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 or in the variable coefficient case Toeplitz-like structure . 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 techniqes 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 [17] 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 [32]. Other applications of fractional calculus are viscoelasticity - for example using the Kelvin-Voigt fractional derivative model [23, 53], electrical circuits [21, 40], electro-analytical chemistry [49] or image processing [54].

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 [38, 30] or finite element [14, 29] 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 [27, 33]). 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 [39] where the discretization via finite differences is considered in a purely algebraic framework.

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

---

*Institute for Mathematics and Scientific Computing, Heinrichstr. 36/III, University of Graz, Austria (`tobias.breiten@uni-graz.at`). Most of this work was completed while this author was with the Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

†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`),

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 2.1 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 4.2 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 $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{m \times n}$, $\mathrm{vec}(\mathbf{U}) = [\mathbf{u}_1^T, \ldots, \mathbf{u}_n^T]^T \in \mathbb{R}^{nm}$ 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 $\mathbf{B}$.

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 [38] 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., [39, Formula (3)], [6, Section 2.3])

$$
{}_a^C D_t^\alpha f(t) = \frac{1}{\Gamma(n - \alpha)} \int_a^t \frac{f^{(n)}(s)ds}{(t - s)^{\alpha - n + 1}}, \tag{2.1}
$$

where $\Gamma(x)$ is the gamma function. Following the discussion in [39] the Caputo derivative is frequently used for the derivative with respect to time. We also define the Riemann-Liouville derivative (see, e.g., [39, 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

$$
{}_a^{RL} D_t^\alpha f(t) = \frac{1}{\Gamma(n - \alpha)} \left(\frac{d}{dt} \right)^n \int_a^t \frac{f(s)ds}{(t - s)^{\alpha - n + 1}}, \qquad a < t < b. \tag{2.2}
$$

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

$$
{}_t^{RL} D_b^\alpha f(t) = \frac{(-1)^n}{\Gamma(n - \alpha)} \left(\frac{d}{dt} \right)^n \int_t^b \frac{f(s)ds}{(s - t)^{\alpha - n + 1}}, \qquad a < t < b. \tag{2.3}
$$

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., [39, Formula (5)]), that is,

$$\frac{d^\alpha f(t)}{d\,|t|^\alpha} =_t D_R^\alpha f(t) = \frac{1}{2} \left({}_a^{RL}D_t^\alpha f(t) + {}_t^{RL}D_b^\alpha f(t) \right). \tag{2.4}$$

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 [43, 38].

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 [30] for the introduction of the basic methodology. According to [30] the following holds

$$_a^{RL}D_x^\alpha f(x, t) = \lim_{M\to\infty} \frac{1}{h^\alpha} \sum_{k=0}^{M} g_{\alpha,k} f(x - kh, t), \tag{2.5}$$

for the left-sided derivative (see (3) in [30] or (7.10) in [38]), where $h = \frac{x-a}{M}$, and $g_{\alpha,k}$ is given by

$$g_{\alpha,k} = \frac{\Gamma(k-\alpha)}{\Gamma(-\alpha)\Gamma(k+1)} = (-1)^k \begin{pmatrix} \alpha \\ k \end{pmatrix}. \tag{2.6}$$

For the efficient computation of the coefficients $g_{\alpha,k}$ one can use the following recurrence relation[2] ([38])

$$g_{\alpha,0} = 1, \quad g_{\alpha,k} = \left(1 - \frac{\alpha+1}{k}\right) g_{\alpha,k-1}. \tag{2.7}$$

This finite difference approximation is of first order [30]. One can analogously obtain an expression for the right-sided derivative via

$$_x^{RL}D_b^\alpha f(x, t) = \lim_{M\to\infty} \frac{1}{h^\alpha} \sum_{k=0}^{M} g_{\alpha,k} f(x + kh, t), \tag{2.8}$$

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

Alternatively, a shifted Grünwald-Letnikov formula is introduced in [30]. This discretization shows advantages regarding the stability when the fractional derivative is part of an unsteady differential equation. The basis of the discretized operator used later is the following shifted expression

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

---

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

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

introduced in [30, 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

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

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 (2.4) 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.,

$$
\frac{u^{n+1} - u^n}{\tau} - {}_a^{RL}D_x^\beta u^{n+1} = f^{n+1}.
\tag{3.3}
$$

The stable discretization of this problem is introduced in [30, 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 by

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

We now approximate the symmetric Riesz derivative as the weighted sum of the left- and right-sided Riemann-Liouville fractional derivative (see also [39, 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 [31, 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

$$\left(\mathbf{I}^{n_x} - \tau\mathbf{L}_\beta^{n_x}\right)\mathbf{u}^{n+1} = \mathbf{u}^n + \tau\mathbf{f}^{n+1} \tag{3.4}$$

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 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)

$$\frac{du(x,y,t)}{dt} - {}_xD_R^{\beta_1}u(x,y,t) - {}_yD_R^{\beta_2}u(x,y,t) = f(x,y,t), \quad (x,y,t) \in \Omega \times (0,T],$$
$$u(x,y,t) = 0, \qquad (x,y,t) \in \Gamma \times [0,T], \tag{3.5}$$
$$u(x,y,0) = 0, \qquad (x,y) \in \Omega,$$

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

$$\frac{1}{\tau}\left(\mathbf{u}^{n+1} - \mathbf{u}^n\right) = \underbrace{\left(\mathbf{I}^{n_y}\otimes\mathbf{L}_{\beta_1}^{n_x} + \mathbf{L}_{\beta_2}^{n_y}\otimes\mathbf{I}^{n_x}\right)}_{\mathbf{L}_{\beta_1,\beta_2}^{n_xn_y}}\mathbf{u}^{n+1} + \mathbf{f}^{n+1}. \tag{3.6}$$

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

$$\left(\mathbf{I}^{n_xn_y} - \tau\mathbf{L}_{\beta_1,\beta_2}^{n_xn_y}\right)\mathbf{u}^{n+1} = \mathbf{u}^n + \tau\mathbf{f}^{n+1}. \tag{3.7}$$

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.,

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

this can be rewritten as a Sylvester equation of the form

$$\left(\frac{1}{2}\mathbf{I}^{n_x} - \tau \mathbf{L}_{\beta_1}^{n_x} \right) \mathbf{U}^{n+1} + \mathbf{U}^{n+1} \left(\frac{1}{2}\mathbf{I}^{n_y} - \tau \mathbf{L}_{\beta_2}^{n_y} \right)^T = \mathbf{U}^n + \tau \mathbf{F}^{n+1}. \tag{3.8}$$

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

$$\begin{aligned}
{}_0^C D_t^\alpha u(x,t) - {}_x D_R^\beta u(x,t) &= f(x,t), & (x,t) &\in (0,1) \times (0,T], \\
u(0,t) = u(1,t) &= 0, & t &\in [0,T], \\
u(x,0) &= 0, & x &\in [0,1],
\end{aligned} \tag{3.9}$$

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

$$\mathbf{T}_\alpha^{n_t+1} \begin{bmatrix} u(x,t_0) \\ u(x,t_1) \\ \vdots \\ u(x,t_{n_t}) \end{bmatrix} - \mathbf{I}^{n_t+1} \begin{bmatrix} {}_x D_R^\beta u(x,t_0) \\ {}_x D_R^\beta u(x,t_1) \\ \vdots \\ {}_x D_R^\beta u(x,t_{n_t}) \end{bmatrix} = \mathbf{I}^{n_t+1} \begin{bmatrix} f(x,t_0) \\ f(x,t_1) \\ \vdots \\ f(x,t_{n_t}) \end{bmatrix}, \tag{3.10}$$

where

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

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

$$\mathbf{T}_\alpha^{n_t} \begin{bmatrix} u(x,t_1) \\ u(x,t_2) \\ \vdots \\ u(x,t_{n_t}) \end{bmatrix} - \mathbf{I}^{n_t} \begin{bmatrix} {}_x D_R^\beta u(x,t_1) \\ {}_x D_R^\beta u(x,t_2) \\ \vdots \\ {}_x D_R^\beta u(x,t_{n_t}) \end{bmatrix} = \begin{bmatrix} f(x,t_1) \\ f(x,t_2) \\ \vdots \\ f(x,t_{n_t}) \end{bmatrix} - \begin{bmatrix} g_{\alpha,1} \\ g_{\alpha,2} \\ \vdots \\ g_{\alpha,n_t} \end{bmatrix} u(x,t_0). \tag{3.11}$$

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

$$\left((\mathbf{T}_\alpha^{n_t} \otimes \mathbf{I}^{n_x}) - (\mathbf{I}^{n_t} \otimes \mathbf{L}_\beta^{n_x}) \right) \mathbf{u} = \tilde{\mathbf{f}} \tag{3.12}$$

where $\mathbf{u} = [\mathbf{u}_1^T, \mathbf{u}_2^T, \ldots, \mathbf{u}_{n_t}^T]^T \in \mathbb{R}^{n_x n_t}$. Similarly, we define $\tilde{\mathbf{f}} = [\mathbf{f}_1^T, \mathbf{f}_2^T, \ldots, \mathbf{f}_{n_t}^T]^T$. Introducing the matrices $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n_t}]$ and $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_{n_t}]$, we can rewrite (3.12) as

$$\mathbf{U}(\mathbf{T}_\alpha^{n_t})^T - \mathbf{L}_\beta^{n_x} \mathbf{U} = \mathbf{F}, \tag{3.13}$$

which shows that $\mathbf{U}$ is the solution to a Sylvester matrix equation (see, e.g., [26]), 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

$$\begin{aligned}
{}^C_0 D_t^\alpha u(x,y,t) - {}_x D_R^{\beta_1} u(x,y,t) - {}_y D_R^{\beta_2} u(x,y,t) &= f(x,y,t), & (x,y,t) &\in \Omega \times (0,T], \\
u(x,y,t) &= 0, & (x,y,t) &\in \Gamma \times [0,T], \\
u(x,y,0) &= 0, & (x,y) &\in \Omega,
\end{aligned} \tag{3.14}$$

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

$$\left(\mathbf{T}_\alpha^{n_t} \otimes \mathbf{I}^{n_x n_y} - \mathbf{I}^{n_t} \otimes \mathbf{L}_{\beta_1, \beta_2}^{n_x n_y} \right) \mathbf{u} = \tilde{\mathbf{f}}. \tag{3.15}$$

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 coefficents. In this section we consider a more general FDE, which involves separable variable coefficients. 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 when, i.e., $p_+(x_1, x_2) = p_+^{(1)}(x_1) p_+^{(2)}(x_2)$, and similarly for the other coefficients. The left and right sided Riemann-Liouville derivatives enter the equation with different coefficients, therefore the block

$$\left(p_+ \, {}^{\mathrm{RL}}_0 D_{x_1}^{\beta_1} + p_- \, {}^{\mathrm{RL}}_{x_1} D_1^{\beta_1} + q_+ \, {}^{\mathrm{RL}}_0 D_{x_2}^{\beta_2} + q_- \, {}^{\mathrm{RL}}_{x_2} D_1^{\beta_2} \right) u(x_1, x_2, t)$$

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,

$$\left(\mathbf{P}_{+,2} \otimes \mathbf{P}_{+,1} \mathbf{T}_{\beta_1} + \mathbf{P}_{-,2} \otimes \mathbf{P}_{-,1} \mathbf{T}_{\beta_1}^T + \mathbf{Q}_{+,2} \mathbf{T}_{\beta_2}^T \otimes \mathbf{Q}_{+,1} + \mathbf{Q}_{-,2} \mathbf{T}_{\beta_2} \otimes \mathbf{Q}_{-,1} \right) \mathbf{vec}(\mathbf{U}). \tag{3.16}$$

A simpler case occurs when the coefficients only depend on one spatial variable; for instance, the following data was used in [51]:

$$\left({}^{\mathrm{C}}_0 D_t^\alpha - p_+ \, {}^{\mathrm{RL}}_0 D_{x_1}^{\beta_1} - p_- \, {}^{\mathrm{RL}}_{x_1} D_1^{\beta_1} - q_+ \, {}^{\mathrm{RL}}_0 D_{x_2}^{\beta_2} - q_- \, {}^{\mathrm{RL}}_{x_2} D_1^{\beta_2} \right) u(x_1, x_2, t) = 0$$

with

$$p_+ = \Gamma(1.2)x_1^{\beta_1}, p_- = \Gamma(1.2)(2 - x_1)^{\beta_1}, q_+ = \Gamma(1.2)x_2^{\beta_2}, q_- = \Gamma(1.2)(2 - x_2)^{\beta_2}.$$

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

$$\mathbf{A} = \mathbf{T}_\alpha^{n_t} \otimes \mathbf{I}^{n_2 n_1} - \mathbf{I}^{n_t} \otimes \mathbf{I}^{n_2} \otimes \left(\mathbf{P}_+\mathbf{T}_{\beta_1} + \mathbf{P}_-\mathbf{T}_{\beta_1}^\top\right) - \mathbf{I}^{n_t} \otimes \left(\mathbf{Q}_+\mathbf{T}_{\beta_2} + \mathbf{Q}_-\mathbf{T}_{\beta_2}^\top\right) \otimes \mathbf{I}^{n_1}, \quad (3.17)$$

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

$$\mathbf{A} = \mathbf{T}_\alpha \otimes \mathbf{I}^{n_1 n_2} - \mathbf{I}^{n_t} \otimes \hat{\mathbf{L}}_{\beta_1}^{n_x, P} \otimes \mathbf{I}^{n_2} - \mathbf{I}_{n_t} \otimes \mathbf{I}^{n_1} \otimes \hat{\mathbf{L}}_{\beta_2}^{n_x, Q}. \qquad (3.18)$$

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

$$p_+(x_1, x_2) = \sum_{j=1}^{r} p_{+,j}(x_1) p_{+,j}(x_2),$$

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

$$\sum_{j=1}^{r} \left(\mathbf{P}_{+,2}^{(j)} \otimes \mathbf{P}_{+,1}^{(j)}\mathbf{T}_{\beta_1} + \mathbf{P}_{-,2}^{(j)} \otimes \mathbf{P}_{-,1}^{(j)}\mathbf{T}_{\beta_1}^T + \mathbf{Q}_{+,2}^{(j)}\mathbf{T}_{\beta_2}^T \otimes \mathbf{Q}_{+,1}^{(j)} + \mathbf{Q}_{-,2}^{(j)}\mathbf{T}_{\beta_2} \otimes \mathbf{Q}_{-,1}^{(j)}\right). \qquad (3.19)$$

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. Nonetheless, the fact that it is a sum of terms associated with Toeplitz matrices will be exploited in our solution strategy.

4. Matrix equations solvers. This section is devoted to the introduction of the methodology that allows us to efficiently solve the problems presented in Sections 3. In all cases, we start our discussion by considering the constant coefficient case, and then we continue with by the separable variable coefficient case. We saw that the discretization of all problems leads to a system that contained a special structure within the matrix, the so-called Toeplitz matrices discussed in the next section. We there recall efficient ways to work with Toeplitz matrices and in particular focus on techniques for fast multiplications and introduce approximations to Toeplitz matrices that can be used as preconditioners.

We then proceed by introducing methods that are well-suited for the numerical solution of large scale linear matrix equations, and in particular of Sylvester equations, as they occur in the case when we have an additional time-fractional or two-dimensional problem. This is followed by a brief

discussion of a tensor-based approach for the time-fractional problem with two space dimensions. We shall mainly report on the recently developed KPIK method [44], which seemed to provide a fully satisfactory performance on the problems tested; the code is used as a stand alone solver. We additionally use this method as a preconditioner within a low-rank Krylov subspace solver.

Lastly, we discuss the tensor-structured problem and introduce a suitable solver based on recently developed tensor techniques [36, 35].

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 r_0, the Krylov subspace of size l is defined as

$$\mathcal{K}_l(\mathcal{A}, r_0) = \text{span}\left\{r_0, \mathcal{A}r_0, \ldots, \mathcal{A}^{l-1}r_0\right\}.$$

As l increases, the space dimension grows, and the spaces are nested, namely $\mathcal{K}_l(\mathcal{A}, r_0) \subseteq \mathcal{K}_{l+1}(\mathcal{A}, r_0)$. In the following we shall also consider wide forms of generalizations of this original definition, from the use of matrices in place of r_0, to the use of sequences of shifted and inverted matrices instead of $\mathcal{A}$.

4.1. Computations with Toeplitz matrices. We briefly discuss the properties of Toeplitz matrices and possible solvers. As Ng points out in [34] many direct solution strategies exist for the solution of Toeplitz systems that can efficiently solve these systems recursively. We mention here [28, 13, 16, 1] 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 specific value for n is. Let us consider a basic Toeplitz matrix of the form

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

Circulant matrices, which take the generic form

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

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 $\mathbf{F}$[3]. In more detail, the diagonalization of $\mathbf{C}$ is written as $\mathbf{C} = \mathbf{F}^{\mathbf{H}}\mathbf{\Lambda}\mathbf{F}$ where $\mathbf{F}$ is again the Fourier

---

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

matrix and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues (see [7, 34]). 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 $\mathbf{F}^{\mathbf{H}}$ times a vector can be done via the Fast Fourier Transform (FFT, [8, 15]). 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 mentioned above 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 $\mathbf{P}_{+}$.

For the constant coefficient case, i.e., the Toeplitz case there exists a variety of different preconditioners [34]. Here we focus on a classical circulant preconditioner introduced by Strang in [48]. 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

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

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 [34].

The computational strategy described above can be used to solve the linear system in (3.4) 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 [52] 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) [20], which requires matrix-vector products with the coefficient matrix; for more details on iterative Krylov subspace solvers we refer the reader to [19, 41, 46]. 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 [27].

The numerical treatment is more complicated in the variable coefficient case. The authors of [11] point out that the use of circulant preconditioners for this case is not as effective and suggest the use of the following approximation

$$-\hat{\mathbf{L}}_{\beta_1}^{n_x,P} = -(\mathbf{P}_+ \mathbf{T}_{\beta_1} + \mathbf{P}_- \mathbf{T}_{\beta_1}^\top) \approx \mathbf{P}_+ \mathbf{A} + \mathbf{P}_- \mathbf{A}^\top =: \mathbf{W} \tag{4.1}$$

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

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

as suggested in [11]. We can use a nonsymmetric Krylov solver such as GMRES [42] 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 factorisation.

One of the major concerns is carrying the approaches for the one-dimensional case over to the higher dimensional settings. We now present techniques well established for matrix equations and show how they can be used for the solution of fractional differential equations in more than one dimension or when the time derivative is also of fractional order.

We are in particular focussing on the connection to matrix equation solvers, which we introduce in the next section. In addition we mention the possible use of low-rank versions [47, 25] 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 [25].

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

$$\mathbf{A}\mathbf{U} + \mathbf{U}\mathbf{B} = \mathbf{C} \tag{4.2}$$

arises in a large variety of applications; we refer the reader to [45] for a detailed description. Note that in general, $\mathbf{A}$ and $\mathbf{B}$ are allowed to have different dimensions, so that the right-hand side $\mathbf{C}$ and the unknown solution $\mathbf{U}$ are rectangular matrices. A unique solution is ensured if $\mathbf{A}$ and $-\mathbf{B}$ have disjoint spectra. Robust numerical procedures for solving (4.2) 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 $\mathbf{B}$ is small and already in triangular form, the way the equation (3.13) is in Problem 2, when n_t is small. Let $\mathbf{U} = [u_1, \ldots, u_{n_B}]$, where n_B is the size of $\mathbf{B}$, and $\mathbf{C} = [c_1, \ldots, c_{n_B}]$. Explicit inspection shows that if $\mathbf{B}$ is upper triangular, then the first column of $\mathbf{U}$ can be obtained by solving the shifted system $(\mathbf{A}+\mathbf{B}_{11}\mathbf{I})u_1 = c_1$. All subsequent columns of $\mathbf{U}$ may be obtained with a backward substitution as

$$(\mathbf{A}+\mathbf{B}_{ii}\mathbf{I})u_i = c_i - \sum_{k=1}^{i-1} u_k \mathbf{B}_{ki}, \quad i = 2, ..., n_B.$$

Each of these systems may be solved by CG equipped with a circulant preconditioner, as for Problem 1. This strategy is appealing when n_B, the size of $\mathbf{B}$, 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, $\mathbf{C}$ usually has much lower rank than the problem dimension, which makes low-rank approximation procedures more appealing. Note that the matrix $\mathbf{C}$ in our case reflects the right-hand side of the FDE and as this often has a certain smoothness, we perform a truncation via a truncated singular value decomposition to obtain a low-rank representation $\mathbf{C}$. For example, instead of solving (3.13) we replace the right-hand side $\mathbf{F}$ by a low-rank approximation $\tilde{\mathbf{F}} \approx \mathbf{F}$ with $\tilde{\mathbf{F}} = \tilde{\mathbf{W}}_1 \tilde{\mathbf{W}}_2^T$ and the matrices $\tilde{\mathbf{W}}_1, \tilde{\mathbf{W}}_2$ only have a small number of columns and are computed via a truncated SVD of $\mathbf{F}$. Based on the low-rank nature of the right-hand side of the Sylvester equations efficient methods seek an approximation $\widetilde{\mathbf{U}} \approx \mathbf{U}$ as the product of low-rank matrices, $\widetilde{\mathbf{U}} = \mathbf{V}\mathbf{Y}\mathbf{W}^T$, for some matrix $\mathbf{Y}$, and $\mathbf{V}, \mathbf{W}$ having much fewer columns than rows. These low-rank approximations are of special interest, since in general $\mathbf{U}$ is dense, and thus hard or impossible to store when $\mathbf{A}$ and $\mathbf{B}$ are truly large, as in our cases. Among these approaches are Krylov subspace projection and ADI methods, the latter being a particular Krylov subspace method [45]. We consider the following general projection methods for (4.2): given two approximation spaces Range($\mathbf{V}$) and Range($\mathbf{W}$), an approximation $\widetilde{\mathbf{U}} = \mathbf{V}\widetilde{\mathbf{Y}}\mathbf{W}^T$ is determined by requiring that the residual matrix $\mathbf{R} := \mathbf{A}\widetilde{\mathbf{U}} + \widetilde{\mathbf{U}}\mathbf{B} - \mathbf{C}$ satisfies[4]

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

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

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

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 $\widehat{\mathbf{Y}}$. Different choices of Range($\mathbf{V}$) and Range($\mathbf{W}$) lead to different

---

[4]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($\mathbf{W} \otimes \mathbf{V}$) [45].

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: assuming $\mathbf{C}$ can be written as $\mathbf{C} = \mathbf{C}_1 \mathbf{C}_2^T$, Rational Krylov subspaces generate the two spaces $\text{Range}(\mathbf{V}) = \text{Range}([\mathbf{C}_1, (\mathbf{A} - \sigma_1 \mathbf{I})^{-1} \mathbf{C}_1, (\mathbf{A} - \sigma_2 \mathbf{I})^{-1} \mathbf{C}_1, \ldots])$ and $\text{Range}(\mathbf{W}) = \text{Range}([\mathbf{C}_2, (\mathbf{B}^T - \eta_1 \mathbf{I})^{-1} \mathbf{C}_2, (\mathbf{B}^T - \eta_2 \mathbf{I})^{-1} \mathbf{C}_2, \ldots])$, 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 $\text{Range}(\mathbf{V})$ 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* [12]. In [44] it was shown for $\mathbf{B} = \mathbf{A}^T$ that such a space can be generated progressively as

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

and expanded until the approximate solution $\widetilde{\mathbf{U}}$ is sufficiently good. Note that in a standard implementation that sequentially generates $\mathbb{EK}(\mathbf{A}, \mathbf{C}_1)$, 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 $\mathbf{C}_1$, although as the iteration proceeds this number could decrease, in case rank deficiency occurs. The implementation of the resulting projection method with $\mathbb{EK}(\mathbf{A}, \mathbf{C}_1)$ as approximation space was called KPIK in [44] for the Lyapunov matrix equation, that is (4.2) with $\mathbf{B} = \mathbf{A}^T$ and $\mathbf{C}_1 = \mathbf{C}_2$.

The procedure in [44] can be easily adapted to the case of general $\mathbf{B}$, so that also the space $\mathbb{EK}(\mathbf{B}^T, \mathbf{C}_2)$ is constructed [45]. For consistency we shall also call KPIK our implementation for the Sylvester equation.

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 time. The two devices are tightly related. Indeed, the generation of $\mathbb{EK}(\mathbf{A}, \mathbf{C}_1)$ requires solving systems with $\mathbf{A}$, whose cost is problem dependent. Therefore, the larger the space, the higher this cost is, increasing the overall computational time. The space dimension determines the rate of convergence of the approximate solution $\widetilde{\mathbf{U}}$ towards $\mathbf{U}$; how the accuracy improves as the space dimension increases was recently analyzed in [22] 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.

From a computational standpoint, our application problem is particularly demanding because the iterative generation of the extended space requires solving systems with $\mathbf{A}$ and $\mathbf{B}$, whose size can be very large; see the numerical experiments. 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. Note that 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 the previously mentioned circulant preconditioned CG method. In the case of the variable coefficient we can use the preconditioned GMRES approach that uses the classical finite difference matrix for the preconditioner.

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 $\mathbf{C}_1 = \mathbf{V}\boldsymbol{\gamma}_1$ and $\mathbf{C}_2 = \mathbf{W}\boldsymbol{\gamma}_2$, we have

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

where the two skinny QR factorizations $[\mathbf{AV}, \mathbf{V}] = \mathbf{Q}_1\boldsymbol{\rho}_1$ and $[\mathbf{AW}, \mathbf{W}] = \mathbf{Q}_2\boldsymbol{\rho}_2$ can be updated as the space expands[5]. 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.

4.3. 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 [27], for $1 < \beta < 2$, we recall the following useful properties of $g_{\beta,k}$:

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

We can now adapt the proof in [27] 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 $\sigma(\mathbf{L}_\beta) \subset (-2h^{-\beta}\beta, 0)$. □

---

[5]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 [45]

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 Problem 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 in Problem 2 and Problem 3 as a solver. Assume that $\mathbf{C} = c_1 c_2^T$. In [4] for $\mathbf{A}$ and $\mathbf{B}$ symmetric, it was shown that the residual Frobenius norm is bounded as

$$\frac{\|\mathbf{R}\|_F}{\|c_1\|\,\|c_2\|} \leq 4 \max\{\gamma_{m_\mathbf{A},\mathbf{A},\mathbf{B}}, \gamma_{m_\mathbf{B},\mathbf{B},\mathbf{A}}\} + \xi(\gamma_{m_\mathbf{A},\mathbf{A},\mathbf{B}} + \gamma_{m_\mathbf{B},\mathbf{B},\mathbf{A}}),$$

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 our Problem 3 is obtained for instance whenever $\beta_1 = \beta_2$, it holds (see [22])

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

where κ is the condition number of $\mathbf{A}$. In the notation of Problem 3, 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 (3.13), that is

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

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 $\nu_{i,j} = \lambda_i(\mathbf{A}) + \mu_j(\mathbf{B})$.

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 [50, 2, 35]. General introductions to this very active field of research can be found in [18, 24] and in the literature mentioned there.

Although a variety of solvers for tensor valued equations exists we do not aim at providing a survey of the available methods, but we rather focus on the DMRG (Density Matrix Renormalization Group) method as presented in [35], which is part of the the tensor train (TT) toolbox [36, 35, 37]. The guiding principle is that in the equation $\boldsymbol{\mathcal{A}}x = f$, the tensor-structured matrix $\boldsymbol{\mathcal{A}}$ and the right-hand side are approximated by a simpler (low tensor rank) matrix $\boldsymbol{\mathcal{A}}_{TT}$ and a vector $\tilde{\mathbf{f}}_{TT}$, respectively. The linear system problem is then formulated as a minimization problem, i.e.,

$$\min \left\|\tilde{\mathbf{f}}_{TT} - \boldsymbol{\mathcal{A}}_{TT}x_{TT}\right\|. \tag{4.3}$$

Alternatively, the iterative solver AMEN[10] has shown great potential and is a suitable candidate to solve our fractional problem as well. In our context, the tensor $\boldsymbol{\mathcal{A}}$ is given by

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

$$= (\mathbf{T}_\alpha^{n_t} \otimes \mathbf{I}^{n_x} \otimes \mathbf{I}^{n_y}) - \left(\mathbf{I}^{n_t} \otimes \frac{1}{h^{\beta_1}} \mathbf{I}^{n_y} \otimes \mathbf{L}_{\beta_1}^{n_x} \right) - \left(\mathbf{I}^{n_t} \otimes \mathbf{L}_{\beta_2}^{n_y} \otimes \frac{1}{h^{\beta_2}} \mathbf{I}^{n_x} \right).$$

One of the most used decompositions is the canonical decomposition of a tensor $\boldsymbol{\mathcal{A}}$ ([24]) given by

$$\boldsymbol{\mathcal{A}}(i_1, i_2, i_3) = \sum_{\alpha=1}^{r} \mathbf{U}_1(i_1, \alpha) \mathbf{U}_2(i_2, \alpha) \mathbf{U}_3(i_3, \alpha)$$

with $\boldsymbol{\mathcal{A}}(i_1, i_2, i_3)$ the entry (i_1, i_2, i_3) of the tensor $\boldsymbol{\mathcal{A}}$, canonical factors $\mathbf{U}_i$, $i = 1, 2, 3$ and canonical rank r. While this is already a significant reduction compared to the original tensor, employing this decomposition can be numerically inefficient. Instead, we use the recently introduced tensor train (TT) format: $\boldsymbol{\mathcal{A}}$ is approximated by

$$\boldsymbol{\mathcal{A}} \approx \boldsymbol{\mathcal{A}}_{TT}(i_1, i_2, i_3) = \mathbf{G}_1(i_1) \mathbf{G}_2(i_2) \mathbf{G}_3(i_3),$$

where $\mathbf{G}_k(i_k)$ is an $r_{k-1} \times r_k$ matrix for each fixed i_k. Note that we find a tensor $\boldsymbol{\mathcal{A}}_{TT}$ of low tensor-rank that approximates the full TT decomposition of the tensor $\boldsymbol{\mathcal{A}}$.

In order to solve this tensor valued equation we now employ the DMRG algorithm with the TT decomposition as introduced by Oseledets in [35]. Before briefly describing the method we note that these techniques have been developed before in quantum systems simulation using different terminology; there the TT format is known as the matrix product state (MPS) (see [18] for references). As pointed out earlier the DMRG method transforms the problem of solving the linear system into a least squares problem (4.3), where all full tensors are replaced by their TT approximations of lower tensor rank, i.e., $\tilde{\mathbf{f}}_{TT}$ is the (TT) tensor approximation of $\tilde{\mathbf{f}}$. The DMRG method can be understood as a modified alternating least squares method. While a standard alternating least squares algorithm fixes all but one core and then optimizes the functional the TT DMRG method fixes all but two cores $\mathbf{G}_k, \mathbf{G}_{k+1}$ to compute a solution. The algorithm then moves to the next pair of cores and so on. For more details we refer to Oseledets [35].

Additionally, one can use a TT version of well-known Krylov methods such as GMRES ([42]) adapted to the TT framework (see [9]) but we have not employed this method here.

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.

As this work discusses various different formulations and several solution strategies we want to summarize the suggested solution methods in Table 5.1.
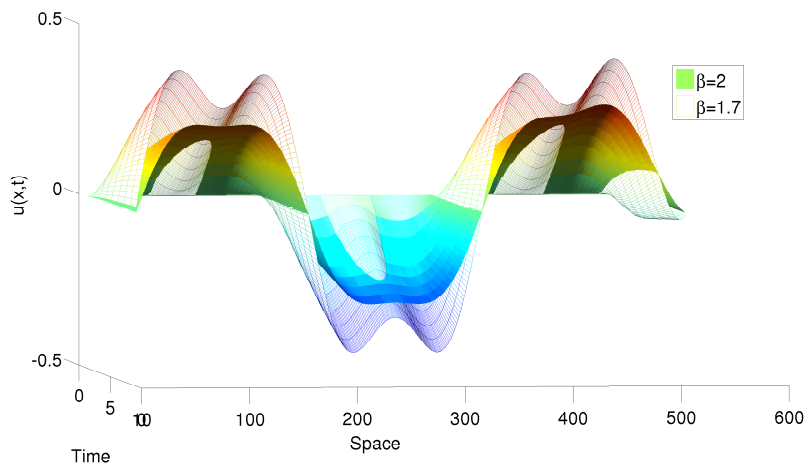
Fig. 5.1: Comparison of numerical solutions when using two different values for β with zero initial condition and zero Dirichlet boundary condition. (510 and 10 space and time points, resp.)

Problem setup	Solution method
Time+1D FDE (3.4)	PCG
Time+2D FDE (3.8)	IKPIK
Frac.Time+1D FDE (3.12)	IKPIK
Frac.Time+2D FDE (3.15)	DMRG

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

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

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

It can be seen that the Strang preconditioner [48] performs exceptionally well with only 6 to 9 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 5.3 that the iteration numbers are very robust using the approximation by the difference matrices, and that timings remain very moderate.

DoF	$\beta = 1.3$ avg. its (time)	$\beta = 1.7$ avg. its (time)
32768	6.0(1.52)	7.0(1.83)
65536	6.0(2.27)	7.0(2.37)
131072	6.0(4.05)	7.6(5.40)
262144	6.0(14.91)	8.4(20.44)
524288	6.0(36.31)	8.9(50.19)
1048576	6.0(63.69)	9.0(91.49)

Table 5.2: Constant coefficient case: Average PCG iteration numbers for 10 time steps and two values of β, as the mesh is refined; in parenthesis is the total CPU time (in secs).

DoF	$\beta = 1.3$ avg. its (time)	$\beta = 1.7$ avg. its (time)
16384	7.0(0.48)	6.3(0.38)
32768	6.8(0.82)	6.0(0.82)
65536	6.4(1.52)	6.0(1.21)
131072	5.9(2.97)	5.9(2.81)
262144	5.4(5.77)	6.0(6.62)
524288	5.1(12.14)	6.0(13.42)
1048576	4.8(40.03)	6.3(48.15)

Table 5.3: Variable coefficient case: Average GMRES 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).

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 5.4 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 digits summarized in Table 5.4 give the average number of IKPIK iterations for 8 time-steps. The variable coefficient is given by

$$p_+ = \Gamma(1.2)x_1^{\beta_1}, p_- = \Gamma(1.2)(2 - x_1)^{\beta_1}, q_+ = \Gamma(1.2)x_2^{\beta_2}, q_- = \Gamma(1.2)(2 - x_2)^{\beta_2}.$$

Note that the largest problem given in Table 5.4 corresponds to $33,554,432$ unknowns. A further increase in the matrix dimensions would require storing the vector of unknowns in a low-rank format.
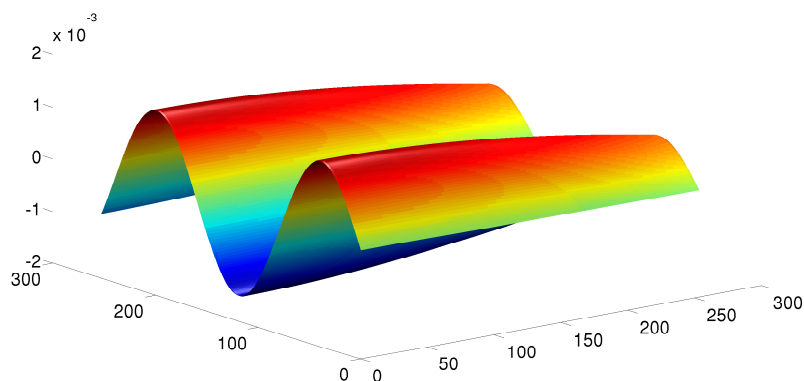
(a) First time-step
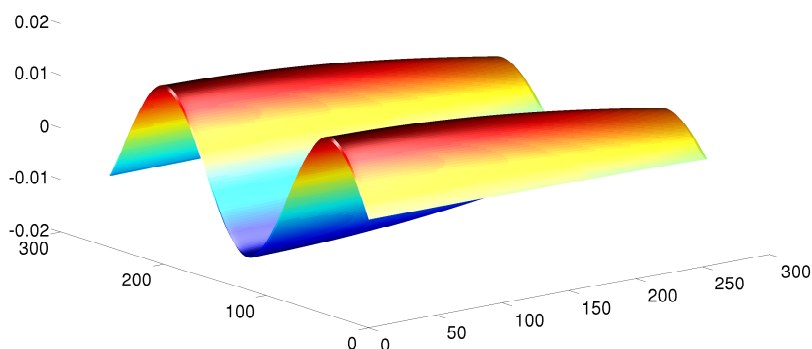
(b) Tenth time-step

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

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

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 5.5 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. 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

n_x	n_y	Variable Coeff.		Constant Coeff.	
		$\beta_1 = 1.3$ $\beta_2 = 1.7$ 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*)

Table 5.4: 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.

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 5.5 this leads to roughly 70 million unknowns. We see that in the constant coefficient case the method performs very

n_t	n_x	Variable Coeff.		
		$\beta = 1.7$ $\alpha = 0.5$ 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*)
n_t	n_x	Constant Coeff.		
		$\beta = 1.7$ $\alpha = 0.5$ 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*)

Table 5.5: Preconditioned low-rank BICG and 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).

robustly 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

$$-\hat{\mathbf{L}}_{\beta_1}^{n_x,P} \approx \mathbf{P}_+ \mathbf{A} + \mathbf{P}_- \mathbf{A}^\top \tag{5.1}$$

is less effective in this setup.

In Figure 5.3 we illustrate the performance of both the preconditioner and the ɪKPIK method for a variety of differentiation parameters in the variable coefficient setup. It can be seen that ɪKPIK 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 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 [36, 35, 37]. The forcing term is given by $f = 100 \sin(10x) \cos(y)$, while zero-Dirichlet conditions are imposed, together with a zero initial condition. The approximations of the right-hand side $\tilde{\mathbf{f}}_{TT}$ and the tensor $\mathbf{A}_{TT}$ use the `round` function within the TT-toolbox. The tolerance for this was set to 10^{-6}. The ᴅᴍʀɢ method recalled in Section 4.4 ([35]) was used, with a convergence tolerance set to 10^{-6}. Figure 5.4 illustrates the different behaviour of the solution for two different values of the temporal differentiation order.

We next illustrate that the performance of ᴅᴍʀɢ 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 5.6 shows the ᴅᴍʀɢ iteration numbers for 4 different mesh sizes both in time and space and three different choices of differentiation orders. The iteration numbers are constant with the mesh size, while they show a very slight increase from 6 or 7 to 11 with the different discretization order. It is also possible to include a preconditioner within ᴅᴍʀɢ but we

n_t	n_y	n_x	$\beta_1 = 1.3, \beta_2 = 1.5$ $\alpha = 0.3$ it	$\beta_1 = 1.7, \beta_2 = 1.9$ $\alpha = 0.7$ it	$\beta_1 = 1.9, \beta_2 = 1.1$ $\alpha = 0.2$ it
512	256	512	7	8	8
256	512	512	7	8	8
1024	512	2048	7	8	9
1024	1024	4096	9	9	9

Table 5.6: Performance of ᴅᴍʀɢ as the mesh is refined, for different values of the differentiation orders.

have not done so here. 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

$$p_+ = \Gamma(1.2)x_1^{\beta_1}x_2, p_- = \Gamma(1.2)(2-x_1)^{\beta_1}x_2, q_+ = \Gamma(1.2)x_2^{\beta_2}x_1, q_- = \Gamma(1.2)(2-x_2)^{\beta_2}x_1$$

and show the results in Table 5.7. Here the newly developed ᴀᴍᴇɴ solver [10] showed slightly better performance than ᴅᴍʀɢ but also was not necessarily tailored for our problem, i.e. no preconditioning was incorporated.
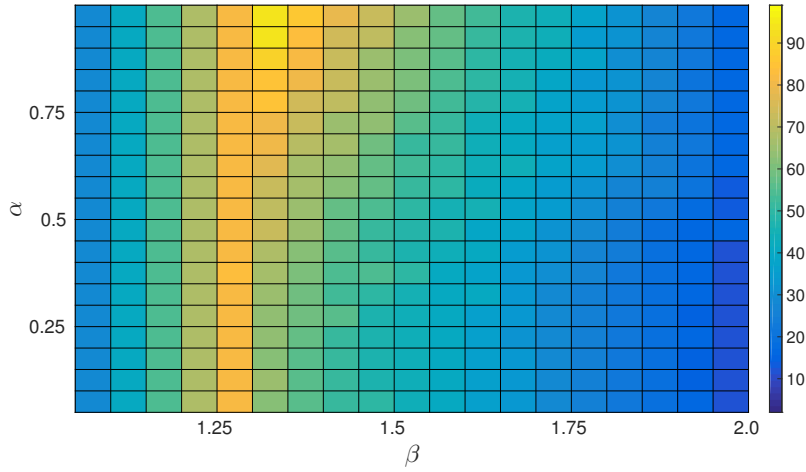
(a) GMRES iterations
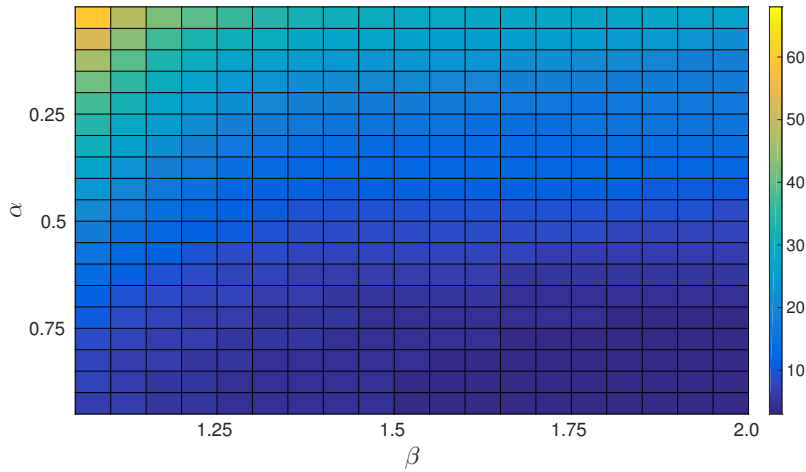
(b) KPIK iterations

Fig. 5.3: 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 $\beta = 1.3$.

6. Conclusions. We pointed out in the introduction that FDE are a crucial tool in mathematical modelling in the coming years. We have introduced four model problems that use the well-established Grünwald-Letnikov scheme (and some of its descendants). These model problems were chosen such that their main numerical features are to be found in many FDE problems. In particular, we derived the discrete linear systems and matrix equations of Sylvester-type that represent

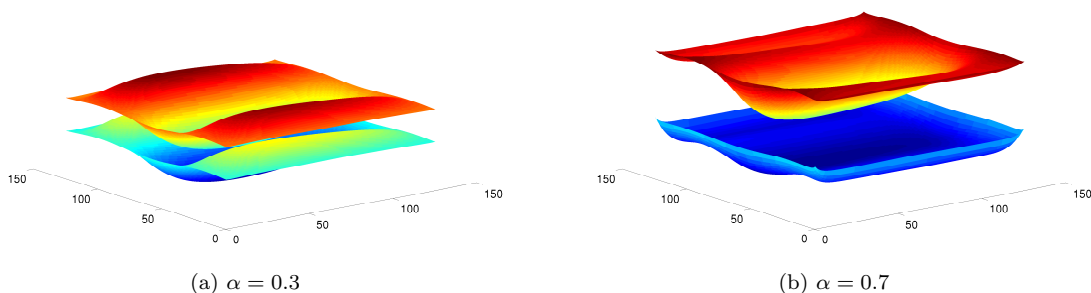(a) $\alpha = 0.3$ (b) $\alpha = 0.7$

Fig. 5.4: Problem 4. Solution at time-step 2 (below) and time-step 20 (above), for two-different orders of the differentiation for the derivative in time.

n_t	n_y	n_x	$\beta_1 = 1.3, \beta_2 = 1.5$ $\alpha = 0.3$ it	$\beta_1 = 1.7, \beta_2 = 1.9$ $\alpha = 0.2$ it	$\beta_1 = 1.9, \beta_2 = 1.1$ $\alpha = 0.7$ it
512	256	512	14	14	15
512	512	512	14	13	14
1024	512	1024	16	18	15
1024	1024	1024	16	20	14

Table 5.7: Performance of AMEN as the mesh is refined, for different values of the differentiation orders.

the discretized version of the space-, time- and space-time-fractional derivative for both constant and variable coefficents. For all problems it was crucial to notice the Toeplitz or Toeplitz-like structure of the discrete differentiation operator. 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 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.

REFERENCES

[1] G. Ammar and W. Gragg, *Superfast solution of real positive definite Toeplitz systems*, SIAM Journal on Matrix Analysis and Applications, 9 (1988), pp. 61–76.

[2] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numerical Linear Algebra with Applications, 20 (2013), pp. 27–43.

[3] R. H. Bartels and G. W. Stewart, *Algorithm 432: Solution of the Matrix Equation $AX + XB = C$*, Comm. of the 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 Journal on Scientific Computing, 34 (2012), pp. A2145–A2172.

[6] M. Caputo, *Elasticità e dissipazione*, Zanichelli Bologna, 1969.

[7] M. Chen, *On the solution of circulant linear systems.*, SIAM J. Numer. Anal., 24 (1987), pp. 668–683.

[8] J. Cooley and J. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of computation, 19 (1965), pp. 297–301.

[9] S. Dolgov, *TT-GMRES: solution to a linear system in the structured tensor format*, Russian Journal of Numerical Analysis and Mathematical Modelling, 28 (2013), pp. 149–172.

[10] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions. part ii: Faster algorithm and application to nonsymmetric systems*, arXiv preprint arXiv:1304.1222, (2013).

[11] M. Donatelli, M. Mazza, and S. Serra-Capizzano, *Spectral analysis and structure preserving preconditioners for fractional diffusion equations*, tech. rep., Technical Report, 2015.

[12] 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.

[13] J. Durbin, *The fitting of time-series models*, Revue de l'Institut International de Statistique, (1960), pp. 233–244.

[14] G. Fix and J. Roof, *Least squares finite-element solution of a fractional order two-point boundary value problem*, Computers & Mathematics with Applications, 48 (2004), pp. 1017–1033.

[15] M. Frigo and S. G. Johnson, *FFTW: An adaptive software architecture for the FFT*, in Proc. 1998 IEEE Intl. Conf. Acoustics Speech and Signal Processing, vol. 3, IEEE, 1998, pp. 1381–1384.

[16] I. Gohberg and I. Feldman, *Convolution equations and projection methods for their solution*, vol. 41, AMS Bookstore, 2005.

[17] R. Gorenflo and F. Mainardi, *Fractional calculus: integral and differential equations of fractional order*, tech. rep., 2008.

[18] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78.

[19] A. Greenbaum, *Iterative methods for solving linear systems*, vol. 17 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[20] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand, 49 (1952), pp. 409–436 (1953).

[21] I. Jesus, J. Machado, and J. Cunha, *Fractional electrical impedances in botanical elements*, Journal of Vibration and Control, 14 (2008), pp. 1389–1402.

[22] L. Knizhnerman and V. Simoncini, *Convergence analysis of the Extended Krylov Subspace Method for the Lyapunov equation*, Numerische Mathematik, 118 (2011), pp. 567–586.

[23] R. Koeller, *Applications of fractional calculus to the theory of viscoelasticity*, ASME, Transactions, Journal of Applied Mechanics(ISSN 0021-8936), 51 (1984), pp. 299–307.

[24] T. Kolda and B. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[25] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl, 31 (2010), pp. 1688–1714.

[26] P. Lancaster and M. Tismenetsky, *Theory of matrices*, vol. 2, Academic press New York, 1969.

[27] S.-L. Lei and H.-W. Sun, *A circulant preconditioner for fractional diffusion equations*, Journal of Computational Physics, 242 (2013), pp. 713–725.

[28] N. Levinson, *The Wiener RMS (root mean square) error criterion in filter design and prediction*, J. Math. and Phys., 25 (1946), pp. 261–278.

[29] F. Liu, V. Anh, and I. Turner, *Numerical solution of the space fractional Fokker–Planck equation*, Journal of Computational and Applied Mathematics, 166 (2004), pp. 209–219.

[30] M. Meerschaert and C. Tadjeran, *Finite difference approximations for fractional advection–dispersion flow*

equations, Journal of Computational and Applied Mathematics, 172 (2004), pp. 65–77.

[31] M. M. Meerschaert and C. Tadjeran, *Finite difference approximations for two-sided space-fractional partial differential equations*, Applied Numerical Mathematics, 56 (2006), pp. 80–90.

[32] R. Metzler and J. Klafter, *The random walk's guide to anomalous diffusion: a fractional dynamics approach*, Physics reports, 339 (2000), pp. 1–77.

[33] T. Moroney and Q. Yang, *A banded preconditioner for the two-sided, nonlinear space-fractional diffusion equation*, Computers & Mathematics with Applications, 66 (2013), pp. 659–667.

[34] M. Ng, *Iterative methods for Toeplitz systems*, Oxford University Press, 2004.

[35] I. Oseledets, *DMRG approach to fast linear algebra in the TT-format*, Comput. Methods Appl. Math., 11 (2011), pp. 382–393.

[36] ———, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317.

[37] I. Oseledets and E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM Journal on Scientific Computing, 31 (2009), pp. 3744–3759.

[38] I. Podlubny, *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*, vol. 198, Access Online via Elsevier, 1998.

[39] I. Podlubny, A. Chechkin, T. Skovranek, Y. Chen, and B. Vinagre Jara, *Matrix approach to discrete fractional calculus II: Partial fractional differential equations*, Journal of Computational Physics, 228 (2009), pp. 3137–3153.

[40] I. Podlubny, I. Petraš, B. Vinagre, P. O'leary, and L. Dorčák, *Analogue realizations of fractional-order controllers*, Nonlinear dynamics, 29 (2002), pp. 281–296.

[41] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.

[42] Y. Saad and M. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput, 7 (1986), pp. 856–869.

[43] S. Samko, A. Kilbas, and O. Marichev, *Fractional integrals and derivatives*, Gordon and Breach Science Publishers Yverdon, 1993.

[44] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.

[45] V. Simoncini, *Computational methods for linear matrix equations*, tech. rep., Università di Bologna, March 2013.

[46] V. Simoncini and D. Szyld, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl, 14 (2007), pp. 1–61.

[47] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM Journal on Scientific Computing, 37 (2015), pp. B1–B29.

[48] G. Strang, *A proposal for Toeplitz matrix calculations*, Studies in Applied Mathematics, 74 (1986), pp. 171–176.

[49] 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*, Microchemical journal, 92 (2009), pp. 58–67.

[50] C. Tobler, *Low-rank tensor methods for linear systems and eigenvalue problems*, PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20320, 2012, 2012.

[51] H. Wang and N. Du, *Fast alternating-direction finite difference methods for three-dimensional space-fractional diffusion equations*, Journal of Computational Physics, 258 (2014), pp. 305–318.

[52] H. Wang, K. Wang, and T. Sircar, *A direct $O(N log^2 N)$ finite difference method for fractional diffusion equations*, Journal of Computational Physics, 229 (2010), pp. 8095 – 8104.

[53] 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.

[54] P. Yi-Fei, *Application of fractional differential approach to digital image processing*, Journal of Sichuan University (Engineering Science Edition), 3 (2007), p. 022.