Tobias Breiten Valeria Simoncini Martin Stoll

Fast iterative solvers for fractional differential equations

MPIMD/14-02 January 31, 2014

Abstract

Fractional differential equations play an important role in science and technology. Many problems can be cast using both fractional time and spatial derivatives. In order to accurately simulate natural phenomena using this technology one needs fine spatial and temporal discretizations. This leads 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. We combine the fast evaluation of Toeplitz matrices and their circulant preconditioners with state-of-the-art linear matrix equation solvers to efficiently solve these problems, both in terms of CPU time and memory requirements. Numerical experiments on typical differential problems with fractional derivatives in both space and time showing the effectiveness of the approaches are reported.

Contents

Tobias Breiten
Institute of Mathematics and Scientific Computing, University of Graz, Heinrichstr. 36, A-8010 Graz, Austria, (tobias.breiten@uni-graz.at)

Valeria Simoncini
Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, 40127 Bologna, Italy (valeria.simoncini@unibo.it)

Martin Stoll
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)

Impressum:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for
Dynamics of Complex Technical Systems

Address:
Max Planck Institute for
Dynamics of Complex Technical Systems
Sandtorstr. 1
39106 Magdeburg

http://www.mpi-magdeburg.mpg.de/preprints/

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 [1] 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 [2]. Other applications of fractional calculus are viscoelasticity - for example using the Kelvin-Voigt fractional derivative model [3, 4], electrical circuits [5, 6], electro-analytical chemistry [7] or image processing [8].

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 the solution of these types of equations. 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 [9, 10] or finite element [11, 12] methods. In this paper we focus on the solution of the discretized equations when a finite difference approximation is used. 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 [13, 14]). Another approach that has recently been studied by Burrage *et al.* is to consider a matrix function approach to solve the discretized system (see [15] for details). Our work here is motivated by some recent results in [16] where the discretization via finite differences is considered in a purely algebraic framework.

Our work differs from previous results in the sense that our aim is to derive fast and reliable solvers for a variety of setups that are sequentially built upon each other. We therefore structure the paper as follows. Section 2.1 is devoted to first establish both Caputo and Riemann-Liouville definitions of a fractional derivative. In Section 3 we give four different problems all of fractional order. The first problem introduced reveals the basic matrix structure that is obtained when tailored finite difference methods are applied. The second problem given in Section 3.2 includes a fractional derivative in both space and time. The resulting matrix structure is then not a simple structured linear system but rather a (linear) matrix Sylvester equation. An even further structured equation is obtained when we next consider a two-dimensional (in space) setup. The fourth problem 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 introduces 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 4.5. 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 $\mathbf{U}$, $\mathrm{vec}(\mathbf{U})$ 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 [9] are both used here and we use this section to briefly recall their definition.

Consider a function $f(t)$ defined on an interval $[a,b]$. Assuming that $f(t)$ is sufficiently often continuously differentiable [9] the Caputo derivative of real order α with $(n-1 \leq \alpha < n)$ is defined as

$$ {}_{a}^{C}D_{t}^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_{a}^{t} \frac{f^{(n)}(s)ds}{(t-s)^{\alpha-n+1}}. \tag{1}$$

Based on the discussion in [16] the Caputo derivative is frequently used for the derivative with respect to time. We also define the Riemann-Liouville derivative: assuming that $f(t)$ is integrable for $t > a$ a left-sided fractional derivative of real order α with $(n-1 \leq \alpha < n)$ is defined as

$$ {}_{a}^{RL}D_{t}^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \left(\frac{d}{dt}\right)^{n} \int_{a}^{t} \frac{f(s)ds}{(t-s)^{\alpha-n+1}}, \qquad a < t < b. \tag{2}$$

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

$$ {}_{t}^{RL}D_{b}^{\alpha} f(t) = \frac{(-1)^{n}}{\Gamma(n-\alpha)} \left(\frac{d}{dt}\right)^{n} \int_{t}^{b} \frac{f(s)ds}{(s-t)^{\alpha-n+1}}, \qquad a < t < b. \tag{3}$$

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-side Riemann-Liouville derivatives, that is,

$$ \frac{d^{\alpha} f(t)}{d\,|t|^{\alpha}} = {}_{t}D_{R}^{\alpha} f(t) = \frac{1}{2}\left({}_{a}^{RL}D_{t}^{\alpha} f(t) + {}_{t}^{RL}D_{b}^{\alpha} f(t)\right). \tag{4}$$

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.

2.2 Numerical approximation

The discretization in the fractional derivatives introduced in Section 2.1 is based on a classical concept in the discretization of FDEs using the formulae by Grünwald-Letnikow [17, 9] defined as

$$ {}_{a}^{RL}D_{t}^{\alpha} f(t) = \lim_{M \to \infty} \frac{1}{h^{\alpha}} \sum_{k=0}^{M} g_{\alpha,k} f(x-kh) \tag{5}$$

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

for the right-side derivative and

$$
{}_{t}^{RL}D_{b}^{\alpha} f(t) = \lim_{M\to\infty} \frac{1}{h^{\alpha}} \sum_{k=0}^{M} g_{\alpha,k} f(x+kh) \tag{6}
$$

for the left-side derivative, which are of first order [10]. Note that $h = \frac{1}{M}$ so that M becomes larger as the mesh-parameter h is refined. The coefficients $g_{\alpha,k}$ are defined as

$$
g_{\alpha,k} = \frac{\Gamma(k-\alpha)}{\Gamma(-\alpha)\Gamma(k+1)} = (-1)^k \left(\begin{array}{c} \alpha \\ k \end{array} \right). \tag{7}
$$

For the efficient computation of the coefficients $g_{\alpha,k}$ one can use the following recurrence relation[2] [9]

$$
g_{\alpha,0} = 1, \quad g_{\alpha,k} = \left(1 - \frac{\alpha+1}{k}\right) g_{\alpha,k-1}. \tag{8}
$$

3 Some model problems and discretizations

In this section we want to introduce four model problems and their corresponding discretizations. We emphasize that these are neither the only relevant formulations nor the only possible ways to discretize each problem. Instead, the purpose of this presentation is to show that rather classical well-studied discretization techniques lead to interesting model problem similarities that can be exploited during the numerical solution of the resulting matrix equations.

3.1 Problem 1

We start with the simple fractional diffusion equation

$$
\frac{du(x,t)}{dt} - {}_{x}D_{R}^{\beta} u(x,t) = f(x,t) \tag{9}
$$

discretized in time by an implicit Euler scheme to give

$$
\frac{u_i^{n+1} - u_i^n}{\tau} = \frac{1}{h_x^{\beta}} \sum_{k=0}^{i+1} g_{\beta,k} u_{i-k+1}^{n+1} + f_i \tag{10}
$$

using the abbreviation $u_i^{n+1} := u(x_i, t_{n+1})$, where n is the index in time.

For the problem (9) the implicit Euler and Crank-Nicholson discretizations are unconditionally stable when a shifted Grünwald-Letnikov formula is used

$$
{}_{x}^{RL}D_{b}^{\beta} u(x,t) = \frac{1}{h_x^{\beta}} \sum_{k=0}^{n_x} g_{\beta,k} u(x-(k-1)h_x, t) + \mathcal{O}(h_x);
$$

one can obtain an approximation for the fractional derivative as

$$
{}_{x}^{RL}D_{b}^{\beta} u(x,t) = \frac{1}{h_x^{\beta}} \sum_{k=0}^{n_x} g_{\beta,k} u(x-(k-1)h_x, t)
$$

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

where n_x denotes the number of grid points in space (see [10] for details).

From now on we assume that a shifted version of the Grünwald-Letnikov formula is used whenever we approximate the Riemann-Liouville derivatives. It is clear that all our techniques apply to the unshifted case as well. Equipped with these tools we can write the right-hand side of (10) as

$$h_x^{-\beta} \begin{bmatrix} g_{\beta,i+1} & g_{\beta,i} & \cdots & \cdots & g_{\beta,1} & g_{\beta,0} \end{bmatrix} \begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_i^{n+1} \\ u_{i+1}^{n+1} \end{bmatrix}$$

and collecting this for all components i into one single matrix system we get

$$\underbrace{h_x^{-\beta} \begin{bmatrix} g_{\beta,1} & g_{\beta,0} & & & & & 0 \\ g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & & & & \\ g_{\beta,3} & g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & & & \\ \vdots & \ddots & g_{\beta,2} & g_{\beta,1} & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & g_{\beta,0} & 0 \\ \vdots & \ddots & \ddots & \ddots & g_{\beta,2} & g_{\beta,1} & g_{\beta,0} \\ g_{\beta,n_x} & g_{\beta,n_x-1} & \cdots & \cdots & & g_{\beta,2} & g_{\beta,1} \end{bmatrix}}_{\mathbf{T}_\beta^{n_x}} \begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{n-1}^{n+1} \\ u_n^{n+1} \end{bmatrix}.$$

Following [16] we approximate the spatial derivative of order $1 \leq \beta \leq 2$ using the symmetric Riesz derivative taken as the weighted sum of the left- and right-sided Riemann-Liouville fractional derivative. Hence, we obtain the differentiation matrix

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

Using this notation it is easy to see that the implicit Euler method for solving (10) requires the solution of the matrix system

$$\left(\mathbf{I}^{n_x} - \tau \mathbf{L}_\beta^{n_x}\right)\mathbf{u}^{n+1} = \mathbf{u}^n + \tau \mathbf{f} \tag{11}$$

at every time-step. We discuss the efficient solution of this system in Section 4.

3.2 Problem 2

The second problem we consider includes an additional time-fractional derivative, which leads to the following equation

$${}_0^C D_t^\alpha u(x,t) - {}_x D_R^\beta u(x,t) = f(x,t). \tag{12}$$

together with a zero initial condition. 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 Grünwald-Letnikov approximation in Section 2.2. We first discretize in time to get

$$\mathbf{T}_\alpha^{n_t+1}\begin{bmatrix} u(x,t_0)\\ u(x,t_1)\\ u(x,t_2)\\ \vdots\\ u(x,t_{n_t})\end{bmatrix} - \mathbf{I}^{n_t+1}\begin{bmatrix} {}_xD_R^\beta u(x,t_0)\\ {}_xD_R^\beta u(x,t_1)\\ {}_xD_R^\beta u(x,t_2)\\ \vdots\\ {}_xD_R^\beta u(x,t_{n_t})\end{bmatrix} = \mathbf{I}^{n_t+1}\begin{bmatrix} f(x,t_0)\\ f(x,t_1)\\ f(x,t_2)\\ \vdots\\ f(x,t_{n_t})\end{bmatrix}, \tag{13}$$

where

$$\mathbf{T}_\alpha^{n_t+1} := \begin{bmatrix} g_{\alpha,0} & & & & & \\ g_{\alpha,1} & g_{\alpha,0} & & & & \\ g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} & & & \\ \ddots & \ddots & \ddots & \ddots & & \\ \ddots & \ddots & \ddots & g_{\alpha,1} & g_{\alpha,0} & \\ g_{\alpha,n_t} & \cdots & \cdots & g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0}\end{bmatrix} = \left[\begin{array}{c|c} g_{\alpha,0} & 0 \\ \hline \vdots & \mathbf{T}_\alpha^{n_t} \\ g_{\alpha,n_t} & \end{array}\right]$$

Here, $\mathbf{T}_\alpha^{n_t+1}$ represents the discrete Caputo derivative. We now want to incorporate the initial condition as we know $u(x,t_0)=u_0$ and hence eliminate the first row from (13) and incorporate the initial condition into the right-hand side to get

$$\mathbf{T}_\alpha^{n_t}\begin{bmatrix} u(x,t_1)\\ u(x,t_2)\\ \vdots\\ u(x,t_{n_t})\end{bmatrix} - \mathbf{I}^{n_t}\begin{bmatrix} {}_xD_R^\beta u(x,t_1)\\ {}_xD_R^\beta u(x,t_2)\\ \vdots\\ {}_xD_R^\beta u(x,t_{n_t})\end{bmatrix} = \begin{bmatrix} f(x,t_1)\\ f(x,t_2)\\ \vdots\\ f(x,t_{n_t})\end{bmatrix} - \begin{bmatrix} g_{\alpha,1}\\ g_{\alpha,2}\\ \vdots\\ g_{\alpha,n_t}\end{bmatrix} u(x,t_0). \tag{14}$$

If we now discretize in space and recall the zero initial condition, the first line in (14), namely

$$g_{\alpha,0}u(x,t_1) - {}_xD_R^\beta u(x,t_1) = f(x,t_1)$$

becomes

$$g_{\alpha,0}\sum_{i=1}^{n_x} u_i^1 e_i - \mathbf{L}_\beta^{n_x}\begin{bmatrix} u_0^1\\ u_1^1\\ \vdots\\ u_{n_x}^1\end{bmatrix} = \sum_{i=1}^{n_x} f_i^1 e_i - g_{\alpha,2}\sum_{i=1}^{n_x} u_i^0 e_i \tag{15}$$

$$g_{\alpha,0}\mathbf{I}^{n_x}\underbrace{\begin{bmatrix} u_1^1\\ u_2^1\\ \vdots\\ u_{n_x}^1\end{bmatrix}}_{\mathbf{u}_0} - \mathbf{L}_\beta^{n_x}\begin{bmatrix} u_0^1\\ u_1^1\\ \vdots\\ u_{n_x}^1\end{bmatrix} = \mathbf{I}^{n_x}\mathbf{f}_1 - g_{\alpha,2}\mathbf{I}^{n_x}\mathbf{u}_0, \tag{16}$$

where $\mathbf{L}_\beta^{n_x} = \frac{1}{2}\left(\mathbf{T}_\beta^{n_x} + (\mathbf{T}_\beta^{n_x})^T\right)$ is the discrete version of the symmetric Riesz derivative as introduced in Problem 1. This space-time discretization leads to the following algebraic linear system in Kronecker form

$$\left((\mathbf{T}_\alpha^{n_t} \otimes \mathbf{I}^{n_x}) - (\mathbf{I}^{n_t} \otimes \mathbf{L}_\beta^{n_x})\right) \mathbf{u} = \tilde{\mathbf{f}} \tag{17}$$

where, using MATLAB notation, $\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \ldots, \mathbf{u}_{n_t}]$, and each $\mathbf{u}_i$ is a vector of dimension n_x associated with the point i in time. Similarly, we define $\tilde{\mathbf{f}} = [\mathbf{f}_1; \mathbf{f}_2; \ldots, \mathbf{f}_{n_t}]$. 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 (17) as

$$\mathbf{U}(\mathbf{T}_\alpha^{n_t})^T - \mathbf{L}_\beta^{n_x} \mathbf{U} = \mathbf{F}, \tag{18}$$

which shows that $\mathbf{U}$ is the solution to a Sylvester matrix equation (see, e.g., [18]), with $\mathbf{T}_\alpha^{n_t}$ lower triangular and $\mathbf{L}_\beta^{n_x}$ a dense symmetric matrix. In Section 4 we shall exploit this key connection to efficiently determine a numerical approximation to $\mathbf{U}$ and thus to $\mathbf{u}$, by working with (18) instead of the much larger problem in (17).

3.3 Problem 3

The next problem is a two-dimensional version of the first problem given by

$$\frac{du(x,y,t)}{dt} - {}_xD_R^{\beta_1} u(x,y,t) - {}_yD_R^{\beta_2} u(x,y,t) = f(x,y,t), \tag{19}$$

with Dirichlet boundary conditions and a zero initial condition. Using again the shifted Grünwald finite difference for the spatial derivatives, gives in the x-direction

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

and then in the y-direction

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

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

$$\frac{1}{\tau}\left(\mathbf{u}^{n+1} - \mathbf{u}^n\right) = \underbrace{\left(\mathbf{I}^{n_y} \otimes \mathbf{L}_{\beta_1}^{n_x} + \mathbf{L}_{\beta_2}^{n_y} \otimes \mathbf{I}^{n_x}\right)}_{\mathbf{L}_{\beta_1,\beta_2}^{n_x n_y}} \mathbf{u}^{n+1} + \mathbf{f}. \tag{20}$$

To proceed with the time stepping, one now has to solve the large linear system of equations

$$\left(\mathbf{I}^{n_x n_y} - \tau \mathbf{L}_{\beta_1,\beta_2}^{n_x n_y}\right) \mathbf{u}^{n+1} = \mathbf{u}^n + \tau \mathbf{f}. \tag{21}$$

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

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

which has a now familiar Kronecker structure. Proceeding as for Problem 2, the linear system $(\mathbf{I}^{n_x n_y} - \tau \mathbf{L}^{n_x n_y}_{\beta_1,\beta_2})\mathbf{u}^{n+1} = \widehat{\mathbf{f}}$, with $\widehat{\mathbf{f}} = \mathbf{u}^n + \tau \mathbf{f}$ is equivalent to the following Sylvester matrix equation,

$$\left(\frac{1}{2}\mathbf{I}^{n_x} - \tau \mathbf{L}^{n_x}_{\beta_1}\right)\mathbf{U} + \mathbf{U}\left(\frac{1}{2}\mathbf{I}^{n_y} - \tau \mathbf{L}^{n_y}_{\beta_2}\right) = \widehat{\mathbf{F}}, \tag{22}$$

in the sense that $\mathbf{u}^{n+1}$ can be obtained from the solution of (22) as $\mathbf{u}^{n+1} = \mathbf{vec}(\mathbf{U})$; here $\widehat{\mathbf{F}}$ is such that $\widehat{\mathbf{f}} = \mathbf{vec}(\mathbf{F})$. Note that $\mathbf{L}^{n_x}_{\beta_1}$ and $\mathbf{L}^{n_y}_{\beta_2}$ are both symmetric, and are in general different.

Once again, we postpone the discussion of how to solve this matrix equation efficiently at each time step, so as to obtain an approximation to $\mathbf{u}^{n+1}$, to Section 4.

3.4 Problem 4

Using the above introduced methodology, in this section we discuss the case when a fractional derivative is used also in time. We assume again that the Caputo time-derivative is employed, so that the FDE is defined by

$${}^{C}_{0}D^{\alpha}_{t} u(x,y,t) - {}_{x}D^{\beta_1}_{R} u(x,y,t) - {}_{y}D^{\beta_2}_{R} u(x,y,t) = f(x,y,t), \tag{23}$$

with a Dirichlet boundary condition and zero initial condition. Following the same steps as in our preceding derivation, we first discretize in time

$$\mathbf{T}^{n_t+1}_{\alpha}\begin{bmatrix} u(x,y,t_0) \\ u(x,y,t_1) \\ u(x,y,t_2) \\ \vdots \\ u(x,y,t_{n_t}) \end{bmatrix} - \mathbf{I}^{n_t+1}\begin{bmatrix} {}_{x}D^{\beta_1}_{R} u(x,y,t_0) + {}_{y}D^{\beta_2}_{R} u(x,y,t_0) \\ {}_{x}D^{\beta_1}_{R} u(x,y,t_1) + {}_{y}D^{\beta_2}_{R} u(x,y,t_1) \\ {}_{x}D^{\beta_1}_{R} u(x,y,t_2) + {}_{y}D^{\beta_2}_{R} u(x,y,t_2) \\ \vdots \\ {}_{x}D^{\beta_1}_{R} u(x,y,t_{n_t}) + {}_{y}D^{\beta_2}_{R} u(x,y,t_{n_t}) \end{bmatrix} = \mathbf{I}^{n_t+1}\begin{bmatrix} f(x,y,t_0) \\ f(x,y,t_1) \\ f(x,y,t_2) \\ \vdots \\ f(x,y,t_{n_t}) \end{bmatrix}. \tag{24}$$

Analogously to Section 3.2 we incorporate boundary and initial conditions. We then eliminate the first row in (24) and consider the first row of the remaining equation,

$$g_{\alpha,0} u(x,y,t_1) - {}_{x}D^{\beta_1}_{R} u(x,y,t_1) - {}_{y}D^{\beta_2}_{R} u(x,y,t_1) = f(x,y,t_1)$$

giving for the left-hand side

$$g_{\alpha,0}\mathbf{I}^{n_x n_y}\mathbf{u}_1 - \underbrace{\left(\frac{1}{h^{\beta_1}}\mathbf{I}^{n_y} \otimes \mathbf{L}^{n_x}_{\beta_1} + \mathbf{L}^{n_y}_{\beta_2} \otimes \frac{1}{h^{\beta_2}}\mathbf{I}^{n_x}\right)}_{\mathbf{L}_{\beta_1,\beta_2}}\mathbf{u}_1.$$

Proceeding in this manner for all time-steps we ob the a space-time discretization written as the following algebraic linear system

$$\left(\mathbf{T}^{n_t}_{\alpha} \otimes \mathbf{I}^{n_x n_y} - \mathbf{I}^{n_t} \otimes \mathbf{L}^{n_x n_y}_{\beta_1,\beta_2}\right)\mathbf{u} = \tilde{\mathbf{f}}. \tag{25}$$

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 4.5.

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 3.1 to 3.4. We saw that the discretization of all problems led 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 Problem 2 and 3, and less explicitly in Problem 4. We shall mainly report on the recently developed KPIK method [19], 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 [20, 21].

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 [22] many direct solution strategies exist for the solution of Toeplitz systems that can efficiently solve these systems often recursively. We mention here [23, 24, 25, 26] 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. 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

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

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 $\mathbf{F}$[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 matrix and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues (see [27, 22]). 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, [28, 29]). 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 $\mathbf{T}$.

There exists a variety of different preconditioners for Toeplitz systems [22]. Here we focus on a classical circulant preconditioner introduced by Strang in [30]. 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 [22].

The computational strategy described above can be used to solve the linear system in (11) associated with Problem 1, namely

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

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

In [31] 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) [32], which requires matrix-vector products with the coefficient matrix; for more details on iterative Krylov subspace solvers we refer the reader to [33, 34, 35]. 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 [13].

4.2 Numerical solution of the Sylvester equation

The numerical solution of linear matrix equations of the form

$$\mathbf{AU} + \mathbf{UB} = \mathbf{C} \tag{26}$$

arises in a large variety of applications; we refer the reader to [36] 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 for $\mathbf{C} \neq 0$ is ensured if $\mathbf{A}$ and $-\mathbf{B}$ have disjoint spectra. Robust numerical procedures for solving (26) 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 [37]. 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 (18) 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

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

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 (18) 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, $\widetilde{\mathbf{U}} = \mathbf{VYW}^T$, 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 [36]. We consider the following general projection methods for (26): 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 $\widetilde{\mathbf{Y}}$. 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: assuming $\mathbf{C}$ can be written as $\mathbf{C} = \mathbf{C}_1\mathbf{C}_2^T$, Rational Krylov subspaces generate the two spaces Range($\mathbf{V}$) = 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 Range($\mathbf{W}$) = 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 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* [38]. In [19] 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 [19] for the Lyapunov matrix equation, that is (26) with $\mathbf{B} = \mathbf{A}^T$ and $\mathbf{C}_1 = \mathbf{C}_2$.

The procedure in [19] 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 [36]. 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 [39] for KPIK applied to the Lyapunov equation, and in [40] as a particular case of Rational Krylov space methods applied to the Sylvester equation; see Section 4.3.

[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($\mathbf{W} \otimes \mathbf{V}$) [36].

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.

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

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 1 *From [13], 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 [13] to get an estimate of the eigenvalues of $\mathbf{L}_\beta = \frac{1}{2}(\mathbf{T}_\beta + \mathbf{T}_\beta^T)$.

[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 [36]

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 L_β cluster towards zero, so that, for instance, the eigenvalues of the two symmetric coefficient matrices in (22) both cluster around one half as $h \to 0$. This clustering property is 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 $\mathbf{C} = c_1c_2^T$. In [40] 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 [39])

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

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 $\frac{1}{2}\mathbf{I} - L_\beta$. In light of Lemma 1, we thus expect a very fast convergence rate of the method as the mesh is refined, and this is fully confirmed in our experiments, also for $\beta_1 \neq \beta_2$.

Finally, if one were to consider the (symmetric) Kronecker formulation of the Sylvester equation in (18), that is

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

(see Section 4.4) 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 Preconditioned Low-rank Krylov methods

We already introduced the CG method as a suitable method to approximate the solution to Problem 1. In general, given the linear system $\mathcal{A}x = b$, methods such as CG [32], MINRES [41] or GMRES [42] approximate the solution x within the Krylov subspace $\mathcal{K}_l(\mathcal{A}, r_0)$. To speed up convergence, a preconditioner is usually applied [34, 43, 44, 45]. In this section we describe the use of certain preconditioned Krylov subspace solvers for the structured problems described in earlier sections, and in particular for the equation of Problem 4. We focus on the solution of the Sylvester equation (26) in Kronecker form,

$$\underbrace{(\mathbf{A} \otimes \mathbf{I}_1) + (\mathbf{I}_2 \otimes \mathbf{B})}_{\mathcal{A}} x = c, \qquad x = \mathbf{vec}(\mathbf{X}), \quad c = \mathbf{vec}(\mathbf{C}).$$

A major difficulty that arises when using this formulation is that all computed vectors in a Krylov subspace iteration are full. Therefore, only very few of them can be kept in memory. This shortcoming is particularly severe for a method like GMRES, which requires storing the whole basis of the approximation space.

One remedy is to approximate the generated vectors p by low-rank approximations $p \approx \mathbf{vec}(\mathbf{U}\mathbf{V}^T)$ with $p = \mathbf{vec}(\mathbf{P})$ and $\mathbf{P} \approx \mathbf{U}\mathbf{V}^\mathbf{T}$. All of the before mentioned Krylov solvers proceed by applying the matrix to a vector during the iteration. We now illustrate that the process of performing $\mathcal{A}\mathbf{vec}(\mathbf{U}\mathbf{V}^T)$ can be performed by maintaining the low-rank nature of the vector $\mathbf{vec}(\mathbf{U}\mathbf{V}^T)$. If truncation can be strongly enforced without affecting the accuracy, this strategy is able to take full advantage of a low rank right-hand side $\mathbf{C}$. Consider then one instance of a matrix-vector product $\mathcal{A}v$ where $v = \mathbf{vec}(\mathbf{U}\mathbf{V}^T)$ has low rank:

$$\mathcal{A}v = (\mathbf{A} \otimes \mathbf{I}_1 + \mathbf{I}_2 \otimes \mathbf{B})\,\mathbf{vec}(\mathbf{U}\mathbf{V}^T) = \mathbf{vec}\left(\mathbf{I}_1\mathbf{U}\mathbf{V}^T\mathbf{A}^T + \mathbf{B}\mathbf{U}\mathbf{V}^T\mathbf{I}_2\right). \tag{27}$$

Using the inverse of the $\mathbf{vec}$ operator, it is easily seen that we can write the last equation as

$$[\mathbf{I}_1\mathbf{U} \quad \mathbf{B}\mathbf{U}]\,[\mathbf{A}\mathbf{V} \quad \mathbf{I}_2\mathbf{V}]^T \approx \tilde{\mathbf{U}}\tilde{\mathbf{V}}^T \tag{28}$$

where we compute $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ to be low-rank representations of the matrices on the left. This can easily be realized in practice employing a truncated singular value decomposition such as `svds` within MATLAB . Alternatively, one can also use a QR factorisation; see, e.g., [46, 47]. Although this strategy seems to be effective in practice, it should be kept in mind that if truncation is performed with a tolerance that is significantly above machine precision, the resulting method looses its standard exact precision arithmetic properties, such as orthogonality among the generated vectors.

We consider the strategy above in Problem 2 as a possible alternative to IKPIK), and we use the "truncated" version of BiCG as a solver [47]. As a preconditioner, we propose the use of a fixed number of IKPIK steps where the Toeplitz systems within IKPIK are solved using a circulant preconditioned Krylov solver. Of course, we typically consider IKPIK as a standalone solver but if this method is used as a preconditioner it is in general not necessary to compute an accurate solution. Naturally, other methods that approximately solve the Sylvester equation can be used as a preconditioner. One such candidate is the alternating direction implicit (ADI) method [48]. This method also computes low-rank solutions to Sylvester equations but needs to be equipped with a set of parameters that are not always easy to obtain. This might not be necessary when the ADI method is used as a preconditioner but is not investigated further in this paper.

4.5 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 [49, 50, 21]. General introductions to this very active field of research can be found in [51, 52] 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 [21], which is part of the the tensor train (TT) toolbox [20, 21, 53]. 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.,

$$\min \left\| \tilde{\mathbf{f}}_{TT} - \boldsymbol{\mathcal{A}}_{TT} x_{TT} \right\|. \tag{29}$$

In our context, the tensor $\boldsymbol{\mathcal{A}}$ presented in Section 3.4 is given by

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

One of the most used decompositions is the canonical decomposition of a tensor $\boldsymbol{\mathcal{A}}$ ([52]) 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 [21]. 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 [51] for references). As pointed out earlier the DMRG method transforms the problem of solving the linear system into a least squares problem (29), 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 [21].

Additionally, one can use a TT version of well-known Krylov methods such as GMRES ([42]) adapted to the TT framework (see [54]) 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 four problems presented earlier. 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.

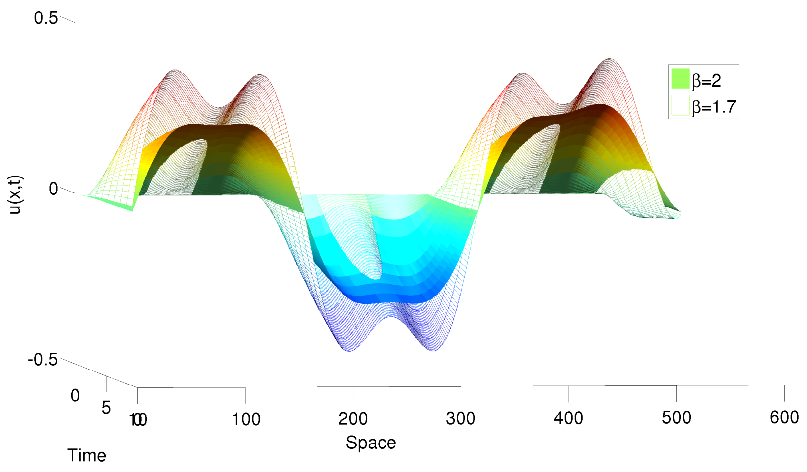


Figure 1: Problem 1. Comparison of numerical solutions when using two different values for β with zero initial condition and zero Dirichlet boundary condition. (510 and 10 space and time points, resp.)

5.1 Problem 1

We start by giving examples for the first problem. The setup here is using a zero-initial condition and a Dirichlet boundary condition $u(0,t) = 0.1$ and $u(1,t) = 0$. The result for two different values of β is shown in Figure 1 and is simply given to illustrate that the parameter β has a significant influence on the solution u. Table 1 shows the result for a variety of discretizations with two values of β. For this problem the forcing term was given by

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

with zero Dirichlet condition. It can be seen that the Strang preconditioner [30] performs exceptionally well with only 2 or 4 iteration numbers and no dependence on the mesh size. A tolerance of 10^{-6} for the relative residual was used for the stopping criterion.

DoF	$\beta = 1.3$	$\beta = 1.7$
	avg. its (time)	avg. its (time)
32768	2(1.01)	4(1.56)
65536	2(1.56)	4(2.35)
131072	2(3.54)	4(5.02)
262144	2(19.7)	4(33.98)
524288	2(21.0)	4(33.35)
1048576	2(101.6)	4(167.6)

Table 1: Problem 1. 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).

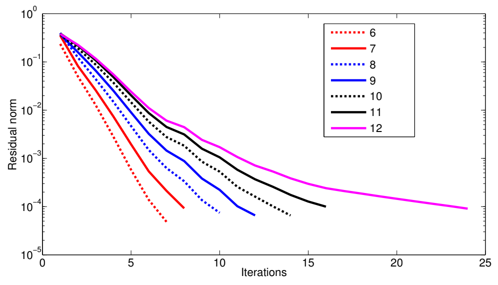


Figure 2: Problem 2. Number of IKPIK iterations to reach a tolerance of 10^{-4}. Number of space unknowns: 2^6 to 2^{12}. Convergence tolerance for the inner GMRES method: 10^{-10}.

5.2 Problem 2

The next and more challenging setup is given when studying the solution of the space-time fractional derivative formulation described earlier in Section 3.2. Here the forcing term was defined by $f = 8\sin(10x)$, while zero Dirichlet condition on the left boundary and $u = 0.01$ on the right side of the spatial interval were used. Figure 2 reports the iteration numbers of IKPIK as the space interval is refined; a tight tolerance of 10^{-10} for the inner solver that applies the inverse within IKPIK was enforced. Thanks to the good performance of the circulant inner preconditioner, this tolerance was achieved in very few inner iterations, without significantly affecting the overall performance.

The outer tolerance was set to 10^{-4}. Table 2 summarizes the results for IKPIK. The performance of the low-rank preconditioned BICG method is also reported for comparison, as described at the end of Section 4.4. The preconditioner within BICG employs 8 (fixed) iterations of IKPIK (fewer if an inner residual norm of 10^{-6} is achieved earlier).

Here we discretized both the temporal and the spatial domain using the same number of elements. 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 2 this leads to roughly 70 million unknowns.

Both methods are quite robust with respect to the number of unknowns and also with respect to varying the fractional derivative. IKPIK greatly outperforms BICG in the number of matrix-vector products and applications of the preconditioner. This is due to the rank-increase within an

		Bicg		iKPIK	
n_t	n_x	$\beta = 1.3$ it(CPUtime) #A/#P	$\beta = 1.7$ it(CPUtime) #A/#P	$\beta = 1.3$ it(CPUtime) #A/#P	$\beta = 1.7$ it(CPUtime) #A/#P
512	512	2 (*13.37*) 1954/1154	2 (*14.11*) 1524/876	15 (*1.28*) 220/160	16 (*1.27*) 217/153
1024	1024	2 (*22.28*) 1995/1191	2 (*19.57*) 1544/892	18 (*2.45*) 262/190	17 (*2.23*) 230/162
2048	2048	2 (*48.51*) 2147/1275	2 (*43.99*) 1545/893	22 (*5.73*) 339/251	18 (*4.41*) 250/178
4096	4096	2 (*162.80*) 2336/1396	2 (*113.5*) 1547/887	26 (*22.99*) 401/297	19 (*16.84*) 272/196
8192	8192	2 (*547.05*) 2336/1396	2 (*377.02*) 1571/907	30 (*75.74*) 461/341	18 (*42.20*) 262/190

Table 2: Problem 2. Preconditioned low-rank BICG and IKPIK, for a variety of meshes and two different values of β. Shown are the iteration numbers and the total CPU time, together with the number of matrix-vector products (#A) and the number of preconditioner evaluations (#P).

iteration of Bicg which then requires a much higher computational cost in the next iteration. We additionally challenged IKPIK on a larger problem where we have 32766 degrees of freedom in space and the same number in the temporal domain. This would lead to an overall linear system size $32766^2 = \mathbf{1,073,676,288}$, roughly a trillion unknowns that IKPIK was able to solve in 16 iterations and in 515 seconds for $\beta = 1.7$.

5.3 Problem 3

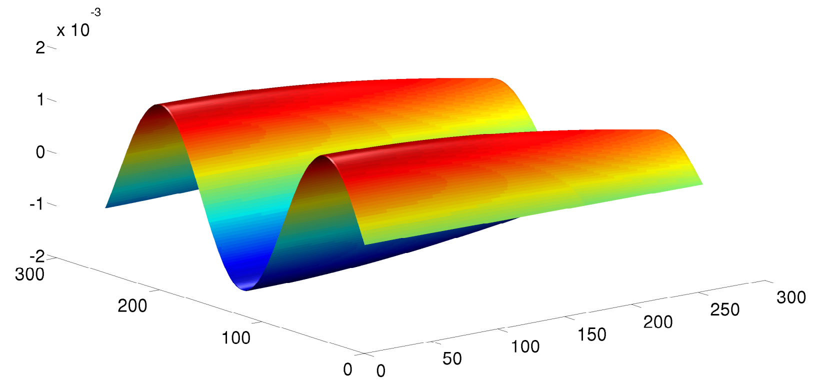
The third problem now includes a second spatial dimension together with a standard derivative in time. The spatial domain is the unit square. The problem is characterized by a zero-Dirichlet condition with a zero-initial condition. The time-dependent forcing term is given as

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

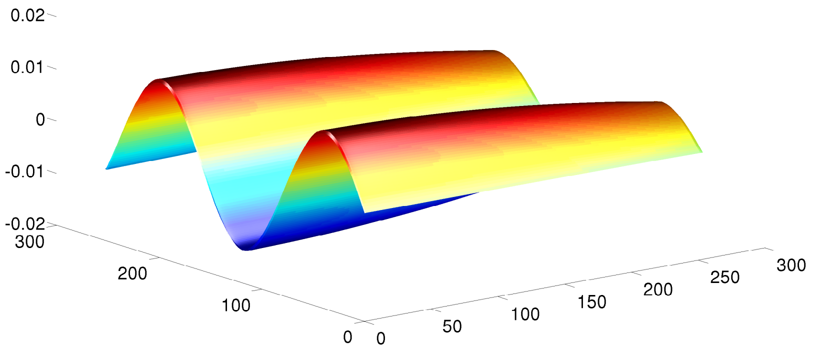
and the solution for this setup is illustrated in Figure 3, where we show the solution at two different time steps.

Table 3 shows the average IKPIK iteration numbers alongside the total number of matrix vector-products and preconditioner applications, as the two-dimensional spatial grid is refined, for different values of the fractional derivative in the two space directions[6]. The tolerance for iKPIK is set to 10^{-8}, to illustrate that we can compute the solution very accurately, and the tolerance for the inner iterative solver is chosen to be 10^{-8}. The digits summarized in Table 3 give the average number of IKPIK iterations for 8 time-steps as well as the number of matrix-vector products and preconditioner

[6]We do not report our numerical experiments with BICG on this problem as they are once again inferior to those of IKPIK

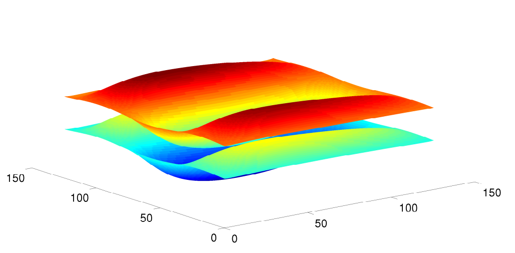


(a) First time-step

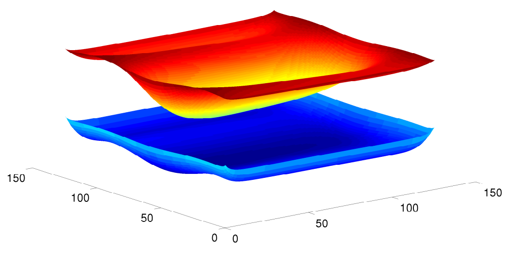


(b) Tenth time-step

Figure 3: Problem 3. First and tenth time-step solutions of the $2D$ fractional differential equation.



(a) $\alpha = 0.3$



(b) $\alpha = 0.7$

Figure 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.

application per time-step. Note that the largest problem given in Table 3 corresponds to $33,554,432$ unknowns. A further increase in the matrix dimensions, which is no problem for our methodology, would make it no longer possible to store the unknowns without being represented in low-rank format.

n_y	n_x	$\beta_1 = 1.3, \beta_2 = 1.9$	$\beta_1 = 1.7, \beta_2 = 1.9$
		aver.it/#A/#P	aver.it/#A/#P
256	512	2.25/59.00/44.00	2.50/108.12/84.62
256	1024	2.25/59.00/44.00	3.00/135.37/104.37
256	4096	2.25/59.00/44.00	3.37/155.00/120.50
1024	8192	2.00/28.75/20.75	2.87/104.62/81.12
4096	8192	2.00/26.00/18.00	2.25/66.12/50.12

Table 3: Problem 3. Performance of IKPIK as the mesh is refined for two different values of (β_1, β_2), in terms of average iteration numbers, total number of matrix-vector products and number of preconditioner evaluations.

5.4 Problem 4

In this section we briefly illustrate how the tensorized problems can be solved. Our implementation is based on the recently developed tensor-train format [20, 21, 53]. 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^{-2}. The DMRG method recalled in Section 4.5 ([21]) was used, with a convergence tolerance set to 10^{-6}. Figure 4 illustrates the different behaviour of the solution for two different values of the temporal differentiation order.

We next illustrate that the performance of DMRG for our tensor equation is robust with respect to changes in the system parameters such as the orders of differentiation and varying meshes. Table 4 shows the DMRG 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

			$\beta_1 = 1.3, \beta_2 = 1.5$	$\beta_1 = 1.7, \beta_2 = 1.9$	$\beta_1 = 1.9, \beta_2 = 1.1$
			$\alpha = 0.3$	$\alpha = 0.7$	$\alpha = 0.2$
n_t	n_y	n_x	it	it	it
512	256	512	6	6	6
256	512	512	6	7	6
1024	512	2048	6	7	11
1024	1024	4096	6	7	11

Table 4: Problem 4. Performance of DMRG as the mesh is refined, for different values of the differentiation orders.

also possible to include a preconditioner within DMRG but we have not done so here. Additionally, it would be desirable to investigate the use of a preconditioned Krylov TT solver that we can equip with a preconditioner based on approximations of the Toeplitz matrices.

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 the discretized version of the space-, time- and space-time-fractional derivative. For all problems it was crucial to notice the Toeplitz-structure of the discrete differentiation operator. While the first 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.

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] R. Gorenflo, F. Mainardi, Fractional calculus: integral and differential equations of fractional order, arXiv preprint arXiv:0805.3823.

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

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

[4] C. Wex, A. Stoll, M. Fröhlich, S. Arndt, H. Lippert, How preservation time changes the linear viscoelastic properties of porcine liver, Biorheology 50 (3) (2013) 115–131.

[5] I. Jesus, J. Machado, J. Cunha, Fractional electrical impedances in botanical elements, Journal of Vibration and Control 14 (9-10) (2008) 1389–1402.

[6] I. Podlubny, I. Petraš, B. Vinagre, P. O'leary, L. Dorčák, Analogue realizations of fractional-order controllers, Nonlinear dynamics 29 (1-4) (2002) 281–296.

[7] C. Tarley, G. Silveira, W. dos Santos, G. Matos, E. da Silva, M. Bezerra, M. Miró, S. Ferreira, Chemometric tools in electroanalytical chemistry: methods for optimization based on factorial design and response surface methodology, Microchemical journal 92 (1) (2009) 58–67.

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

[9] 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.

[10] M. Meerschaert, C. Tadjeran, Finite difference approximations for fractional advection–dispersion flow equations, Journal of Computational and Applied Mathematics 172 (1) (2004) 65–77.

[11] G. Fix, J. Roof, Least squares finite-element solution of a fractional order two-point boundary value problem, Computers & Mathematics with Applications 48 (7) (2004) 1017–1033.

[12] F. Liu, V. Anh, I. Turner, Numerical solution of the space fractional Fokker–Planck equation, Journal of Computational and Applied Mathematics 166 (1) (2004) 209–219.

[13] S.-L. Lei, H.-W. Sun, A circulant preconditioner for fractional diffusion equations, Journal of Computational Physics.

[14] T. Moroney, Q. Yang, A banded preconditioner for the two-sided, nonlinear space-fractional diffusion equation, Computers & Mathematics with Applications.

[15] K. Burrage, N. Hale, D. Kay, An efficient implicit FEM scheme for fractional-in-space reaction-diffusion equations, SIAM Journal on Scientific Computing 34 (4) (2012) A2145–A2172.

[16] I. Podlubny, A. Chechkin, T. Skovranek, Y. Chen, B. Vinagre Jara, Matrix approach to discrete fractional calculus II: Partial fractional differential equations, Journal of Computational Physics 228 (8) (2009) 3137–3153.

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

[18] P. Lancaster, M. Tismenetsky, Theory of matrices, Vol. 2, Academic press New York, 1969.

[19] V. Simoncini, A new iterative method for solving large-scale Lyapunov matrix equations, SIAM J. Sci. Comput. 29 (3) (2007) 1268–1288.

[20] I. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317.

[21] I. Oseledets, DMRG approach to fast linear algebra in the TT-format, Comput. Methods Appl. Math. 11 (3) (2011) 382–393.

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

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

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

[25] I. Gohberg, I. FelÊździman, Convolution equations and projection methods for their solution, Vol. 41, AMS Bookstore, 2005.

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

[27] M. Chen, On the solution of circulant linear systems., SIAM J. Numer. Anal. 24 (1987) 668–683. doi:10.1137/0724044.

[28] J. Cooley, J. Tukey, An algorithm for the machine calculation of complex Fourier series, Mathematics of computation 19 (90) (1965) 297–301.

[29] M. Frigo, 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.

[30] G. Strang, A proposal for Toeplitz matrix calculations, Studies in Applied Mathematics 74 (2) (1986) 171–176.

[31] H. Wang, K. Wang, T. Sircar, A direct $O(Nlog^2N)$ finite difference method for fractional diffusion equations, Journal of Computational Physics 229 (21) (2010) 8095 – 8104. doi:http://dx.doi.org/10.1016/j.jcp.2010.07.011.

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

[33] 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.

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

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

[36] V. Simoncini, Computational methods for linear matrix equations, Tech. rep., Università di Bologna (March 2013).

[37] R. H. Bartels, G. W. Stewart, Algorithm 432: Solution of the Matrix Equation $AX + XB = C$, Comm. of the ACM 15 (9) (1972) 820–826.

[38] V. Druskin, L. Knizhnerman, Extended Krylov subspaces: approximation of the matrix square root and related functions, SIAM J. Matrix Anal. Appl. 19 (3) (1998) 755–771.

[39] L. Knizhnerman, V. Simoncini, Convergence analysis of the Extended Krylov Subspace Method for the Lyapunov equation, Numerische Mathematik 118 (3) (2011) 567–586.

[40] B. Beckermann, An Error Analysis for Rational Galerkin Projection Applied to the Sylvester Equation, SIAM J. Numer. Anal. 49 (6) (2011) 2430–2450.

[41] C. Paige, M. Saunders, Solutions of sparse indefinite systems of linear equations, SIAM J. Numer. Anal 12 (4) (1975) 617–629.

[42] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput 7 (3) (1986) 856–869.

[43] H. Elman, D. Silvester, A. Wathen, Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005.

[44] M. Benzi, G. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer 14 (2005) 1–137.

[45] M. Benzi, Preconditioning techniques for large linear systems: a survey, Journal of Computational Physics 182 (2) (2002) 418–477.

[46] D. Kressner, C. Tobler, Krylov subspace methods for linear systems with tensor product structure, SIAM J. Matrix Anal. Appl 31 (4) (2010) 1688–1714.

[47] M. Stoll, T. Breiten, A low-rank in time approach to PDE-constrained optimization, Submitted.

[48] P. Benner, R.-C. Li, N. Truhar, On the ADI method for Sylvester equations, Journal of Computational and Applied Mathematics 233 (4) (2009) 1035–1045.

[49] C. Tobler, Low-rank tensor methods for linear systems and eigenvalue problems, Ph.D. thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20320, 2012 (2012).

[50] J. Ballani, L. Grasedyck, A projection method to solve linear systems in tensor format, Numerical Linear Algebra with Applications 20 (1) (2013) 27–43.

[51] L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, arXiv preprint arXiv:1302.7121.

[52] T. Kolda, B. Bader, Tensor decompositions and applications, SIAM Review 51 (3) (2009) 455–500.

[53] I. Oseledets, E. Tyrtyshnikov, Breaking the curse of dimensionality, or how to use SVD in many dimensions, SIAM Journal on Scientific Computing 31 (5) (2009) 3744–3759.

[54] S. Dolgov, TT-GMRES: solution to a linear system in the structured tensor format, Russian Journal of Numerical Analysis and Mathematical Modelling 28 (2) (2013) 149–172.