[\mathbf{B}^T\mathbf{W},\, \mathbf{W}] = \mathbf{Q}_2\boldsymbol{\rho}_2$ can be updated as the space expands[4]. Therefore,

$$\|\mathbf{R}\| = \left\| \boldsymbol{\rho}_1 \begin{bmatrix} 0 & \mathbf{Y} \\ \mathbf{Y} & -\boldsymbol{\gamma}_1\boldsymbol{\gamma}_2^T \end{bmatrix} \boldsymbol{\rho}_2^T \right\|,$$

whose storage and computational costs do not depend on the problem size, but only on the approximation space dimensions.

4.2. Computations with Toeplitz matrices. We briefly discuss the properties of Toeplitz matrices and possible solvers. This is needed for the simplest problem (3.3), but also within the Sylvester solvers presented earlier.

As Ng points out in [39], many direct solution strategies exist for the solution of Toeplitz systems that can efficiently solve these systems recursively. We mention here [1, 15, 18, 33] among others. One is nevertheless interested in finding iterative solvers for the Toeplitz matrices, as this further reduces the complexity. Additionally, as we want to use the Toeplitz solver within a possible preconditioner for the Sylvester equations, we are not necessarily interested in computing the solution to full accuracy. Note that, for convenience reasons, we simply use n for the dimensionality of the matrices in the general discussion following about Toeplitz matrices. It will be clear from the application and the discussion in Section 3 what the

[4]The residual norm computation could be made even cheaper in the exact case since then the skinny QR factorization would not have to be done explicitly [54].

ETNA
Kent State University
http://etna.math.kent.edu

specific value for n is. Let us consider a basic Toeplitz matrix of the form

$$\mathbf{T} = \begin{bmatrix} t_0 & t_{-1} & \dots & t_{2-n} & t_{1-n} \\ t_1 & t_0 & t_{-1} & & t_{2-n} \\ \vdots & t_1 & t_0 & \ddots & \vdots \\ t_{n-2} & & \ddots & \ddots & t_{-1} \\ t_{n-1} & t_{n-2} & \dots & t_1 & t_0 \end{bmatrix}. \tag{4.2}$$

Circulant matrices, which take the generic form

$$\mathbf{C} = \begin{bmatrix} c_0 & c_{n-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \end{bmatrix},$$

are special Toeplitz matrices as each column of a circulant matrix is a circulant shift of its preceding column.

It is well known that a circulant matrix $\mathbf{C}$ can be diagonalized using the Fourier matrix $\mathbf{F}$[5]. In more detail, the diagonalization of $\mathbf{C}$ is written as $\mathbf{C} = \mathbf{F}^H \mathbf{\Lambda} \mathbf{F}$, where $\mathbf{F}$ is again the Fourier matrix and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues (see [7, 39]). In order to efficiently compute the matrix-vector multiplication with $\mathbf{C}$, the matrix-vector multiplication using $\mathbf{F}$ and $\mathbf{\Lambda}$ needs to be available. The evaluation of $\mathbf{F}$ and $\mathbf{F}^H$ times a vector can be done via the Fast Fourier Transform (FFT, [8, 17]). The computation of the diagonal elements of $\mathbf{\Lambda}$ is done employing one more FFT. Overall, this means that the matrix vector multiplication with $\mathbf{C}$ can be replaced by applications of the FFT.

The $n \times n$ Toeplitz matrices (4.2) are not circulant but can be embedded into a $2n \times 2n$ circulant matrix as follows

$$\begin{bmatrix} \mathbf{T} & \mathbf{B} \\ \mathbf{B} & \mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix},$$

with

$$\mathbf{B} = \begin{bmatrix} 0 & t_{n-1} & \dots & t_2 & t_1 \\ t_{1-n} & 0 & t_{n-1} & & t_2 \\ \vdots & t_{1-n} & 0 & \ddots & \vdots \\ t_{-2} & & \ddots & \ddots & t_{n-1} \\ t_{-1} & t_{-2} & \dots & t_{1-n} & 0 \end{bmatrix}.$$

This new structure allows one to exploit the FFT in matrix-vector multiplications with $\mathbf{T}$.

Note that these techniques can also be used when the variable coefficient case is considered, as multiplications with

$$\mathbf{I} - \hat{\mathbf{L}}_{\beta_1}^{n_x,P} = \mathbf{I} - \left(\mathbf{P}_+ \mathbf{T}_{\beta_1} + \mathbf{P}_- \mathbf{T}_{\beta_1}^\top\right)$$

can be performed using FFTs and simple diagonal scaling with $\mathbf{P}_-$ and $\mathbf{P}_+$.

[5] In MATLAB this matrix can be computed via `F=fft(eye(n))`.

ETNA
Kent State University
http://etna.math.kent.edu

For the constant coefficient case, i.e., the Toeplitz case, there exists a variety of different preconditioners [39]. Here we focus on a classical circulant preconditioner introduced by Strang in [57]. The idea is to approximate the Toeplitz matrix $\mathbf{T}$ by a circulant $\mathbf{C}$ that can in turn be easily inverted by means of the FFT machinery. The diagonals c_j of this $\mathbf{C}$ are determined as

$$c_j = \begin{cases} t_j, & 0 \le j \le \lfloor n/2 \rfloor \\ t_{j-n}, & \lfloor n/2 \rfloor < j < n \\ c_{n+j} & 0 < -j < n \end{cases}.$$

Here $k := \lfloor n/2 \rfloor$ is the largest integer less or equal than $n/2$. Note that other circulant approximations are possible but will not be discussed here [39].

The computational strategy described above can be used to solve the linear system in (3.3) associated with Problem 1, namely

$$\left(\mathbf{I}^{n_x} - \tau \mathbf{L}_\beta^{n_x}\right) \mathbf{u}^{n+1} = \mathbf{u}^n + \mathbf{f}.$$

In [62] it was shown that the coefficient matrix $\mathbf{I}^{n_x} - \tau \mathbf{L}_\beta^{n_x}$ is a strictly diagonally dominant M-matrix. This allows us to use a symmetric Krylov subspace solver such as the Conjugate Gradient method (CG) [23], which requires matrix-vector products with the coefficient matrix; for more details on iterative Krylov subspace solvers we refer the reader to [22, 49, 55]. The coefficient matrix has Toeplitz structure. Therefore the circulant approximation $\mathbf{C} \approx \mathbf{I}^{n_x} - \tau \mathbf{L}_\beta^{n_x}$ can be used as a preconditioner for CG. For the fast convergence of CG it is sufficient that the eigenvalues of the preconditioned matrix form a small number of tiny clusters, which is known to be the case for the Strang circulant preconditioner, since it gives a single eigenvalue cluster around 1 [32].

The numerical treatment is more complicated in the variable coefficient case. The authors of [13] point out that the use of circulant preconditioners for this case is not as effective, and suggest the use of the following approximation

$$-\hat{\mathbf{L}}_{\beta_1}^{n_x,P} = -(\mathbf{P}_+ \mathbf{T}_{\beta_1} + \mathbf{P}_- \mathbf{T}_{\beta_1}^\top) \approx \mathbf{P}_+ \mathbf{A} + \mathbf{P}_- \mathbf{A}^\top =: \mathbf{W}, \tag{4.3}$$

where $\mathbf{W} \approx -\hat{\mathbf{L}}_{\beta_1}^{n_x,P}$ is used for preconditioning purposes whenever a system involving $\hat{\mathbf{L}}_{\beta_1}^{n_x,P}$ needs to be solved and

$$\mathbf{A} = \frac{1}{h_x^{\beta_1}} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix},$$

which is simply the central difference approximation to the 1D Laplacian. This approximation is good when the fractional differentiation parameter is close to two, i.e., ≥ 1.5. Alternatively, for smaller differentiation parameters one could use

$$\mathbf{A} = \frac{1}{h_x^{\beta_1}} \begin{bmatrix} 1 & -1 & & \\ & 1 & \ddots & \\ & \ddots & \ddots & -1 \\ & & & 1 \end{bmatrix},$$

ETNA
Kent State University
http://etna.math.kent.edu

as suggested in [13]. We can use a nonsymmetric Krylov solver such as GMRES [50] where the matrix multiplication is performed using the FFT based approach presented above, and the tridiagonal matrix $\mathbf{P}_+\mathbf{A}+\mathbf{P}_-\mathbf{A}^\top$ as preconditioner by means of its LU factorization. One of the major concerns is carrying the approaches for the one-dimensional case over to the higher dimensional settings. The techniques for matrix equations presented earlier are well suited for this scenario. We are in particular focusing on the connection to matrix equation solvers for which we study the convergence behaviour in the next section. In addition we mention the possible use of low-rank versions [30, 56] of well-known Krylov subspace solvers that have become popular as they allow for the solution of tensor-valued equations while maintaining the low-rank nature of the solution. This approach is certainly applicable in our case when combined with a suitable preconditioner. For more detailed discussion we refer to [30].

4.3. Considerations on the convergence of the iterative solvers in the constant coefficient case. The performance of the iterative methods discussed so far depends, in the symmetric case, on the distribution of the eigenvalues of the coefficient matrices, and, in the nonsymmetric case, on more complex spectral information such as the field of values. We start with providing a simple but helpful bound on the spectrum of the matrix obtained after discretizing the fractional derivatives.

LEMMA 1. *For $1 < \beta < 2$, the spectrum of the matrix $\mathbf{L}_\beta := \frac{1}{2}(\mathbf{T}_\beta + \mathbf{T}_\beta^T)$ is contained in the open interval $(-2h^{-\beta}\beta, 0)$.*

Proof. From [32], for $1 < \beta < 2$, we recall the following useful properties of $g_{\beta,k}$:

$$g_{\beta,0} = 1,\ g_{\beta,1} = -\beta < 0,\ g_{\beta,2} > g_{\beta,3} > \cdots > 0, \quad \sum_{k=0}^{\infty} g_{\beta,k} = 0,\ \sum_{k=0}^{n} g_{\beta,k} < 0,\ \forall n \geq 1.$$

We can now adapt the proof in [32] to get an estimate of the eigenvalues of $\mathbf{L}_\beta = \frac{1}{2}(\mathbf{T}_\beta + \mathbf{T}_\beta^T)$. Recall the structure of $\mathbf{T}_\beta$:

$$\mathbf{T}_\beta = h^{-\beta} \begin{bmatrix} g_{\beta,1} & g_{\beta,0} & & & & & 0 \\ g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & & & & \\ g_{\beta,3} & g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & & & \\ \vdots & \ddots & g_{\beta,2} & g_{\beta,1} & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & g_{\beta,0} & 0 \\ \vdots & \ddots & \ddots & \ddots & g_{\beta,2} & g_{\beta,1} & g_{\beta,0} \\ g_{\beta,n} & g_{\beta,n-1} & \cdots & \cdots & & g_{\beta,2} & g_{\beta,1} \end{bmatrix}.$$

Hence, the Gershgorin circles of $\mathbf{L}_\beta$ are all centered at $h^{-\beta}g_{\beta,1} = -\beta h^{-\beta}$. Moreover, the largest radius is obtained in row $\lfloor \frac{n}{2} + 1 \rfloor$ and thus is at most (depending on n being odd or even)

$$r_{\max} = h^{-\beta} \sum_{k=0,k\neq 1}^{\lfloor \frac{n}{2}+1 \rfloor} g_{\beta,k} < -h^{-\beta} g_{\beta,1} = h^{-\beta}\beta.$$

This now implies $\sigma(\mathbf{L}_\beta) \subset (-2h^{-\beta}\beta, 0)$. ∎

The result shows that, as the spatial mesh is refined, the eigenvalues of $\mathbf{L}_\beta$ spread out on the negative half of the real line, and preconditioning is needed for a fast convergence of the iterative scheme for (3.1). The eigenvalue properties are also crucial to assess the performance of the Extended Krylov subspace method described in the previous section, which we use both

ETNA
Kent State University
http://etna.math.kent.edu

for (3.4) and (3.6) as a solver. Assume that $\mathbf{G} = \mathbf{g}_1\mathbf{g}_2^T$. In [4], for $\mathbf{A}$ and $\mathbf{B}$ symmetric, it was shown that the residual Frobenius norm is bounded as

$$\frac{\|\mathbf{R}\|_F}{\|\mathbf{g}_1\|\,\|\mathbf{g}_2\|} \leq 4\max\{\gamma_{m_\mathbf{A},\mathbf{A},\mathbf{B}}, \gamma_{m_\mathbf{B},\mathbf{B},\mathbf{A}}\} + \xi(\gamma_{m_\mathbf{A},\mathbf{A},\mathbf{B}} + \gamma_{m_\mathbf{B},\mathbf{B},\mathbf{A}}),$$

where $\gamma_{m_\mathbf{A},\mathbf{A},\mathbf{B}}$ and $\gamma_{m_\mathbf{B},\mathbf{B},\mathbf{A}}$ are quantities associated with the approximation spaces in $\mathbf{A}$ and in $\mathbf{B}$ of dimension $m_\mathbf{A}$ and $m_\mathbf{B}$, respectively, and ξ is an explicit constant independent of the approximation space dimensions. For $\mathbf{A} = \mathbf{B}$, which in (3.6) is obtained for instance whenever $\beta_1 = \beta_2$, it holds (see [27])

$$\gamma_{m_\mathbf{A},\mathbf{A},\mathbf{B}} = \gamma_{m_\mathbf{B},\mathbf{B},\mathbf{A}} = \left(\frac{\kappa^{1/4}-1}{\kappa^{1/4}+1}\right)^{2m_\mathbf{A}}, \qquad \kappa = \lambda_{\max}(\mathbf{A})/\lambda_{\min}(\mathbf{A}).$$

In the notation of (3.6), the rate of convergence of the Extended Krylov subspace method when $\beta_1 = \beta_2$ is thus driven by $\kappa^{1/4}$, where κ is the condition number of $\frac{1}{2}\mathbf{I} - L_\beta$.

Finally, if one were to consider the (symmetric) Kronecker formulation of the Sylvester equation in (3.8), that is,

$$(\mathbf{I}_1 \otimes \mathbf{A} + \mathbf{B} \otimes \mathbf{I}_2)\mathbf{vec}(\mathbf{U}) = \mathbf{vec}(\mathbf{G}),$$

the following estimate for the eigenvalues of the coefficient matrix would hold:

$$\sigma(\mathbf{I}_1 \otimes \mathbf{A} + \mathbf{B} \otimes \mathbf{I}_2) \subset \left(1, 1 + \frac{2\beta_1\tau}{h^{\beta_1}} + \frac{2\beta_2\tau}{h^{\beta_2}}\right).$$

The assertion follows from the fact that the eigenvalues ν of $\mathbf{I}_1 \otimes \mathbf{A} + \mathbf{B} \otimes \mathbf{I}_2$ are given by the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$ via $\nu_{i,j} = \lambda_i(\mathbf{A}) + \mu_j(\mathbf{B})$. Therefore, if the Conjugate Gradient method were applied to the Kronecker formulation above, classical results would ensure that its convergence would asymptotically depend on $(1 + \frac{2\beta_1\tau}{h^{\beta_1}} + \frac{2\beta_2\tau}{h^{\beta_2}})^{\frac{1}{2}}$, which can be significantly larger than the quantity $\kappa^{\frac{1}{4}}$ obtained with the matrix equation formulation. As a consequence, solving with the Conjugate Gradient method is expected to be much slower than with the matrix-oriented approach.

4.4. Tensor-valued equation solver. The efficient solution of tensor-valued equations has recently seen much progress regarding the development of effective methods and their corresponding analysis [2, 40, 59]. General introductions to this very active field of research can be found in [21, 29] and in the literature mentioned there. A crucial aspect in tensor-based strategies is deciding on the format employed to represent tensors, for which different algorithms can be exploited; here we focus on the tensor train (TT) representation [40, 41, 44]. While for a three-dimensional setup other formats such as the well known Tucker format [21, 29] can be more efficient than the TT format, this property is lost in higher dimensions, which is the case when we consider three spatial dimensions plus time or additional parameter dimensions. Additionally, the availability of a suitable software implementation of the used algorithms via the TT toolbox [42] is an advantage of the tensor train format. In the following we shall briefly introduce the TT format, while we point to the literature for details.

The first operation we need for high-dimensional data is reshaping. To this end, suppose $\mathbf{u}$ is the solution of (3.9). Its elements can be naturally enumerated by three indices i_1, i_2, i_3, corresponding to the discretization in time and the two spatial dimensions, respectively. Introducing a *multi-index*

$$\overline{i_1 i_2 i_3} = (i_1 - 1)n_t n_x + (i_2 - 1)n_x + i_3,$$

ETNA
Kent State University
http://etna.math.kent.edu

we can denote $\mathbf{u} = \left[\mathbf{u}(\overline{i_1 i_2 i_3})\right]_{i_1,i_2,i_3=1}^{n_t,n_x,n_y}$, and consider $\mathbf{u}$ as a three-dimensional *tensor* with elements $\mathbf{u}(i_1,i_2,i_3)$. The TT decomposition aims to approximate $\mathbf{u}$ as follows,

$$\mathbf{u}(i_1,i_2,i_3) \approx \sum_{s_1,s_2=1}^{r_1,r_2} \mathbf{u}^{(1)}_{s_1}(i_1)\mathbf{u}^{(2)}_{s_1,s_2}(i_2)\mathbf{u}^{(3)}_{s_2}(i_3) \;\Leftrightarrow\; \mathbf{u} \approx \sum_{s_1,s_2=1}^{r_1,r_2} \mathbf{u}^{(1)}_{s_1} \otimes \mathbf{u}^{(2)}_{s_1,s_2} \otimes \mathbf{u}^{(3)}_{s_2}. \tag{4.4}$$

The indices r_1, r_2 are called TT ranks and the $\mathbf{u}^{(m)}$, $m = 1,2,3$ are the so-called TT blocks, with $\mathbf{u}^{(1)} \in \mathbb{R}^{n_t \times r_1}$, $\mathbf{u}^{(2)} \in \mathbb{R}^{r_1 \times n_x \times r_2}$ and $\mathbf{u}^{(3)} \in \mathbb{R}^{r_2 \times n_y}$. When fixing the indices we get for $\mathbf{u}^{(2)}(j) \in \mathbb{R}^{r_1 \times r_2}$ a matrix slice, for $\mathbf{u}^{(2)}_{s_1,s_2} \in \mathbb{R}^{n_x}$ a vector, and for $\mathbf{u}^{(2)}_{s_1,s_2}(j)$ a scalar. The values of r_1, r_2 depend on the accuracy enforced in (4.4).

Let us next represent the (tensorized) linear operator (3.9) in the proposed TT format. We recall that $\boldsymbol{\mathcal{A}}$ is given by

$$\begin{aligned}\boldsymbol{\mathcal{A}} &= \mathbf{T}_\alpha^{n_t} \otimes \mathbf{I}^{n_x} \otimes \mathbf{I}^{n_y} - \mathbf{I}^{n_t} \otimes \left(\frac{1}{h^{\beta_1}} \mathbf{I}^{n_y} \otimes \mathbf{L}_{\beta_1}^{n_x} + \mathbf{L}_{\beta_2}^{n_y} \otimes \frac{1}{h^{\beta_2}} \mathbf{I}^{n_x} \right) \\ &= (\mathbf{T}_\alpha^{n_t} \otimes \mathbf{I}^{n_x} \otimes \mathbf{I}^{n_y}) - \left(\mathbf{I}^{n_t} \otimes \frac{1}{h^{\beta_1}} \mathbf{I}^{n_y} \otimes \mathbf{L}_{\beta_1}^{n_x} \right) - \left(\mathbf{I}^{n_t} \otimes \mathbf{L}_{\beta_2}^{n_y} \otimes \frac{1}{h^{\beta_2}} \mathbf{I}^{n_x} \right).\end{aligned}$$

For applying the tensor $\boldsymbol{\mathcal{A}}$ to a tensor in TT format, we use the following approximation of $\boldsymbol{\mathcal{A}}$ (see [10])

$$\boldsymbol{\mathcal{A}}(i,j) = \boldsymbol{\mathcal{A}}(i_1 i_2 i_3, j_1 j_2 j_3) \approx \sum_{\gamma_1,\gamma_2=1}^{r_1,r_2} A^{(1)}_{\gamma_1}(i_1,j_1) A^{(2)}_{\gamma_1,\gamma_2}(i_2,j_2) A^{(3)}_{\gamma_2}(i_3,j_3). \tag{4.5}$$

Assume that the right-hand side of the equation to be solved is given in the TT format (4.4), and that the matrix product operator uses the structure in (4.5). Then a suitable approximate solution to the FDE (3.9) will also be sought in the same TT format. For convenience we do not introduce additional notation but assume the operations discussed in the following are all within the TT framework. Again, similar reasonings would be possible using other tensor formats.

A way to approximate the solution of the problem defined by (3.9) is to first recast it as a least squares problem, that is,

$$\left\| \tilde{\mathbf{f}} - \boldsymbol{\mathcal{A}}\mathbf{u} \right\| \Rightarrow \min. \tag{4.6}$$

A simple class of techniques for solving (4.6) is the so-called *Alternating Least Squares (ALS)* scheme where the minimization proceeds by fixing all but one core of the optimization problem, and then solving this simpler problem while repeatedly iterating over all cores. In [43] several issues associated with this basic ALS approach are discussed, among them its slow convergence. An improved scheme is the so-called DMRG (Density Matrix Renormalization Group) method [24, 40], where all but two cores are fixed and the alternating optimization proceeds iterating over these pairs of cores. In our experiments we have used the more recent approach AMEN (Alternating Minimal Energy) method [12]. This algorithm is an improvement on ALS schemes with steepest descent, in the sense that it uses computed information in a more timely way and can work with local operations using only one or two cores. For a detailed discussion and comparisons, we refer to [12]. We also mention that one could use a TT version of well-known Krylov methods such as GMRES ([50]) adapted to the TT framework (see [9]), or further exploit the Toeplitz structure of the involved matrices (see the QTT framework [10, 11, 26]). We have not explored these possibilities in this work.

ETNA
Kent State University
http://etna.math.kent.edu

The existence of low-tensor-rank solutions for fractional differential equations has, to the best of our knowledge, not received any attention, and will be the study of further work. In spirit, the closest results are found in [20], where high-dimensional Laplacians are discussed. For more results we refer to [52, 60].

5. Numerical results. In this section we report on our numerical experience with the algorithms proposed in the previous sections. All tests are performed on a Linux Ubuntu Compute Server using 4 Intel Xeon E7-8837 CPUs running at 2.67 GHz, each equipped with 256 GB of RAM using MATLAB® 2012.

We illustrate the effectiveness of our proposed methods by testing the convergence for all problems presented earlier both in the constant and variable coefficient case. Our goal is to obtain robustness with respect to the discretization parameter in both the temporal and the spatial dimensions. Additionally, we are testing all problems for a variety of different orders of differentiation to illustrate that the methods are suitable for reasonable parameter choices.

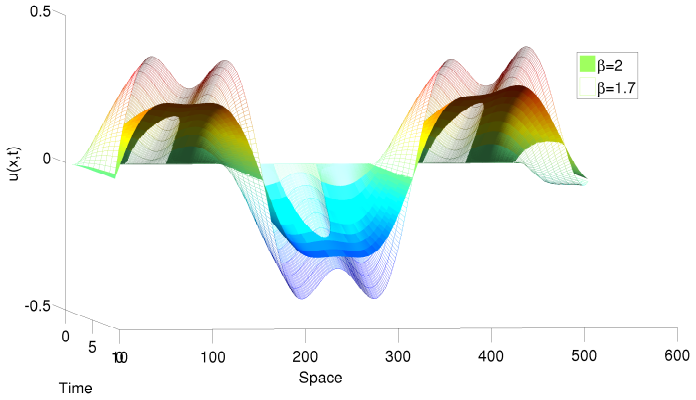


FIG. 5.1. *Comparison of numerical solutions when using two different values for β with zero initial condition and zero Dirichlet boundary condition (*510 *and* 10 *space and time points, resp.).*

As this work discusses various different formulations and several solution strategies, we want to summarize the suggested solution methods in Table 5.1.

TABLE 5.1
Overview of the solvers. Iterative solution in the constant coefficient case always uses circulant preconditioned CG*; in the variable coefficient case a preconditioned nonsymmetric solver.*

Problem setup	Solution method
Time+1D FDE (3.3)	PCG
Time+2D FDE (3.5)	IKPIK
Frac.Time+1D FDE (3.7)	IKPIK
Frac.Time+2D FDE (3.9)	AMEN

5.1. Fractional in space: constant and variable coefficients.

One-dimensional example. We are using a zero-initial condition and a zero Dirichlet boundary condition. The result for two different values of β is shown in Figure 5.1 and is simply given to illustrate that the parameter β has a significant influence on the solution u. Table 5.2 shows the result for a variety of discretization levels with two values of β. For this problem the forcing term was given by

ETNA
Kent State University
http://etna.math.kent.edu

$$f = 80\sin(20x)\cos(10x).$$

It can be seen that the Strang preconditioner [57] performs exceptionally well with only 6 to 8 iterations needed for convergence and no dependence on the mesh size. A tolerance of 10^{-6} for the relative residual was used for the stopping criterion. The time-step is chosen to be $\tau = h_x/2$.

For the variable coefficient case we consider the following parameters $p_+ = \Gamma(1.2)x_1^{\beta_1}$ and $p_- = \Gamma(1.2)(2-x_1)^{\beta_1}$. It is easily seen from Table 5.2 that the iteration numbers are very robust using the approximation by the difference matrices, and that timings remain very moderate.

TABLE 5.2
Average PCG (constant coefficient case) and GMRES (variable coefficient case) iteration numbers for 8 time steps and two values of β, as the mesh is refined; in parenthesis is the total CPU time (in secs).

	Constant Coeff.		Variable Coeff.	
n_x	$\beta = 1.3$	$\beta = 1.7$	$\beta = 1.3$	$\beta = 1.7$
32768	6.0 (*0.43*)	7.0 (*0.47*)	6.8 (*0.90*)	6.0 (*0.81*)
65536	6.0 (*0.96*)	7.0 (*0.97*)	6.4 (*1.93*)	6.0 (*1.75*)
131072	6.0 (*1.85*)	7.0 (*2.23*)	5.9 (*3.93*)	5.9 (*3.89*)
262144	6.0 (*7.10*)	7.1 (*8.04*)	5.4 (*12.78*)	6.0 (*13.52*)
524288	6.0 (*15.42*)	7.8 (*19.16*)	5.1 (*25.71*)	6.0 (*27.40*)
1048576	6.0 (*34.81*)	8.0 (*41.76*)	4.9 (*51.02*)	6.3 (*62.57*)

Two-dimensional example. This problem now includes a second spatial dimension together with a standard derivative in time. The spatial domain is the unit square. The problem is characterized by a zero-Dirichlet condition with a zero-initial condition. The time-dependent forcing term is given as

$$F = 100\sin(10x)\cos(y) + \sin(10t)xy$$

and the solution for this setup is illustrated in Figure 5.2, where we show the solution at two different time steps.

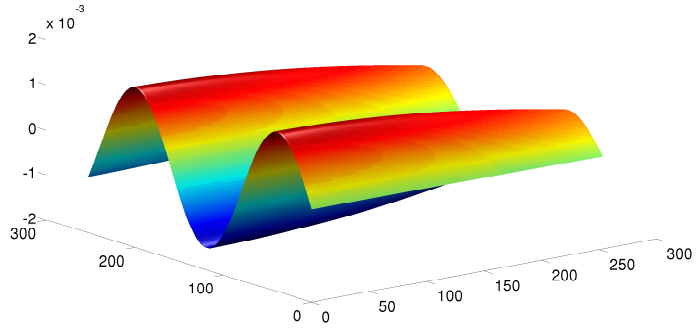
Table 5.3 shows the average IKPIK iteration numbers alongside the total computing time. The tolerance for IKPIK is set to 10^{-6}, to illustrate that we can compute the solution very accurately, and the tolerance for the inner iterative solver is chosen to be 10^{-12}. The results summarized in Table 5.3 give the average number of IKPIK iterations for 8 time-steps. The variable coefficient is given by

$$p_+ = \Gamma(1.2)x^{\beta_1}, p_- = \Gamma(1.2)(2-x)^{\beta_1}, q_+ = \Gamma(1.2)y^{\beta_2}, q_- = \Gamma(1.2)(2-y)^{\beta_2}.$$

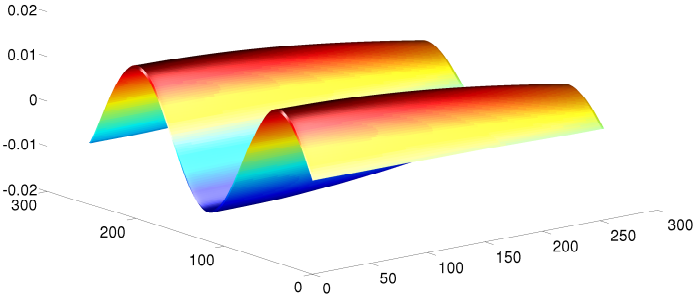
Note that the largest problem given in Table 5.3 corresponds to $16,777,216$ unknowns, which is no problem for our methodology, but a further increase would not make it possible to store the unknowns without being represented in low-rank format.

5.2. Time and space fractional. Additionally we consider the challenging case when also the temporal derivative is of fractional type.

ETNA
Kent State University
http://etna.math.kent.edu



a) First time-step



b) Tenth time-step

FIG. 5.2. *First and tenth time-step solutions of the* $2D$ *fractional differential equation.*

TABLE 5.3
The solver IKPIK *is presented, for a variety of meshes and two different values of* β *in both the constant and the variable coefficient cases. Shown are the iteration numbers and the total CPU time.*

		Variable Coeff.		Constant Coeff.	
n_x	n_y	$\beta_1 = 1.3$ $\beta_2 = 1.7$ it (CPUtime)	$\beta_1 = 1.7$ $\beta_2 = 1.9$ it (CPUtime)	$\beta_1 = 1.3$ $\beta_2 = 1.7$ it (CPUtime)	$\beta_1 = 1.7$ $\beta_2 = 1.9$ it (CPUtime)
1024	1024	2.3 (*19.35*)	2.5 (*15.01*)	2.9 (*9.89*)	2.5 (*18.51*)
1024	2048	2.8 (*47.17*)	2.9 (*22.25*)	3.0 (*23.07*)	3.2 (*22.44*)
2048	2048	3.0 (*76.72*)	2.6 (*36.01*)	2.0 (*51.23*)	2.3 (*34.43*)
4096	4096	3.0 (*171.30*)	2.6 (*199.82*)	2.0 (*164.20*)	2.2 (*172.24*)

One spatial dimension. Here the forcing term was defined by $f = 8\sin(10x)$, while a zero Dirichlet condition on the spatial interval was used.

Table 5.4 summarizes the results for IKPIK. Here we discretized both the temporal and the spatial domain using the same number of elements. For the variable coefficient case we use the finite difference approximation presented earlier. We noticed that the convergence of the outer IKPIK solver was dependent on the accuracy. We did not observe this dependence

ETNA
Kent State University
http://etna.math.kent.edu

for the results shown in Table 5.3 earlier. Due to the effectiveness of the preconditioner we set a tight tolerance for the inner scheme of 10^{-13}, and use a tolerance of 10^{-6} for the IKPIK convergence. When stating the degrees of freedom for each dimension one has to note that implicitly this method is solving a linear system of tensor form that has the dimensionality $n_t n_x \times n_t n_x$ so for the largest example shown in Table 5.4 this leads to roughly 70 million unknowns. We see that in the constant coefficient case the method performs very robustly

TABLE 5.4
IKPIK, *for a variety of meshes and different values of α and β. Shown are the iteration numbers, the elapsed CPU time, and the maximum iteration numbers for the inner solver (MI).*

		Variable Coeff.		
n_t	n_x	$\beta = 1.7$ $\alpha = 0.5$ it (CPUtime, MI)	$\beta = 1.1$ $\alpha = 0.9$ it (CPUtime, MI)	$\beta = 1.9$ $\alpha = 0.2$ it (CPUtime, MI)
1024	1024	10 (*0.33, 31*)	43 (*1.87, 39*)	4 (*0.14, 15*)
2048	2048	11 (*0.59, 33*)	57 (*4.92, 48*)	4 (*0.21, 15*)
4096	4096	13 (*2.64, 36*)	74 (*30.52, 60*)	4 (*0.54, 15*)
8192	8192	14 (*5.51, 39*)	95 (*91.13, 75*)	4 (*1.31, 16*)
		Constant Coeff.		
n_t	n_x	$\beta = 1.7$ $\alpha = 0.5$ it (CPUtime, MI)	$\beta = 1.1$ $\alpha = 0.9$ it (CPUtime, MI)	$\beta = 1.9$ $\alpha = 0.2$ it (CPUtime, MI)
1024	1024	6 (*0.17, 18*)	14 (*0.39, 15*)	4 (*0.13, 17*)
2048	2048	6 (*0.27, 18*)	16 (*0.79, 18*)	4 (*0.21, 17*)
4096	4096	6 (*0.65, 20*)	18 (*2.52, 19*)	4 (*0.45, 17*)
8192	8192	7 (*2.43, 20*)	20 (*6.51, 20*)	4 (*1.25, 20*)

with respect to mesh-refinement and that also the iteration numbers needed for IKPIK are of moderate size when the differentiation parameter is changed. From the results for the variable coefficient case it can be seen that

$$-\hat{\mathbf{L}}_{\beta_1}^{n_x,P} \approx \mathbf{P}_+\mathbf{A} + \mathbf{P}_-\mathbf{A}^\top =: \hat{\mathbf{P}}_{\beta_1}^{n_x,P}$$

is not always equally effective in this setup. The eigenvalues of $(\hat{\mathbf{P}}_{\beta_1}^{n_x,P})^{-1}\hat{\mathbf{L}}_{\beta_1}^{n_x,P}$ are shown in Figure 5.3.

In Figure 5.4 we illustrate the performance of both the preconditioner and the IKPIK method for a variety of differentiation parameters in the variable coefficient setup. It can be seen that IKPIK often only needs a small number of iterations whereas the preconditioner is not always as effective as the one in the constant coefficient case.

Two and three dimensions. In this section we briefly illustrate how the tensorized problems can be solved. Our implementation is based on the recently developed tensor-train format [40, 41, 44]. The forcing term in the first three-dimensional setup is given by the rank-one tensor

$$\tilde{\mathbf{f}} = \mathbf{1} \otimes \exp((\mathbf{x} - 0.5)^2) \otimes \exp((\mathbf{y} - 0.5)^2) \otimes \exp((\mathbf{z} - 0.5)^2),$$

where the terms $\exp((\star - 0.5)^2)$ are to be understood componentwise with $\mathbf{x}, \mathbf{y}, \mathbf{z}$ the vectors representing the mesh-grid of appropriate dimensionality. Furthermore, zero-Dirichlet conditions are imposed, together with a zero initial condition. The approximations of the

ETNA
Kent State University
http://etna.math.kent.edu

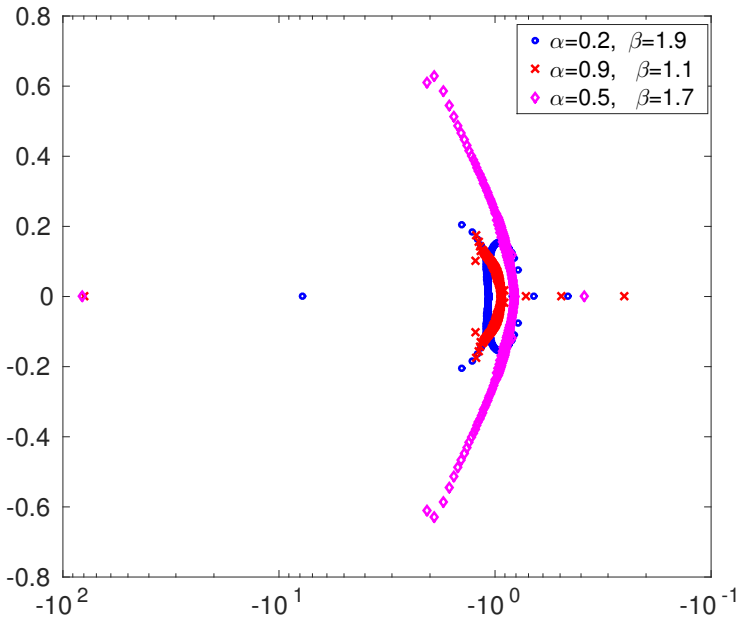


FIG. 5.3. *Eigenvalues shown for the three setups presented in Table 5.4.*

right-hand side $\tilde{\mathbf{f}}$ and the tensor $\mathbf{A}$ use the `round` function within the TT-toolbox, with "round" tolerance set to 10^{-6}. The AMEN method recalled in Section 4.4 ([12]) was used, with a convergence tolerance also set to 10^{-6}. We next illustrate that the performance of AMEN for our tensor equation with constants coefficients is robust with respect to changes in the system parameters such as the orders of differentiation and varying meshes. Table 5.5 shows the AMEN iteration numbers for 3 different mesh sizes both in time and space and two different choices of differentiation orders. The iteration numbers are relatively robust with changing mesh-size, while they show some dependence on the choice of the differentiation orders. We stress that the solvers were not tailored to these specific cases, rather they were used as black boxes. We note that the storage of the full solution to the problem presented in Table 5.5 would require $n_t n_x n_y n_z$ elements, which for the finest discretization shown is $2^{44} \approx 10^{13}$ elements. The low-rank solution in the TT format requires storage for $r_0 r_1 n_t + r_1 r_2 n_x + r_2 r_3 n_y + r_3 r_4 n_z$ elements. We then obtain

$$\frac{\text{mem}(TT)}{\text{mem}(FULL)} = \frac{60 \cdot 2^{11}}{2^{44}} \approx 6 \cdot 10^{-9},$$

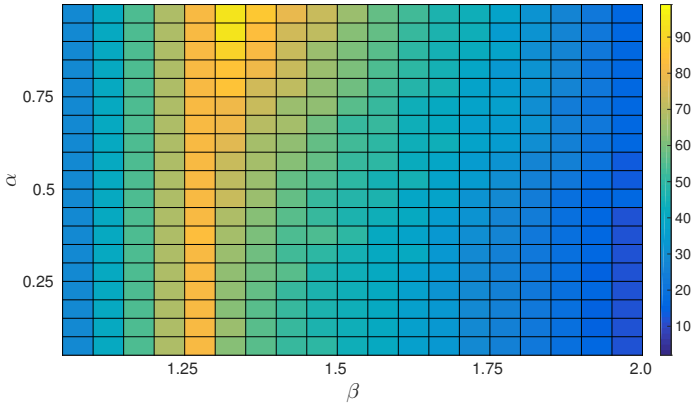
which illustrates the potential of the low tensor-rank method. Figure 5.5 shows the computed solution to the above mentioned problem in three-dimensional space.

We also want to illustrate the performance of the tensor method for the use with a variable coefficient setup as presented earlier. For this we consider the coefficients

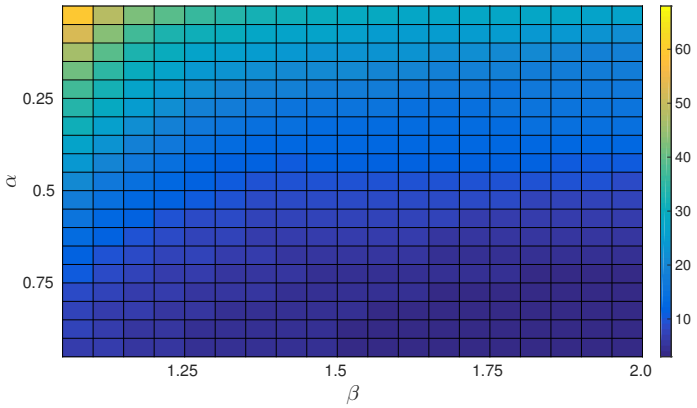
$$p_+ = \Gamma(1.2)x^{\beta_1}y,\ p_- = \Gamma(1.2)(2-x)^{\beta_1}y,\ q_+ = \Gamma(1.2)y^{\beta_2}x,\ q_- = \Gamma(1.2)(2-y)^{\beta_2}x.$$

The forcing term is given by $\tilde{\mathbf{f}} = \mathbf{1} \otimes \sin(2\pi\mathbf{x}) \otimes \exp((\mathbf{y}-0.5)^2)$, while zero Dirichlet boundary conditions are imposed, together with a zero initial condition. We show the results

ETNA
Kent State University
http://etna.math.kent.edu



a) GMRES iterations



b) KPIK iterations

FIG. 5.4. *Results for a mesh* 1024 *degrees of freedom both in space and time. We put a mesh of width* 0.05 *on both the differentiation parameter in time and space.* $\alpha \in [0.05, 1], \beta \in [1.05, 2]$. *We switch between preconditioners at* $\beta = 1.3$.

TABLE 5.5
Performance of AMEN *as the mesh is refined, for different values of the differentiation orders.*

				$\beta_2 = 1.9, \beta_3 = 1.3$ $\beta_1 = 1.7, \alpha = 0.7$	$\beta_2 = 1.5, \beta_3 = 1.2$ $\beta_1 = 1.7, \alpha = 0.7$
n_t	n_x	n_y	n_z	it	it
256	256	256	256	5	5
512	512	512	512	5	5
1024	1024	1024	1024	6	4
2048	2048	2048	2048	6	5

in Table 5.6. We note that in the first two parameter setups there is a benign increase in the iteration numbers while the dimensionality is increased. We want to point out that for these computations AMEN was used in a standard way. We believe that incorporating within AMEN preconditioners that are tailored for the variable coefficient matrices will allow for a better convergence. This and the use of the even more efficient QTT [26] representation is future

ETNA
Kent State University
http://etna.math.kent.edu

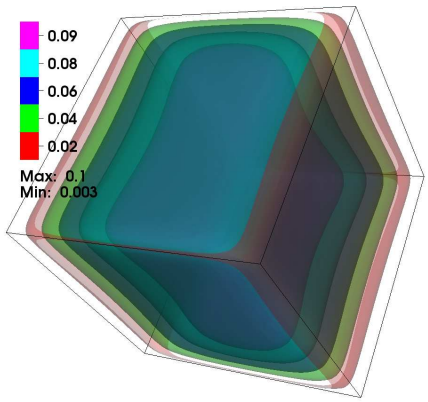


FIG. 5.5. *Problem* (3.9). *Solution at time-step* 12 *for a three-dimensional setup.*

work and beyond the scope of this paper.

TABLE 5.6
Performance of AMEN *as the mesh is refined, for different values of the differentiation orders.*

			$\beta_1 = 1.3, \beta_2 = 1.5$ $\alpha = 0.3$	$\beta_1 = 1.7, \beta_2 = 1.9$ $\alpha = 0.2$	$\beta_1 = 1.9, \beta_2 = 1.1$ $\alpha = 0.7$
n_t	n_y	n_x	it	it	it
256	256	256	8	6	6
512	512	512	10	9	6
1024	1024	1024	16	16	7

6. Conclusions. We have introduced four problems that use the well-established Grünwald-Letnikov scheme (and some of its descendants), as a sample of emerging mathematical models that rely on FDEs. We derived the discrete linear systems and matrix Sylvester-type equations that represent the discretized version of the space-, time- and space-time-fractional derivative for both constant and variable coefficients. For all problems it was crucial to notice the Toeplitz structure of the discrete differentiation operator whenever present. While the simplest model problem only required a circulant preconditioner in combination with the preconditioned CG method, the more involved problems needed further study. For realistic discretization levels it was no longer possible to explicitly store the approximate solution vectors. We thus considered low-rank matrix equations solvers and in particular focused on the successful KPIK method in its inexact version IKPIK. This method was then extremely effective when we again used the circulant preconditioned Krylov solver to evaluate any linear systems

ETNA
Kent State University
http://etna.math.kent.edu

with the inverse of a differentiation matrix. The last and most challenging problem was then solved using recently developed tensor-methodology and, while still future research needs to be devoted to understanding these solvers better, the numerical results are very promising.

Acknowledgements. Martin Stoll would like to thank Anke Stoll for introducing him to fractional calculus. Additionally, he is indebted to Akwum Onwunta and Sergey Dolgov for help with the TT Toolbox. The research of the second author was partially supported by the Italian research fund INdAM-GNCS 2013 "Strategie risolutive per sistemi lineari di tipo KKT con uso di informazioni strutturali".

REFERENCES

[1] G. S. AMMAR AND W. B. GRAGG, *Superfast solution of real positive definite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 61–76.
[2] J. BALLANI AND L. GRASEDYCK, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., 20 (2013), pp. 27–43.
[3] R. H. BARTELS AND G. W. STEWART, *Algorithm 432: Solution of the Matrix Equation $AX + XB = C$*, Comm. ACM, 15 (1972), pp. 820–826.
[4] B. BECKERMANN, *An error analysis for rational Galerkin projection applied to the Sylvester equation*, SIAM J. Numer. Anal., 49 (2011), pp. 2430–2450.
[5] K. BURRAGE, N. HALE, AND D. KAY, *An efficient implicit FEM scheme for fractional-in-space reaction-diffusion equations*, SIAM J. Sci. Comput., 34 (2012), pp. A2145–A2172.
[6] M. CAPUTO, *Elasticità e Dissipazione*, Zanichelli, Bologna, 1969.
[7] M. K. CHEN, *On the solution of circulant linear systems*, SIAM J. Numer. Anal., 24 (1987), pp. 668–683.
[8] J. W. COOLEY AND J. W. TUKEY, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.
[9] S. DOLGOV, *TT-GMRES: solution to a linear system in the structured tensor format*, Russian J. Numer. Anal. Math. Modelling, 28 (2013), pp. 149–172.
[10] ———, *Tensor Product Methods in Numerical Simulation of High-Dimensional Dynamical Problems*, PhD Thesis, MPI Leipzig, 2014.
[11] S. DOLGOV, J. W. PEARSON, D. V. SAVOSTYANOV, AND M. STOLL, *Fast tensor product solvers for optimization problems with fractional differential equations as constraints*, Appl. Math. Comput., 273 (2016), pp. 604–623.
[12] S. V. DOLGOV AND D. V. SAVOSTYANOV, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J. Sci. Comput., 36 (2014), pp. A2248–A2271.
[13] M. DONATELLI, M. MAZZA, AND S. SERRA-CAPIZZANO, *Spectral analysis and structure preserving preconditioners for fractional diffusion equations*, J. Comput. Phys., 307 (2016), pp. 262–279.
[14] V. DRUSKIN AND L. KNIZHNERMAN, *Extended Krylov subspaces: approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 755–771.
[15] J. DURBIN, *The fitting of time-series models*, Rev. Inst. Internat. Stat., 28 (1960), pp. 233–244.
[16] G. J. FIX AND J. P. ROOP, *Least squares finite-element solution of a fractional order two-point boundary value problem*, Comput. Math. Appl., 48 (2004), pp. 1017–1033.
[17] M. FRIGO AND S. G. JOHNSON, *FFTW: An adaptive software architecture for the FFT*, in Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, IEEE Conference Proceedings, Los Alamitos, 1998, pp. 1381–1384.
[18] I. GOHBERG AND I. FELDMAN, *Convolution Equations and Projection Methods for their Solution*, AMS, Providence, 2005.
[19] R. GORENFLO AND F. MAINARDI, *Fractional calculus: integral and differential equations of fractional order*, Preprint on arXiv, arXiv:0805.3823, 2008. http://arxiv.org/abs/0805.3823
[20] L. GRASEDYCK, *Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure*, Computing, 72 (2004), pp. 247–265.
[21] L. GRASEDYCK, D. KRESSNER, AND C. TOBLER, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78.
[22] A. GREENBAUM, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.
[23] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436 (1953).
[24] S. HOLTZ, T. ROHWEDDER, AND R. SCHNEIDER, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713.
[25] I. JESUS, J. MACHADO, AND J. CUNHA, *Fractional electrical impedances in botanical elements*, J. Vib. Control, 14 (2008), pp. 1389–1402.

ETNA
Kent State University
http://etna.math.kent.edu

[26] B. KHOROMSKIJ, *$O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling*, Const. Approx., 34 (2011), pp. 257–280.
[27] L. KNIZHNERMAN AND V. SIMONCINI, *Convergence analysis of the extended Krylov subspace method for the Lyapunov equation*, Numer. Math., 118 (2011), pp. 567–586.
[28] R. C. KOELLER, *Applications of fractional calculus to the theory of viscoelasticity*, Trans. ASME J. Appl. Mech., 51 (1984), pp. 299–307.
[29] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[30] D. KRESSNER AND C. TOBLER, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2009/10), pp. 1688–1714.
[31] P. LANCASTER AND M. TISMENETSKY, *The Theory of Matrices*, 2nd ed., Academic Press, Orlando, 1985.
[32] S.-L. LEI AND H.-W. SUN, *A circulant preconditioner for fractional diffusion equations*, J. Comput. Phys., 242 (2013), pp. 715–725.
[33] N. LEVINSON, *The Wiener RMS (root mean square) error criterion in filter design and prediction*, J. Math. Phys. Mass. Inst. Tech., 25 (1947), pp. 261–278.
[34] F. LIU, V. ANH, AND I. TURNER, *Numerical solution of the space fractional Fokker-Planck equation*, J. Comput. Appl. Math, 166 (2004), pp. 209–219.
[35] M. M. MEERSCHAERT AND C. TADJERAN, *Finite difference approximations for fractional advection-dispersion flow equations*, J. Comput. Appl. Math., 172 (2004), pp. 65–77.
[36] ———, *Finite difference approximations for two-sided space-fractional partial differential equations*, Appl. Numer. Math., 56 (2006), pp. 80–90.
[37] R. METZLER AND J. KLAFTER, *The random walk's guide to anomalous diffusion: a fractional dynamics approach*, Phys. Rep., 339 (2000), p. 77.
[38] T. MORONEY AND Q. YANG, *A banded preconditioner for the two-sided, nonlinear space-fractional diffusion equation*, Comput. Math. Appl., 66 (2013), pp. 659–667.
[39] M. K. NG, *Iterative Methods for Toeplitz Systems*, Oxford University Press, New York, 2004.
[40] I. OSELEDETS, *DMRG approach to fast linear algebra in the TT-format*, Comput. Methods Appl. Math., 11 (2011), pp. 382–393.
[41] ———, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.
[42] I. OSELEDETS, S. DOLGOV, V. KAZEEV, D. SAVOSTYANOV, O. LEBEDEVA, P. ZHLOBICH, T. MACH, AND L. SONG, *TT-Toolbox*. https://github.com/oseledets/TT-Toolbox
[43] I. V. OSELEDETS AND S. V. DOLGOV, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739.
[44] I. V. OSELEDETS AND E. E. TYRTYSHNIKOV, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 31 (2009), pp. 3744–3759.
[45] I. PODLUBNY, *Fractional Differential Equations*, Academic Press, San Diego, 1998.
[46] I. PODLUBNY, A. CHECHKIN, T. SKOVRANEK, Y. CHEN, AND B. M. VINAGRE JARA, *Matrix approach to discrete fractional calculus. II. Partial fractional differential equations*, J. Comput. Phys., 228 (2009), pp. 3137–3153.
[47] I. PODLUBNY, I. PETRÁŠ, B. M. VINAGRE, P. O'LEARY, AND L. DORČÁK, *Analogue realizations of fractional-order controllers*, Nonlinear Dynam., 29 (2002), pp. 281–296.
[48] Y. PU, *Application of fractional differential approach to digital image processing*, J. Sichuan Univ. (Eng. Sci. Ed.), 3 (2007), pp. 124–132.
[49] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[50] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[51] S. SAMKO, A. KILBAS, AND O. MARICHEV, *Fractional Integrals and Derivatives*, Gordon and Breach, Yverdon, 1993.
[52] R. SCHNEIDER AND A. USCHMAJEW, *Approximation rates for the hierarchical tensor format in periodic Sobolev spaces*, J. Complexity, 30 (2014), pp. 56–71.
[53] V. SIMONCINI, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.
[54] ———, *Computational methods for linear matrix equations*, SIAM Rev., to appear, 2016.
[55] V. SIMONCINI AND D. B. SZYLD, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.
[56] M. STOLL AND T. BREITEN, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci. Comput., 37 (2015), pp. B1–B29.
[57] G. STRANG, *A proposal for Toeplitz matrix calculations*, Stud. Appl. Math., 74 (1986), pp. 171–176.
[58] C. TARLEY, G. SILVEIRA, W. DOS SANTOS, G. MATOS, E. DA SILVA, M. BEZERRA, M. MIRÓ, AND S. FERREIRA, *Chemometric tools in electroanalytical chemistry: methods for optimization based on factorial design and response surface methodology*, Microchem. J, 92 (2009), pp. 58–67.

ETNA
Kent State University
http://etna.math.kent.edu

[59] C. TOBLER, *Low-Rank Tensor Methods for Linear Systems and Eigenvalue Problems*, PhD Thesis, Diss. Nr. 20320, ETH Zürich, 2012.
[60] E. E. TYRTYSHNIKOV, *Tensor approximations of matrices generated by asymptotically smooth functions*, Mat. Sb., 194 (2003), pp. 147–160.
[61] H. WANG AND N. DU, *Fast alternating-direction finite difference methods for three-dimensional space-fractional diffusion equations*, J. Comput. Phys., 258 (2014), pp. 305–318.
[62] H. WANG, K. WANG, AND T. SIRCAR, *A direct $O(N \log^2 N)$ finite difference method for fractional diffusion equations*, J. Comput. Phys., 229 (2010), pp. 8095–8104.
[63] C. WEX, A. STOLL, M. FRÖHLICH, S. ARNDT, AND H. LIPPERT, *How preservation time changes the linear viscoelastic properties of porcine liver*, Biorheology, 50 (2013), pp. 115–131.