Fast Tensor Product Solvers for Optimization Problems with Fractional Differential Equations as Constraints

Sergey Dolgov\textsuperscript{a}, John W. Pearson\textsuperscript{b}, Dmitry V. Savostyanov\textsuperscript{c}, Martin Stoll\textsuperscript{a}

\textsuperscript{a}Numerical Linear Algebra for Dynamical Systems, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Germany
\textsuperscript{b}School of Mathematics, Statistics and Actuarial Science, University of Kent, Cornwallis Building (East), Canterbury, Kent, CT2 7NZ, United Kingdom
\textsuperscript{c}School of Computing, Engineering and Mathematics, University of Brighton, Moulsecoomb, Brighton, BN2 4GJ, United Kingdom

Abstract
Fractional differential equations have recently received much attention within computational mathematics and applied science, and their numerical treatment is an important research area as such equations pose substantial challenges to existing algorithms. An optimization problem with constraints given by fractional differential equations is considered, which in its discretized form leads to a high-dimensional tensor equation. To reduce the computation time and storage, the solution is sought in the tensor-train format. We compare three types of solution strategies that employ sophisticated iterative techniques using either preconditioned Krylov solvers or tailored alternating schemes. The competitiveness of these approaches is presented using several examples with constant and variable coefficients.

Keywords: Fractional calculus, Iterative solvers, Sylvester equations, Preconditioning, Low-rank methods, Tensor equations, Schur complement

1. Introduction
While the study of derivatives of arbitrary order is a long-standing subject area \cite{31}, its use in science and engineering has soared over recent years. Fractional calculus is often used due to the inadequateness of traditional schemes to describe certain phenomena, such as anomalous diffusion, anaelasticity \cite{16} and viscoelasticity \cite{45, 88}. The applications include electrical circuits \cite{40, 73}, electro-analytical chemistry \cite{84}, biomechanics \cite{28}, and image processing \cite{91}.

Over the last decade many researchers have worked on efficient numerical schemes for the discretization and solution of fractional differential equations (FDEs). Historically, the finite difference-based discretization techniques are arguably the most popular \cite{51, 52, 71, 72, 73}. Adomian decomposition should be mentioned as a popular semi-analytical approach \cite{1}, although its use is limited. Recently, (discontinuous) finite element schemes for FDEs have also received considerable attention \cite{17, 59, 90}. Since the fractional differential operators are, in fact, integral operators, the matrix of the corresponding linear system is usually

\begin{flushleft}
Email addresses: dolgov@mpi-magdeburg.mpg.de (Sergey Dolgov), J.W.Pearson@kent.ac.uk (John W. Pearson), d.savostyanov@brighton.ac.uk (Dmitry V. Savostyanov), stollm@mpi-magdeburg.mpg.de (Martin Stoll)
\end{flushleft}
dense, and the numerical complexity and storage grow rapidly with the grid size, especially
for the problems posed in higher dimensions (e.g. three spatial plus a temporal dimension).
To perform computations faster, we compress the solution in the low-rank format. This
approach has already been applied to fractional calculus in [13, 75].

Most commonly, the fractional calculus literature focuses on the solution of the equation
itself, the so-called ‘direct problem’. In this paper we consider the ‘inverse problem’, namely
the computation of the forcing term (right-hand side of the FDE), that is best suited to
describing a desired property or measured data. For this we study an optimization problem
with constraints given by FDEs. In the context of partial differential equations (PDEs),
problems of this type are often referred to as PDE-constrained optimization problems and
have been studied extensively over the last decades (see [38, 85] for introductions to the
field). Optimal control problems for FDEs have previously been studied in literature such as
[2, 8, 55, 56, 65, 74]. However, they were mostly considering one-dimensional spatial
domains, since the direct treatment of higher dimensions was too expensive. To overcome
computational challenges, in this paper we rely on the recent advances in the development
of numerical algorithms and solvers, particularly on data-sparse low-rank formats.

The goal of our paper is to present efficient numerical methods that allow the fast and
accurate solution of the large optimization problem at hand. The paper is organized as
follows. In Section 2.1 we recall some of the most important definitions needed for fractional
derivatives. This is followed by Section 2.2 where we introduce the basic optimization problem
subject to fractional differential equations posed in an increasing number of dimensions, along
with the discretization of both the objective function and the differential equation. Section
3 presents three strategies that are well-suited to solving the discretized problem. This
is followed by a discussion of numerical algorithms in Section 4 where we introduce the
tensor-train format and several iterative solvers, either of Krylov subspace type or using an
alternating framework. The effectiveness of our approach is shown in Section 5 where we
compare our solvers using several numerical experiments. We also present results for the
more challenging variable coefficient case and observe satisfying results.

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 [72] where the discretization via finite differences is
considered in a purely algebraic framework.

2. Fractional calculus and Grünwald formulae

In this section we briefly recall the concept of fractional derivatives, and use this to
state the matrix systems that result from discretizing the problems we consider using a
finite difference method. The literature on fractional derivatives is vast and we refer to
[18, 31, 35, 54, 69, 70] for general introductions to this topic.

2.1. The fractional derivative

In fractional calculus there are several definitions of fractional derivatives. The Caputo
and the Riemann-Liouville fractional derivatives [69] are among the most commonly used in
applications and we use this section to briefly recall their definitions.

For a function \( f(t) \) defined on an interval \([a, b]\), the Caputo derivative of real order \( \alpha \) with
$n - 1 < \alpha < n$, $n \in \mathbb{N}$, is defined as the following integral

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

assuming that it is convergent (see [19, 31, 51, 69, 78] for more details). Based on the discussion in [72], the Caputo derivative is frequently used for the derivative with respect to time. The left-sided Riemann-Liouville derivative of real order $\alpha$ with $n - 1 < \alpha < n$, is defined by

$$RL_{a}^{\alpha} D_t^{\alpha} f(t) = \frac{1}{\Gamma(n - \alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(s)}{(t - s)^{\alpha-n+1}},$$

for $a < t < b$. The right-sided Riemann-Liouville fractional derivative is given by

$$RL_{t}^{\alpha} D_t^{\alpha} f(t) = \frac{(-1)^n}{\Gamma(n - \alpha)} \frac{d^n}{dt^n} \int_t^b \frac{f(s)}{(s - t)^{\alpha-n+1}},$$

for $a < t < b$. Finally, the symmetric Riesz derivative of order $\alpha$ is the half-sum of the left and right-side Riemann-Liouville derivative, i.e.,

$$R_{t}^{\alpha} D_t^{\alpha} f(t) = \frac{1}{2} \left( RL_{a}^{\alpha} D_t^{\alpha} f(t) + RL_{t}^{\alpha} D_t^{\alpha} f(t) \right).$$

In this work we do not advocate a particular method that is most suitable for the description of a natural phenomenon: we simply want to illustrate that the above formulations of FDEs, when coupled with certain types of discretization approaches, lead to similar structures on the discrete level. Our goal is to give guidelines and offer numerical schemes for the efficient and accurate solution of problems of various forms. For a discussion on the smoothness assumptions of the function $f(\cdot)$ we refer to [69], i.e., the Riemann-Liouville formulation requires a weaker differentiability assumption in contrast to the $n$-times differentiability required otherwise (cf. [69] Chapter 2.3).

2.2. Model problems

In this section we introduce some FDE-constrained optimization problems. Consider the classical misfit problem, where we want to minimize the difference between the state $y$ and the desired state (or observation) $\bar{y}$, with an additional regularization representing the cost of the control $u$. The constraint linking the state to the control is given by the following FDE

$$\begin{align*}
(0D_t^\alpha - R_{x_1}^{\beta_1}) y(x, t) + u(x, t) &= f(x, t), \\
(0D_t^\alpha - R_{x_2}^{\beta_2}) y(x_1, x_2, t) + u(x_1, x_2, t) &= f(x_1, x_2, t), \\
(0D_t^\alpha - R_{x_3}^{\beta_3}) y(x_1, x_2, x_3, t) + u(x_1, x_2, x_3, t) &= f(x_1, x_2, x_3, t)
\end{align*}$$

in one, two, and three dimensions, respectively. We consider the FDEs in space-time cylinder domains $Q := \Omega \times [0, T]$, with $\Omega \subseteq \mathbb{R}^d$, $d \in \{1, 2, 3\}$. We also denote $\text{supp}(\bar{y}) = Q_\bar{y}$ and $\text{supp}(u) = Q_u$ and assume that both the observation and the control are supported over the cylinders $Q_\bar{y} = \Omega_\bar{y} \times [0, T]$ and $Q_u = \Omega_u \times [0, T]$. Our cost function is, therefore

$$J(y, u) := \frac{1}{2} \iint_{Q_\bar{y}} (y - \bar{y})^2 dx dt + \frac{\gamma}{2} \iint_{Q_u} u^2 dx dt$$

(2)
where \( \gamma \) is a regularization parameter indicating at what ratio one prioritizes minimizing \( u \), and obtaining \( y \) that is close in some sense to the desired state \( \bar{y} \).

We will follow the discretize-then-optimize approach for PDE-constrained optimization problems [11, 36] (details on the numerical treatment of FDEs can be found in [51, 52, 69, 71, 72]). We consider \( \Omega = [0, 1]^d \) and assume that \( \Omega_u \) and \( \Omega_y \) are also cubes.

Each function is discretized by collocation on a uniform tensor product grid in space and time

\[
x_k(i_k) = i_k h_k, \quad i_k = 1, \ldots, n_k, \quad h_k = \frac{1}{n_k + 1}, \quad k = 1, 2, 3, \quad \text{and} \quad t(m) = m \tau, \quad m = 1, \ldots, n_t, \quad \tau = \frac{T}{n_t}.
\]

(3)

After discretization, vectors containing values of functions \( y, \bar{y}, u, f \) on the grid are denoted also as \( y, \bar{y}, u, f \), respectively. We assume zero boundary and initial conditions, \( y(x_1, \ldots, x_d, t) = 0 \) if \( x_k \leq 0, x_k \geq 1 \) or \( t \leq 0 \). Note that it is possible to include inhomogeneous boundary and initial conditions [50, 69]. This typically does not affect the matrix structure and so the methods presented here also apply for this setup. If the grid sizes are the same in all spatial directions, we denote \( h = h_1 = h_2 = h_3 \).

Note that our formulations here are tailored to domains that allow for tensorized grids, as we aim to separate the variables with respect to all spatial and temporal dimensions. While this does exclude certain applications, many interesting and relevant models can still be considered such as fractional phase-field models [12, 14, 15] or the fractional Fokker-Planck equation [49]. They also appear in many applications such as the anomalous diffusion [53], pattern formation using fractional derivatives [29], and also the simulation of fractional phase-field equations such as the Allen-Cahn equation [15]. Additionally, the low-rank approximation presented in this paper also applies in the case where the spatial domain is more complicated. In this case one can often still separate spatial and temporal dimensions, or other parameter dimensions, which are often found for problems with uncertainties and/or model parameters.

2.3. Grünwald-Letnikov formula

A classical method for the discretization of FDEs is based on the formula of Grünwald and Letnikov here illustrated for the one-dimensional case. In particular, for the spatial derivative we use its shifted version

\[
\frac{\partial }{\partial x} D_x^\beta y(x) \approx \frac{1}{h^\beta} \sum_{i=0}^{n} g_{\beta,i} y(x - (i - 1)h)
\]

(4)

which is proven to be of the first order [50, 51, 52], provided that \( y \in W_1^{1+\beta}(\mathbb{R}) \). The coefficients \( g_{\beta,i} \) are defined by

\[
g_{\beta,i} = \frac{\Gamma(i - \beta)}{\Gamma(-\beta) \Gamma(i + 1)} = (-1)^i \binom{\beta}{i},
\]

(5)

and can be computed efficiently using the recurrent formula [69]: \( g_{\beta,0} = 1, g_{\beta,i} = (1 - \frac{\beta + 1}{i}) g_{\beta,i-1} \) for \( i = 1, 2, \ldots, n \).
Collecting all degrees of freedom into one matrix, we obtain the discretization of the spatial Riemann-Liouville derivative as follows

\[
\begin{bmatrix}
g_{\beta,1} & g_{\beta,0} & 0 & \ldots & \ldots & \ldots & 0 \\
g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & \ddots & \vdots & & \vdots \\
g_{\beta,3} & g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & \ddots & \vdots & \vdots \\
& \ddots & g_{\beta,2} & g_{\beta,1} & \ddots & \vdots & \vdots \\
& & \ddots & \ddots & \ddots & \vdots & \vdots \\
& & & \ddots & \ddots & \ddots & \vdots \\
g_{\beta,n} & g_{\beta,n-1} & \ldots & \ldots & g_{\beta,3} & g_{\beta,2} & g_{\beta,1} \\
\end{bmatrix}
\rightarrow
h^{-\beta}
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
\vdots \\
\vdots \\
\vdots \\
y_{n-1} \\
y_n \\
\end{bmatrix}
\]

(6)

The matrix \( T_\beta \) is a Toeplitz matrix, which we discuss later in more detail. As in [72] we approximate the spatial derivative of order \( 1 \leq \beta \leq 2 \) using the symmetric Riesz derivative taken as the half sum of the left- and right-sided Riemann-Liouville fractional derivatives. The resulting differentiation matrix is given by

\[
L_\beta := \frac{1}{2} (T_\beta + T_\beta^T).
\]

The discretization in time is done via the non-shifted Grünwald–Letnikov formula [69, 78]

\[
C_0 D_\alpha^\tau y(t) \approx \frac{1}{\tau^\alpha} \sum_{m=0}^{n-1} g_{\alpha,m} y(t - m\tau),
\]

(7)

and the Caputo derivative leads to a Toeplitz matrix \( C_\alpha \) of the lower triangular form

\[
C_\alpha = \tau^{-\alpha}
\begin{bmatrix}
g_{\alpha,0} & 0 & \ldots & \ldots & \ldots & 0 \\
g_{\alpha,1} & g_{\alpha,0} & \ddots & \vdots & & \vdots \\
g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} & \ddots & \vdots & \vdots \\
& \ddots & \ddots & \ddots & \vdots & \vdots \\
& & \ddots & \ddots & \ddots & \vdots \\
& & & \ddots & \ddots & \ddots & \vdots \\
g_{\alpha,n-1} & g_{\alpha,n-2} & \ldots & \ldots & g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} \\
\end{bmatrix}
\]

(8)

2.4. Problem structure

Since the spatial grids introduced on \( \Omega, \Omega_y \) and \( \Omega_u \) have Cartesian product structure, e.g., these domains are cubes in \( \mathbb{R}^d \), the discretized differential operators in (1b) and (1c) can be written using the tensor-product notation,

\[
R^D_{x_1} + R^D_{x_2} \rightarrow L = L_{\beta_1} \otimes I_{n_2} + I_{n_1} \otimes L_{\beta_2},
\]

\[
R^D_{x_1} + R^D_{x_2} + R^D_{x_3} \rightarrow L = L_{\beta_1} \otimes I_{n_2} \otimes I_{n_3} + I_{n_1} \otimes L_{\beta_2} \otimes I_{n_3} + I_{n_1} \otimes I_{n_2} \otimes L_{\beta_3},
\]

(9)

for \( d = 2 \) and \( d = 3 \), respectively. Here and later \( I_n \) denotes an identity matrix of size \( n \).
Asuming that $L$ denotes the discretization of spatial derivatives \( [9] \), we obtain the discretization of the FDEs as follows

\[
Ay + M_3u = g, \quad A = C_\alpha \otimes I_n - I_{n_t} \otimes L,
\]

(10)

where $n$ denotes the total number of space mesh points, $M_3u$ is the discretization of the control term, and $g$ represents boundary conditions, initial conditions and additional forcing terms. Note that vectors $y$ and $g$ are of the size $N = nn_t$, where $n = n_1n_2$ for a two-dimensional problem and $n = n_1n_2n_3$ for a three-dimensional problem. The vector $u$ is formally of a smaller size, since we only consider its non-zero elements, which are all on $Q_u \subseteq Q$. Denoting the numbers of grid points on $\Omega_u$ as $n'$, and hence the number of grid points in $Q_u$ is $N' = n'n_t$, we conclude that $M_3$ is the $N \times N'$ matrix. In our case of the collocation discretization, $M_3$ contains unitary rows at positions, corresponding to the grid points in $Q_u$.

Using standard integration rules, we discretize the cost function (2) as follows

\[
J(y, u) = \frac{1}{2} (y - \bar{y})^\top M_1 (y - \bar{y}) + \frac{\gamma}{2} u^\top M_2 u,
\]

(11)

where $M_1$ and $M_2$ contain the quadrature weights used to evaluate the norm. We use the trapezoidal rule, coinciding with the rectangular rule due to zero boundary conditions. We make the vector $\bar{y}$ of size $N$. Since the function $\bar{y}(x_1, x_2, x_3, t)$ is defined only on $\Omega_\bar{y}$, we populate $\bar{y}$ by zeros at the remaining points, lying in $\Omega \setminus \Omega_\bar{y}$. Let us denote by $\theta_{\bar{y}} \in \mathbb{R}^N$ a binary vector, containing 1’s at indices belonging to $\Omega_\bar{y}$. Then

\[
M_1 = \tau h_1 \cdots h_d \cdot \text{diag} (\theta_{\bar{y}}), \quad \text{and} \quad M_2 = \tau h_1 \cdots h_d \cdot I_{N'}.
\]

(12)

A standard Lagrangian approach results in the following discrete functional for which we need to compute stationary points:

\[
\Lambda(y, u, p) = J(y, u) + p^\top V^\top (Ay + M_3u - g),
\]

(13)

where $p$ corresponds to the discretized adjoint variable $p$. The matrix $V$ is used to associate $p$ with a grid function (see [11]), and is written similarly to (12), $V = \tau h_1 \cdots h_d \cdot I_N$. To obtain the required optimality conditions, we differentiate $\Lambda$ with respect to $y$, $u$ and $p$. This leads to the following first order, Karush-Kuhn-Tucker (KKT), system:

\[
\begin{bmatrix}
M_1 & 0 & A^\top V \\
0 & \gamma M_2 & M_3^\top V \\
V^\top A & V^\top M_3 & 0
\end{bmatrix}
\begin{bmatrix}
y \\ u \\ p
\end{bmatrix}
= \begin{bmatrix}
M_1 \bar{y} \\
0 \\
V^\top g
\end{bmatrix} = : \begin{bmatrix}
f_1 \\
f_2 \\
f_3
\end{bmatrix},
\]

(14)

For simplicity, we merge $V$ into both $A$ and $M_3$ and proceed with the following system matrix:

\[
A = \begin{bmatrix}
M_1 & 0 & A^\top \\
0 & \gamma M_2 & M_3^\top \\
A & M_3 & 0
\end{bmatrix}.
\]

The invertibility of the matrix follows from the constraint block $[A \quad M_3]$ having full-rank, and the fact that the $(1, 1)$-block is positive definite on the kernel of the constraints, i.e.,

\[
[-u^\top M_3^\top A^{-\top} \quad u^\top] 
\begin{bmatrix}
M_1 & 0 \\
0 & \gamma M_2
\end{bmatrix} 
\begin{bmatrix}
-A^{-1}M_3u \\
u
\end{bmatrix} = u^\top M_3^\top A^{-\top} M_1 A^{-1} M_3 u + \gamma u^\top M_2 u > 0
\]

(see [10]). There are various ways to solve the saddle point system and in the next section we discuss several approaches.
3. Solution strategies

In this section we discuss possible methods to solve the system (14). The classical monolithic approach is not applicable in our case: a direct solver is prohibitively expensive due to the problem size. Therefore, we consider three types of iterative methods: a low-rank MINRES method applied to (14) and alternating iterative methods applied to two variants of Schur complements.

3.1. Krylov solver and preconditioning

For linear systems the use of iterative methods of Krylov type [77] is well established. For standard saddle point problems in particular this is a method of choice, and is considered a state of the art when the problem is of very high dimension (see [10, 26]). Such a solver proceeds by building up a Krylov subspace

\[ \mathcal{K}_l(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{l-1}r_0\}, \]

in terms of the initial residual vector \( r_0 \). The solution at step \( l \) is then computed in some (quasi-)optimal way within the Krylov subspace \( \mathcal{K}_l(A, r_0) \).

As the convergence can sometimes be very slow the problem is modified using a preconditioning matrix \( P \), i.e.,

\[ Ay = f \iff P^{-1}Ay = P^{-1}f. \]

We again refer to [10, 26] for a detailed overview of preconditioners for saddle point problems. In this paper, we follow a result presented in [57], where it was shown that for a saddle point system

\[ \begin{bmatrix} \Phi & \Psi^\top \\ \Psi & 0 \end{bmatrix} y = f, \]

a good preconditioner can be chosen in the form

\[ P = \begin{bmatrix} \tilde{\Phi} & 0 \\ 0 & \tilde{S} \end{bmatrix} \quad \text{or} \quad P = \begin{bmatrix} \tilde{\Phi} & 0 \\ \Psi & -\tilde{S} \end{bmatrix}, \]

where \( \tilde{\Phi} \approx \Phi \) and \( \tilde{S} \approx S := \Psi \Phi^{-1} \Psi^\top \), the (negative) Schur complement. For our problem, we commence with the approximation of the following (1,1)-block:

\[ \Phi = \begin{bmatrix} M_1 & 0 \\ 0 & \gamma M_2 \end{bmatrix}. \]

In the case of the full observation and control, \( \Omega_y = \Omega_u = \Omega \), all \( M_i, i = 1, 2, 3 \), are simply scaled identity matrices. They can be easily inverted, so \( \Phi \) can be “approximated” exactly. For the Schur complement \( S = \Psi \Phi^{-1} \Psi^\top \) with the constraint matrix \( \Psi = [A \ M_3] \) it is typically much harder to find a robust approximation. For this we study the structure of the Schur complement further:

\[ S = AM_1^{-1}A^\top + \frac{1}{\gamma} M_3 M_2^{-1} M_3^\top. \tag{15} \]

The authors have recently obtained robust Schur complement approximations using a matching approach [66, 67, 83] that in our case leads to

\[ \tilde{S} = \tilde{A}M_1^{-1} \tilde{A}^\top, \quad \tilde{A} = A + \frac{1}{\sqrt{\gamma}} I_n, \otimes I_n. \tag{16} \]
Before discussing the efficient inversion of the matrix $\tilde{A}$, we briefly analyze the effectiveness of our Schur complement approximation. It is clear that the effectiveness of our solver for the matrix system depends to a large extent on how well the exact Schur complement as given in (15) is represented by our approximation (16). Following [68], we measure this by examining the Rayleigh quotient

$$R := \frac{v^T S v}{v^T \bar{S} v}, \quad v \in \mathbb{R}^N, \quad v \neq 0.$$ 

**Theorem 1.** Consider the FDE-constrained control problem (1)–(2) with Grünwald-Letnikov discretizations [6], [8] on the same cubic domain $\Omega = \Omega_g = \Omega_u$ for the observation, control and system state, with the simplifying assumption that $M_1 = M_2 = M_3$ are scaled mass matrices. Let $0 < \alpha \leq 1$ and $1 \leq \beta_i \leq 2$, $i = 1, 2, 3$. Then it holds $\frac{1}{2} \leq R < 1$.

**Proof.** Following [66], we write

$$R = \frac{a_1^T a_1 + a_2^T a_2}{(a_1 + a_2)^T (a_1 + a_2)},$$

where $a_1 = \frac{1}{\sqrt{\tau h^d}} (C_\alpha \otimes I_n - I_{n_t} \otimes L^T) v$, $a_2 = \sqrt{\frac{\gamma h^d}{\tau}} v$, using the form of $M_1$, $M_2$, $M_3$. We first note that $a_2^T a_2 = \frac{h^d}{\gamma} v^T v > 0$. This means that we may write

$$\frac{1}{2} (a_1 - a_2)^T (a_1 - a_2) \geq 0 \iff a_1^T a_1 + a_2^T a_2 \geq \frac{1}{2} (a_1 + a_2)^T (a_1 + a_2) \iff R \geq \frac{1}{2}.$$

To prove the upper bound for $R$, we now consider the quantity

$$a_1^T a_2 + a_2^T a_1 = \frac{1}{\sqrt{\gamma}} v^T ((C_\alpha + C_\alpha^T) \otimes I_n - I_{n_t} \otimes (L + L^T)) v.$$

It has been shown in [13, Lemma 1] that the matrix $I_{n_t} \otimes L$ is negative semi-definite for $1 \leq \beta_i \leq 2$ – we can therefore write that $v^T (I_{n_t} \otimes (L + L^T)) v = 2 v^T (I_{n_t} \otimes L) v \leq 0$, using the symmetry of $L$.

The matrix $C_\alpha + C_\alpha^T$ is symmetric positive definite for $\alpha < 1$, since $C_\alpha$ is diagonally dominant. It is clear that the coefficient $g_{a,k}$ is positive for $k = 0$ and negative for all $k = 1, 2, \ldots$, if $\alpha < 1$, so we need to show that $\sum_{k=0}^{\infty} g_{a,k} > 0$. This follows from the binomial expansion $(1 + z)^\alpha = \sum_{k=0}^{\infty} \binom{\alpha}{k} z^k$, evaluated at $z = -1$. In our case, $\text{Re } \alpha > 0$ and $|z| = 1$, and this series converges absolutely [34, page 397]. Therefore, the following inequality holds, $\sum_{k=0}^{n_t} g_{a,k} > \sum_{k=0}^{\infty} g_{a,k} = \sum_{k=0}^{\infty} \binom{\alpha}{k} (-1)^k = 0$. Hence, we have that $v^T ((C_\alpha + C_\alpha^T) \otimes I_n) v > 0$, using the standard property of Kronecker product that the eigenvalues of $K_1 \otimes K_2$ are equal to those of $K_1$ multiplied by those of $K_2$ [48, Chapter 13].

Combining these findings gives us that $a_1^T a_2 + a_2^T a_1 > 0$. We therefore have that

$$R = \frac{a_1^T a_1 + a_2^T a_2}{a_1^T a_1 + a_2^T a_2 + a_1^T a_2 + a_2^T a_1} < \frac{a_1^T a_1 + a_2^T a_2}{a_1^T a_1 + a_2^T a_2 + a_1^T a_2 + a_2^T a_1} = 1.$$

$\square$

The general case of incomplete observation or control is analytically much more challenging, but there are heuristic techniques [66, 67] for augmenting the matrices $M_1$ and $M_3$ which define fairly good (from computational experience) approximations $\tilde{\Phi}$ and $\tilde{S}$.
3.2. First Schur Complement

This approach requires the invertibility of the $(1, 1)$-block $\Phi$ and is mainly concerned with inverting the (negative) Schur complement matrix

$$S = AM^{-1}A^T + M_3(\gamma M_2)^{-1}M_3^T.$$  \hspace{1cm} (17)

It proceeds by solving the following three systems,

$$-Sp = f_3 - AM_1^{-1}f_1 - M_3(\gamma M_2)^{-1}f_2,$$

$$(\gamma M_2)u = f_2 - M_3^Tp,$$

$$M_1y = f_1 - A^Tp.$$  \hspace{1cm} (18)

The disadvantage of this approach is that it requires the invertibility of both observation and control matrices for the existence of $S$, which may be restrictive. However, when the matrices $M_1$ and $M_2$ are invertible, their inversion is usually very cheap, e.g. $M_1$ and $M_2$ are scaled identities. As we observe through numerical experiments, this scheme is superior when both observation and control are defined on the whole domain.

3.3. Second Schur Complement

Now we consider the case when control and observation domains have different sizes, so that $M_3$ is a rectangular matrix. In contrast to $M_1$, the matrix $M_2$ can be assumed to be non-singular, since it corresponds to the regularization term. Therefore, the following decomposition can be verified straightforwardly:

$$\begin{bmatrix}
M_1 & 0 & A^T \\
0 & \gamma M_2 & M_3^T \\
A & M_3 & 0
\end{bmatrix} = \begin{bmatrix}
I & -M_1A^{-1}M_3 & M_1 \\
0 & \gamma M_2 & 0 \\
0 & 0 & A
\end{bmatrix} \begin{bmatrix}
0 & 0 & M_1A^{-1}M_3(\gamma M_2)^{-1}M_3^T + A^T \\
0 & I & (\gamma M_2)^{-1}M_3^T \\
I & A^{-1}M_3 & 0
\end{bmatrix}.$$  \hspace{1cm} (19)

The matrix $A$ specifies the underlying FDE model, and is also invertible. Using this factorization the solution of (14) can be computed as follows. First, define

$$\begin{bmatrix}
\tilde{f}_1 \\
\tilde{f}_2 \\
\tilde{f}_3
\end{bmatrix} := \begin{bmatrix}
I & -M_1A^{-1}M_3 & M_1 \\
0 & \gamma M_2 & 0 \\
0 & 0 & A
\end{bmatrix}^{-1} \begin{bmatrix}
f_1 \\
f_2 \\
f_3
\end{bmatrix}.$$  

We compute these intermediate vectors by solving $Af_3 = f_3$, $(\gamma M_2)\tilde{f}_2 = f_2$, $Ag_2 = M_3\tilde{f}_2$ and computing $\tilde{f}_1 = f_1 + M_1g_2 - M_1\tilde{f}_3$. Note that if $f_2 = 0$ as in (14), it is trivially observed that $g_2 = 0$. Now we eliminate the second factor in (19) by solving the following systems:

$$(M_1A^{-1}M_3(\gamma M_2)^{-1}M_3^T + A^T)p = \tilde{f}_1,$$  \hspace{1cm} (20a)

$$(\gamma M_2)u = f_2 - M_3^Tp,$$  \hspace{1cm} (20b)

$$Ay = f_3 - M_3u.$$  \hspace{1cm} (20c)

Both $M_2$ and $A$ have Kronecker-product structures by assumption, hence the action of their inverses by vectors can be computed efficiently using tensor product algorithms.
The computation of \( p \) is more complicated: we have to assemble the Schur complement first, which itself requires matrix inversion. We cast this problem as the solution of a larger linear system: we assemble \( \hat{A} = A \otimes I_N \) and solve

\[
\hat{A}q = \text{vec} \left( M_3 (\gamma M_2)^{-1} M_3^T \right), \quad q = \text{vec}(Q),
\]

(21)

where \( \text{vec}(\cdot) \) stretches matrix entries to a vector. Now the matrix in (20a) becomes \( M_1 Q + A^T \).

In tensor product form, the storage complexity of \( \hat{A} \) is not much higher than that of \( A \). A certain difficulty arises from the storage of \( Q \): while \( A \) has a convenient tensor product structure, its inversion may consume a considerable amount of memory even in a compressed form. This occurs, for example, if the control is given on the entire domain, and the right-hand side in (21) is an identity matrix. Fortunately, in practically interesting cases of a small control domain, the column size of \( M_3 \) is also small. The right-hand side in (21) is a low-rank matrix and yields smaller storage requirements for \( Q \) to achieve the same accuracy.

4. Numerical algorithms

We will now discuss solvers for the matrix systems (14)–(21). While for the one-dimensional FDE the problem has similarities to saddle point problems involving matrix equations of Sylvester type [81], the higher-dimensional setup requires the use of more specialized tensor solvers. In particular we discuss the tensor-train format introduced in [62].

4.1. The tensor-train decomposition

In a one-dimensional problem, where we separate the spatial and time variable, a suitable approach for the FDE problem is a matrix based low-rank decomposition of Krylov vectors. While this is an important case that we discuss in the next section in some more detail, in higher dimensions even low-rank factors become infeasible. Further data compression can be achieved with more advanced high-dimensional tensor product decompositions. In this paper we use the simple, but robust, Tensor Train (TT) format and its extension, the Quantized Tensor Train (QTT) decomposition.

The TT format is derived by applying the low-rank approximation recurrently [62]. Given a \( d \)-index array \( y = [y(i_1, \ldots, i_d)] \), with indices varying in ranges \( i_k = 1, \ldots, n_k, k = 1, \ldots, d \), we may reshape a tensor \( y \) into a matrix \( Y_1 \in \mathbb{R}^{n_1 \times (n_2 \cdots n_d)} \) by the grouping of indices. To do this we introduce the notation

\[
\overline{i_2 \ldots i_d} = i_2 + (i_3 - 1)n_2 + \cdots + (i_d - 1)n_2 \cdots n_{d-1},
\]

(22)

and define \( Y_1(i_1, \overline{i_2 \ldots i_d}) = y(i_1, \ldots, i_d) \) for all admissible index values. Since \( Y_1 \) is a matrix, we may apply the low-rank singular value decomposition (SVD):

\[
Y_1 \approx U_1 \Sigma_1 V_1^T, \quad \text{where} \quad U_1 \in \mathbb{R}^{n_1 \times r_1}, \quad V_1 \in \mathbb{R}^{n_2 \cdots n_d \times r_1}.
\]

The first factor \( U_1 \) is of moderate dimension, and can be stored as \( y_{\alpha_1}^{(1)}(i_1) = U_1(i_1, \alpha_1) \), where \( \alpha_1 = 1, \ldots, r_1 \). The remaining matrix \( \Sigma_1 V_1^T \) depends on indices \( \alpha_1 \) and \( \overline{i_2 \ldots i_d} \). Now we regroup these indices as follows:

\[
Y_2(\overline{\alpha_1 i_2, i_3 \ldots i_d}) = \Sigma_1(\alpha_1, \alpha_1) V_1^T(\alpha_1, \overline{i_2 \ldots i_d}),
\]
and compute the next SVD,
\[ Y_2 \approx U_2 \Sigma_2 V_2^T, \quad \text{where} \quad U_2 \in \mathbb{R}^{1 \times r_2}, \ V_2 \in \mathbb{R}^{n_2 \times 1}. \]  
Again, \( U_2 \) can be reshaped to a moderately-sized 3-dimensional tensor \( y^{(2)}_{\alpha_1, \alpha_2}(i_2) = U_2(\alpha_1 i_2, \alpha_2) \), and the decomposition continued for \( \Sigma_2 V_2^T \). Finally, we arrive at the TT format:
\[ y(i_1, \ldots, i_d) \approx \sum_{\alpha_1, \ldots, \alpha_d = 1}^{r_{i_1}, \ldots, r_{i_d}} y^{(1)}_{\alpha_1}(i_1) y^{(2)}_{\alpha_1, \alpha_2}(i_2) \cdots y^{(d-1)}_{\alpha_{d-2}, \alpha_{d-1}}(i_{d-1}) y^{(d)}_{\alpha_{d-1}}(i_d), \]  
where \( r_k \) are called TT ranks. The storage size in the TT format (24) is \( O(dnr^2) \), where \( r \gtrsim r_k \) and \( n \gtrsim n_k \).

A similar construction is introduced for discretized operators in high dimensions. Given a matrix \( A = [A(i_1, \ldots, i_d; j_1, \ldots, j_d)] \in \mathbb{R}^{(n_1 \times n_d) \times (n_1 \times n_d)} \), we decompose it as
\[ A(i_1, \ldots, i_d, j_1, \ldots, j_d) = \sum_{\beta_1, \ldots, \beta_d = 1}^{R_1, \ldots, R_d} A^{(1)}_{\beta_1}(i_1, j_1) A^{(2)}_{\beta_1, \beta_2}(i_2, j_2) \cdots A^{(d)}_{\beta_{d-1}}(i_d, j_d), \]  
which is consistent with the Kronecker product \( A = A^{(1)} \otimes A^{(2)} \) in the case \( d = 2 \) and \( R_1 = 1 \), and allows a natural multiplication with (24), returning the result in the same form.

In the course of computing the matrix-by-vector product \( f = Ay \), the TT ranks of \( A \) and \( y \) are multiplied, i.e. \( r_k(f) = R_k(A) r_k(y) \), \( k = 1, \ldots, d - 1 \). In many cases they are unnecessarily large for achieving the required accuracy. However, as soon as \( f \) is already written in the format, its SVD re-compression steps, e.g. (23), can be implemented without high-dimensional tensors appearing. Only QR and SVD decompositions of \( nr \times r \) matrices are involved. The total complexity of this procedure is \( O(dnr^3) \), which opens up the possibility of using any standard iterative method, such as MINRES, as soon as \( r \) remains moderate during the course of the iterations.

The multi-index concept (22) allows us to compress even “one-dimensional” matrices and vectors, which lack a method for separating variables at first glance. Let us consider \( y = [y(i)] \in \mathbb{R}^n \), with \( n = 2^l \). Then we may write \( i \) in the binary coding,
\[ i = i_1 \ldots i_l = i_1 + 2(i_2 - 1) + \cdots + 2^{l-1}(i_l - 1), \quad i_s \in \{1, 2\}, \quad s = 1, \ldots, l. \]
As a result, a vector \( y \) is reshaped to an \( l \)-dimensional tensor \( y \), i.e. \( y(i_1, \ldots, i_l) = y(i) \), and the TT approximation can be applied to \( y \). The resulting TT format was called the Quantized TT (QTT) [43, 60, 61, 86]. If the TT ranks of \( y \) are moderate, the total storage reduces to a logarithmic amount \( O(lr^2) = O(\log n) \).

For many elementary functions and operators, their TT/QTT formats can be written analytically, for example, the discretized Laplace operator [42], the sine, exponential and polynomial functions, sampled on uniform grids in one [43, 63] and many dimensions [20, 44].

The discretisation of fractional order derivatives on a uniform grid is given as a convolution with a vector, which admits a low-rank QTT representation [75]. The corresponding Toeplitz matrix will also have a low QTT rank as shown in [41]. Therefore, the tensor product format can be applied to solve the problem (14).

4.2. Tensor product Krylov methods

For expository purposes we start the discussion by considering the case with one spatial and one temporal dimension, which leads to a matrix valued problem. Following this we discuss the more general tensor-valued equation and the corresponding solvers.
Matrix case

It was recently noted that the vectors $y, u, p \in \mathbb{R}^{n \times t}$ for space-time saddle point problems can be written as

$$y = \text{vec}(Y) = \text{vec}([y_1, \ldots, y_{nt}]),$$
$$u = \text{vec}(U) = \text{vec}([u_1, \ldots, u_{nt}]),$$
$$p = \text{vec}(P) = \text{vec}([p_1, \ldots, p_{nt}]).$$

Note that we can now perform any iterative scheme in matrix form for the unknowns rather than in vector form. This can best be seen by exploiting the identity

$$(-C_{\alpha} \otimes I_n + I_{nt} \otimes L) \text{vec}(Y) = \text{vec}(-I_n Y C_{\alpha}^T + L Y I_{nt}^T),$$

and then neglecting the vec operator on the right-hand-side. Additionally one can now use a low-rank decomposition, for instance via the singular value decomposition (SVD), such that $Y = Y_1 Y_2^T$, $U = U_1 U_2^T$, and $P = P_1 P_2^T$ are the solutions obtained using a low-rank Krylov method. The main ingredients of such a scheme are that the right-hand side and the initial guess are decomposed into low-rank form, which is then maintained throughout the iteration. In more detail, assume that the initial residual is given by

$$R_Y = W_Y V_Y^T \quad \text{with} \quad W_Y \in \mathbb{R}^{n \times r_1}, \quad V_Y \in \mathbb{R}^{n_t \times r_1},$$
$$R_U = W_U V_U^T \quad \text{with} \quad W_U \in \mathbb{R}^{n \times r_2}, \quad V_U \in \mathbb{R}^{n_t \times r_2},$$
$$R_P = W_P V_P^T \quad \text{with} \quad W_P \in \mathbb{R}^{n \times r_3}, \quad V_P \in \mathbb{R}^{n_t \times r_3},$$

where $r_i \ll n$. If the residual is not initially of this form, then we assume that it can be well approximated in this way. Every Krylov scheme now performs matrix-vector multiplications of $A$ and the initial residual given in (26). In [83] it was shown that the matrix-vector product with the matrix (14) can maintain the low-rank structure, only requiring an additional truncation step that can be performed cheaply compared to the cost of the full method. The construction of such a truncation function $T_\varepsilon$ for the matrix case is further described in [17] [83].

Tensor case

The previously discussed matrix setup is of course a special case of the more general tensor problem. As we noted, algebraic operations (matrix, scalar products and additions) and the SVD re-compression procedure in the TT format allow to rewrite any classical iterative method, keeping all vectors in the tensor format and performing only structured operations [64] [4] [21] [47]. Let us denote the compression (or truncation) procedure from a vector $y$ to a vector $\tilde{y} \approx y$ as

$$\tilde{y} = T_\varepsilon(y),$$

where by $\varepsilon$ we denote the relative truncation accuracy in the Frobenius norm. In particular, the TT-MINRES algorithm can be written as shown in Algorithm [4].

Again the convergence behavior of the MINRES algorithm depends on the system parameters and, in order to achieve robust convergence, we need to construct a suitable preconditioner.
3, we require an efficient tensor product solver, which is discussed next.

Of both $C$ an extended Krylov subspace that involves two sequences with each of the system matrices $S$ylvester equations \cite{81}. This method approximates the solution of the Sylvester equation in this method, first developed in \cite{25, 80} for Lyapunov equations and later adapted for the case of Sylvester equations \cite{81}. The ADI scheme often gives very good approximations but requires a set of shift parameters to guarantee convergence, which can be difficult to obtain. Hackbusch and Grasedyck suggest the use of a multigrid scheme \cite{32} that maintains the low-rank nature of the solution throughout the iteration. More reliable is the Krylov-plus-inverted-Krylov (KPIK) method, first developed in \cite{25, 80} for Lyapunov equations and later adapted for the case of Sylvester equations \cite{81}. This method approximates the solution of the Sylvester equation in an extended Krylov subspace that involves two sequences with each of the system matrices $C_α$ and $\frac{1}{\sqrt{α}}I_n - \tilde{L}$ in $\tilde{A}$. Additionally, the sequence requires the inverse or approximate inverse of both $C_α$ and $\frac{1}{\sqrt{α}}I_n - L$ from \cite{8} and \cite{9}. Since they are Toeplitz matrices, we can employ fast iterative solvers using circulant preconditioners, following \cite{13}.

In the tensor case, for the matrix $\tilde{A}$, as well as for Schur complement approaches in Section 3, we require an efficient tensor product solver, which is discussed next.

\begin{algorithm}
\begin{algorithmic}
\Require Right-hand side $f$, initial vector $y$ in the TT format, matrix $\mathcal{A}$ as a $\text{MatVec}$ procedure $g = \mathcal{T}_\varepsilon(\mathcal{A}y)$, preconditioner $\mathcal{P}$ (possibly identity), accuracy $\varepsilon$.
\Ensure Improved solution $y$.
\STATE 1: Start: compute $v_2 = \mathcal{T}_\varepsilon(f - \mathcal{A}y), z_1 = \mathcal{T}_\varepsilon(\mathcal{P}v_2), \gamma_1 = \sqrt{(z_1, v_2)}$.
\STATE 2: Initialize $c_1 = c_2 = \gamma_0 = 1, s_1 = s_2 = 0, \eta = \gamma_1, v_1 = w_1 = w_2 = [0, \ldots , 0]^\top$.
\STATE 3: Iterations:
\FOR {$j = 1, 2, \ldots, m$} \STATE 4: \STATE 5: $z_1 = z_1/\gamma_1, v_3 = \mathcal{T}_\varepsilon(\mathcal{A}z_1)$.
\STATE 6: $\delta = (z_1, v_3), v_3 = \mathcal{T}_\varepsilon(\mathcal{P}v_3, \gamma_2 = \sqrt{(z_2, v_3)}$. \{Orthogonalize the Krylov vector\}
\STATE 7: $z_2 = \mathcal{T}_\varepsilon(\mathcal{P}v_3), \gamma_2 = \sqrt{(z_2, v_3)}$.
\STATE 8: $\alpha_0 = c_1 \delta - c_0 s_1 \gamma_1, \alpha_1 = \sqrt{\alpha_0^2 + \gamma_2^2}, \alpha_2 = s_1 \delta + c_0 c_1 \gamma_1, \alpha_3 = s_0 \gamma_1$.
\STATE 9: $c_0 = c_1, c_1 = \alpha_0/\alpha_1, s_0 = s_1, s_1 = \gamma_2/\alpha_1$.
\STATE 10: $w_3 = \frac{1}{\alpha_1} \mathcal{T}_\varepsilon(z_1 - \alpha_3 w_1 - \alpha_2 w_2)$. \{Orthogonalize the preconditioned vector\}
\STATE 11: $y = \mathcal{T}_\varepsilon(y + c_1 \eta w_3)$. \{Correct the solution\}
\STATE 12: $\eta = -s_1 \eta, \gamma_0 = \gamma_1, \gamma_1 = \gamma_2, v_1 = v_2, v_2 = v_3, z_1 = z_2, v_1 = w_1, w_2 = w_3$.
\STATE 13: \textbf{if} $|\eta| < \varepsilon \sqrt{(b, \mathcal{P}b)}$ \textbf{then} Stop.
\ENDFOR
\end{algorithmic}
\end{algorithm}
4.3. Alternating solvers

It is often observed that Krylov vectors may require much larger TT ranks than the solution of the problem, see e.g., [24]. Alternating iterative tensor algorithms avoid this problem by seeking directly the elements of the tensor format for the solution. The Alternating Least Squares method (see [46] and references there) was developed to fit given data to a low-rank model. The Density Matrix Renormalization Group (DMRG) algorithm [89] was initially proposed for solution of ground state eigenvalue problems in quantum physics, and then extended to linear systems [23, 37, 39].

Given a TT format [24] for the approximate solution \( y \), we may collect its first, resp. last, \( k \) TT factors into the interface matrices,

\[
Y^{(1:k)}(i_1 \ldots i_k, \alpha_k) = \sum_{\alpha_1, \ldots, \alpha_{k-1}=1}^{r_1, \ldots, r_{k-1}} y^{(1)}(i_1) y^{(2)}(i_2) \cdots y^{(k)}(i_k), \quad Y^{(1:k)} \in \mathbb{R}^{n_1 \cdots n_k \times r_k},
\]

\[
Y^{(k:d)}(\alpha_{k-1} i_k \ldots i_d) = \sum_{\alpha_{k-1}, \ldots, \alpha_{d-1}=1}^{r_{k-1}, \ldots, r_d-1} y^{(k)}(\alpha_k) \cdots y^{(d)}(i_d), \quad Y^{(k:d)} \in \mathbb{R}^{r_{k-1} \times n_{k-1} \cdots n_d}.
\]

Then we assemble the frame matrix,

\[
Y_{\neq k} = Y^{(1:k-1)} \otimes I_{n_k} \otimes (Y^{(k+1:d)})^\top \in \mathbb{R}^{n_1 \cdots n_d \times r_{k-1} n_k r_k},
\]

which includes all TT factors but \( y^{(k)} \), hence the notation. Note that the column size of the frame matrix is exactly equal to the number of elements in the TT block \( y^{(k)} \). The TT format [24] can be seen as a linear map, induced by the frame matrix, i.e., \( y = Y_{\neq k} y^{(k)} \), where we denote by \( y^{(k)} \) the vector of elements of the \( k \)-th TT block, \( y^{(k)}(\alpha_{k-1} i_k \alpha_k) = y^{(k)}(\alpha_{k-1}, \alpha_k)(i_k) \).

Now let us consider \( Ay = f \) as an overdetermined linear linear system on \( y^{(k)} \), provided the TT format [24] is inserted instead of \( y \), i.e., \( AY_{\neq k} y^{(k)} = f \). A simple way to solve this system is to project it onto the frame matrix. We solve

\[
A_k y^{(k)} = f_k, \quad A_k = Y_{\neq k}^\top A Y_{\neq k}, \quad f_k = Y_{\neq k}^\top f.
\]

Using the orthogonalizations of the TT blocks via the QR decompositions of their matrix reshapings, we can always make the frame matrix orthogonal [79], that is \( Y_{\neq k}^\top Y_{\neq k} = I \). This ensures the stability of the problem, \( \text{cond}(A_k) \leq \text{cond}(A) \) if \( A + A^\top > 0 \). Iterating for all dimensions \( k = 1, \ldots, d \) (hence the name “alternating”), we obtain the simple one-block DMRG, or Alternating Linear Scheme (ALS) [37] algorithm.

However, in this scheme all frame matrices, and hence all TT blocks, have fixed sizes, prescribed by the TT ranks of the initial guess. This is very inconvenient: in most cases it is difficult to predict TT ranks of the solution for a given accuracy, and we would like to determine them adaptively. Another problem is the local convergence of the ALS method [76]: the result is highly dependent on the initial guess and may give an unsatisfactory approximation.

There are two principal ways to solve the first (technical) problem of the TT rank adaptivity during the iterative process. Note that the main issue is how to increase the ranks; to decrease them, it is sufficient to perform the SVD re-compression procedure, as pointed out in Section 4.1. One way to increase the ranks is given by the DMRG algorithm in its initial,
two-block version \[29\]. Instead of one block \(y^{(k)}\), we update both \(y^{(k)}\) and \(y^{(k+1)}\) at each step and adapt the TT rank \(r_k\). However, for linear systems with non-symmetric matrices, even the two-block DMRG method may converge to an incorrect solution \[24\]. A better way to increase TT ranks is to enrich TT blocks explicitly. On step \(1 \leq k \leq d\), after the solution of \(\text{[29]}\), we may expand

\[
y^{(k)}(i_k) := \begin{bmatrix} y^{(k)}(i_k) & z^{(k)}_k(i_k) \end{bmatrix}, \quad y^{(k+1)}(i_{k+1}) := \begin{bmatrix} y^{(k+1)}(i_{k+1}) & 0 \end{bmatrix},
\]

where \(z^{(k)}_k \in \mathbb{R}^{r_{k-1} \times n_k \times \rho_k}\) is some auxiliary tensor, and the zero-block in \(y^{(k+1)}\) has the corresponding sizes \(\rho_k \times n_{k+1} \times r_{k+1}\). This step does not impact the whole solution \(y\). However, when we proceed to the next block \(y^{(k+1)}\), the interface \(Y^{(1:k)}\) and the frame \(Y_{\neq k+1}\) matrices carry \(z^{(k)}_k\), so the Galerkin reduction \(\text{[29]}\) in the step \(k + 1\) is performed to a wider basis than in the ALS method. This can not only increase the TT rank, but also facilitate the convergence, if we select the augmentation \(z^{(k)}_k\) properly.

The idea of the AMEn algorithm proposed in \[24\] is to combine the alternating iteration with the steepest descent method. The latter updates the solution by adding the scaled residual, \(y := y + zh\), where \(z \approx f - Ay\) and \(h\) is a scalar weight. If \(y\) and \(z\) are defined by their TT formats, the summation is computed as

\[
y(i_1 \ldots i_d) + zh(i_1 \ldots i_d) = \begin{bmatrix} y^{(1)}(i_1) & z^{(1)}(i_1) \end{bmatrix} \begin{bmatrix} y^{(2)}(i_2) \ z^{(2)}(i_2) \ \vdots \ y^{(d)}(i_d) \ z^{(d)}(i_d) \end{bmatrix}.
\]

Note that the first factor here has the same form as the enrichment of \(y^{(k)}\) in \(\text{[30]}\); therefore, \(\text{[30]}\) performs the first step of the TT format addition. This is sufficient, since in the next step we solve \(\text{[29]}\) and recover correct \(z\)-related entries in \(y^{(k+1)}\) that minimize the energy function if \(A = A^T > 0\). Due to this property, the new method was called AMEn (alternating minimal energy).

To compute \(z^{(k)}_k\) in practice, it is sufficient to provide a very rough approximation of the residual. We prepare some initial guess for \(z\) in the TT format with ranks \(\rho_1, \ldots, \rho_{d-1}\) and update it towards \(f - Ay\) by the secondary ALS iteration, minimizing the distance \(||z - (f - Ay)||_2||\). The entire procedure is summarized in Algorithm 2. Notice that after the enrichment \(\text{[30]}\) we need to orthogonalize \(y^{(k)}\) explicitly in order to make the frame matrices orthogonal.

In most cases, the enrichment ranks \(\rho_k \lesssim \rho \lesssim 10\) are sufficient. To make the computational complexity in Lines 4,5,7 independent of \(d\), we reuse some intermediate data during the subsequent iteration \(k = 1, \ldots, d\) (see \[22\], \[23\], \[24\] for details).

At this juncture we wish to remark on the utility of the approach presented when nonlinear problems, such as the previously mentioned phase-field equations, are considered. Our crucial contribution in this work is the introduction of low-rank solvers for the saddle point problems that arise for the linear problems. For nonlinear problems one typically applies an outer solver such as a Picard-type or Newton-type method \[55\]. At the heart of these methods, and indeed almost all nonlinear solution strategies, lies the solution of a large-scale system in saddle point form for which the low-rank strategies we have described can be applied. In future work we wish to carry out a thorough analysis of this setting: from our experience with low-rank methods for classical derivative and nonlinear equations the structure of the saddle point system \[14\] will be similar, but the particular blocks, e.g. the FDE block \(A\), can consist of a sum of tensors due it being the linearization of a nonlinear low-rank operator.
**Require:** Matrix $A$, right-hand side $f$, initial guesses $y$ and $z$ in the TT formats.

**Ensure:** Improved solution $y$ in the TT format.

1: while not converged and iteration limit is not hit do
2: Orthogonalize $y$ and $z$ s.t. $Y^{(k:d)}$ and $Z^{(k:d)}$ are orthogonal for $k = 2, \ldots, d$.
3: for $k = 1, \ldots, d$ do
4: Build and solve (29) for the TT block $y^{(k)}$.
5: Update the residual block $z^{(k)} = \left( Z^{(1:k−1)} \otimes I_{nk} \otimes (Z^{(k+1:d)})^\top \right)^\top (f − Ay)$.
6: if $k < d$ then
7: Compute the enrichment block $z^{(k)}_k = \left( Y^{(1:k−1)} \otimes I_{nk} \otimes (Z^{(k+1:d)})^\top \right)^\top (f − Ay)$.
8: Perform the expansion (30).
9: Orthogonalize $y^{(k)}$ and $z^{(k)}$ s.t. $Y^{(1:k)}$ and $Z^{(1:k)}$ are orthogonal.
10: end if
11: end for
12: end while

**Algorithm 2:** AMEn method

5. Numerical results

We solve the problem (14) in the QTT format for different inputs and parameters, and compare three procedures:

“SC1” the First Schur Complement scheme (18), where the AMEn Algorithm 2 is applied to solve linear systems;

“SC2” the Second Schur Complement scheme (20), again with the AMEn solver; and

“MR” the MINRES Algorithm 1 with the preconditioner (16).

In the MR approach, the linear systems with the matrices $A$ and $\tilde{A}^\top$ are also solved using the AMEn method. The three components $y$, $u$ and $p$ and the blocks in the matrix (14) are kept in separate TT formats. Since the matrix $A$ has considerable TT ranks (up to 20), it is not efficient to multiply it by vectors exactly and then perform the SVD truncation. Instead, we approximate the matrix-by-vector product iteratively using a special variant of Algorithm 2, where $A = I$ and $f = \tilde{A}y$. Further implementation details are given in the Appendix.

As a widely used error indicator, we consider the relative discrepancy of the state vector $y$ in the Frobenius norm. Given some reference vector $y_*$, we denote

$$E(y_*) = \frac{\|y - y_*\|^2}{\|y_*\|^2}. \quad (31)$$

The examples were implemented using the TT-Toolbox package and conducted on one core of the MPI otto cluster, an Intel Xeon X5650 CPU at 2.67GHz, in Matlab R2012a.

5.1. Complete data, two-dimensional space

In the first test both observation and control are defined on the whole domain $[0,1]^2$. Therefore, $M_1$ and $M_3$ are identity matrices, and both Schur complements are well defined. We investigate their performance and compare with the TT-MINRES approach.

\[ \text{The source codes are available at } \url{http://www.mpi-magdeburg.mpg.de/2866211/supplementaries.zip} \]

\[ \text{http://github.org/oseledets/TT-Toolbox version Nov 10, 2014} \]
Table 1: Complete data test. Left: CPU times (sec.), maximal TT ranks and iterations vs. grid level $l$. Right: discrepancies between solutions computed by SC2, MINRES, Full and SC1 schemes.

<table>
<thead>
<tr>
<th>$l$</th>
<th>SC1</th>
<th>SC2</th>
<th>MR</th>
<th>Full</th>
<th>SC2</th>
<th>MR</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>rank</td>
<td>time</td>
<td>rank</td>
<td>iter</td>
<td>time</td>
<td>$\mathcal{E}(\bar{y}_{SC1})$</td>
</tr>
<tr>
<td>3</td>
<td>0.596</td>
<td>10</td>
<td>1.396</td>
<td>9</td>
<td>2.980</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>0.489</td>
<td>12</td>
<td>1.532</td>
<td>14</td>
<td>4.823</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>1.021</td>
<td>18</td>
<td>4.103</td>
<td>21</td>
<td>11.45</td>
<td>26</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>2.280</td>
<td>22</td>
<td>11.69</td>
<td>27</td>
<td>24.70</td>
<td>36</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>5.061</td>
<td>25</td>
<td>27.61</td>
<td>36</td>
<td>83.99</td>
<td>49</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>10.46</td>
<td>28</td>
<td>55.45</td>
<td>41</td>
<td>245.7</td>
<td>62</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>18.90</td>
<td>29</td>
<td>99.94</td>
<td>45</td>
<td>984.9</td>
<td>78</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>28.03</td>
<td>29</td>
<td>175.1</td>
<td>51</td>
<td>3494</td>
<td>100</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>47.49</td>
<td>29</td>
<td>257.8</td>
<td>55</td>
<td>9905</td>
<td>255</td>
<td>25</td>
</tr>
</tbody>
</table>

The constraint problem is the equation (1b) in 2 spatial variables plus time, with $\alpha = 0.5$, $\beta_1 = 1.5$, $\beta_2 = 1.5$ and $f = 0$. The level-$l$ discretization grid contains $2^l \times 2^l \times 2^{2l}$ points, lying uniformly in space and time, i.e.,

$$x_1(i) = ih, \quad x_2(j) = jh, \quad t(k) = k\tau,$$

$$i,j = 1,\ldots,2^l, \quad k = 1,\ldots,2^{2l}, \quad h = \frac{1}{2^l + 1}, \quad \tau = h^2.$$  

The target function is

$$\bar{g}(x_1, x_2, t) = 10 \cos(10x_1) \sin(x_1 x_2).$$

Boundary and initial conditions are taken to be homogeneous. We vary the grid level $l$, the regularization parameter $\gamma$ and the tensor approximation tolerance $\varepsilon$. We use the same threshold $\varepsilon$ both for the tensor approximation and also as the stopping tolerance for our numerical schemes.

**Performance with respect to the grid size**

We fix $\gamma = 10^{-6}$, $\varepsilon = 10^{-6}$, and vary the grid size in the range $l = 3, \ldots, 11$. The results are shown in Table 1. We can see that the iteration numbers for MINRES are quite robust with respect to the varying mesh-sizes. Nevertheless, the rank increase with each refinement slows the method down significantly. The SC2 approach also suffers from a rank increase but less significantly than the MINRES. As this is the most benign case with full observation and control domain, the SC1 method performs outstandingly with almost no increase in the ranks for the final refinements. We further compare the solutions of the three different approaches and can see that the MINRES and SC1 approach show the best coincidence.

To demonstrate the importance of the QTT compression, we also compare our approaches with the “Full” scheme, where (14) is solved via classical MINRES with the full storage of all vectors. Toeplitz matrices were multiplied by vectors via the FFT, and the linear systems with $\tilde{A}$ and $\tilde{A}^\top$ from (16) were solved using the BICGSTAB method. The solution accuracy is good, but the CPU time grows dramatically and makes the calculations prohibitively expensive for $l > 6$.

**Performance with respect to the regularization parameter (Table 2)**

In the second experiment we examine the behavior of the methods with respect to the regularization parameter $\gamma$. For an application the choice of regularization parameter is often
Table 2: Complete data test. Left: CPU times (sec.) and discrepancies with the observation data. Right: discrepancies with SC1.

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>$\text{SC1}$</th>
<th>$\text{SC2}$</th>
<th>$\text{MR}$</th>
<th>$\text{SC2}$</th>
<th>$\text{MR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mathcal{E}(\bar{y})$</td>
<td>$\mathcal{E}(\bar{y})$</td>
<td>$\mathcal{E}(\bar{y})$</td>
<td>$\mathcal{E}(y_{SC1})$</td>
<td>$\mathcal{E}(y_{SC1})$</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>23.91, 9.33e-1</td>
<td>31.56, 9.33e-1</td>
<td>4884, 9.33e-1</td>
<td>4.53e-5, 4.12e-5</td>
<td></td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>14.47, 3.31e-1</td>
<td>26.97, 3.31e-1</td>
<td>707.0, 3.31e-1</td>
<td>1.39e-4, 6.94e-6</td>
<td></td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>5.226, 9.53e-2</td>
<td>27.72, 9.53e-2</td>
<td>8838, 9.53e-2</td>
<td>1.76e-4, 9.51e-7</td>
<td></td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>1.354, 5.27e-3</td>
<td>29.51, 5.29e-3</td>
<td>17.18, 5.27e-3</td>
<td>4.23e-4, 1.18e-6</td>
<td></td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>1.217, 5.75e-5</td>
<td>31.26, 4.28e-4</td>
<td>5.531, 5.75e-5</td>
<td>4.24e-4, 9.91e-7</td>
<td></td>
</tr>
<tr>
<td>$10^{-12}$</td>
<td>1.189, 1.57e-6</td>
<td>29.43, 3.78e-4</td>
<td>2.450, 5.75e-7</td>
<td>3.78e-4, 1.48e-6</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Complete data test. Left: CPU times (sec.) vs. tensor approximation tolerance $\varepsilon$. Right: discrepancies of SC2 and Minres with SC1.

<table>
<thead>
<tr>
<th>$\varepsilon$</th>
<th>$\text{SC1}$</th>
<th>$\text{SC2}$</th>
<th>$\text{MR}$</th>
<th>$|y_{SC2} - y_{SC1}|$</th>
<th>$|y_{MR} - y_{SC1}|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>0.728, 1.471</td>
<td>3.400</td>
<td>2.50e-1, 7.06e-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>1.449, 4.899</td>
<td>9.869</td>
<td>1.38e-2, 1.01e-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>4.962, 27.53</td>
<td>75.07</td>
<td>1.99e-4, 9.11e-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>17.26, 163.6</td>
<td>504.1</td>
<td>1.18e-6, 8.57e-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>37.34, 880.2</td>
<td>2846</td>
<td>3.76e-9, 6.27e-10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

determined by techniques such as $L$-curve analysis [33] and discrepancy principle [27]. As we do not focus on a particular application but rather a methodological development we want to illustrate the robustness of our method regarding for a wide range of regularization parameters. In our experiments we use the grid level is $l = 7$, and $\varepsilon = 10^{-6}$. According to Table 2 we can see that the computed state approaches the desired state further when $\gamma$ gets smaller. Due to the Dirichlet boundary conditions, we neither observe nor control the boundary, and consider only the inner grid points in the discrete model. The error indicator $\mathcal{E}(\bar{y})$ involves only the inner points as well. Therefore, it is possible to approach $\bar{y}$ when $\gamma \to 0$. As a by-product, the CPU times of all methods decrease with $\gamma$. Since the matrices $M_1, M_2$ and $M_3$ are scaled identities, the Schur complement (17) becomes $S = \frac{1}{\tau h^2} AA^\top + \frac{\tau h^2}{\gamma} I$, which tends to an identity when $\gamma \to 0$, and becomes easier to invert. This is clearly reflected by the SC1 and MR methods. However, the SC2 is slower: its bottleneck is the inversion of $A$, which is performed independently of $\gamma$.

Performance with respect to the accuracy threshold (Table 3) Finally, we fix $l = 7$, $\gamma = 10^{-6}$, and vary $\varepsilon$ from $10^{-1}$ to $10^{-10}$; we show the results in Table 3. It is not surprising that all methods require more computing time to obtain the desired accuracy. The discrepancies demonstrate almost perfect linear dependence on $\varepsilon$.

5.2. Incomplete data, two-dimensional space

We now investigate the following more challenging scenario, reducing the observation and control domains to $\Omega_{\bar{y}} = \Omega_u = \left[\frac{3}{8}, \frac{5}{8}\right]^2$. Consequently, the first Schur Complement scheme is not applicable, and we test the second approach. Other parameters are the same as in the previous example.
Table 4: Incomplete data test. Left: CPU times (sec.), maximal TT ranks and iterations vs. grid level \( l \). Right: solution errors.

<table>
<thead>
<tr>
<th>( l )</th>
<th>SC2 time</th>
<th>SC2 rank</th>
<th>MR time</th>
<th>MR rank</th>
<th>Iter</th>
<th>Full time</th>
<th>Full rank</th>
<th>SC2 ( \mathcal{E}(y_\star) )</th>
<th>MR ( \mathcal{E}(\bar{y}) )</th>
<th>SC2 ( \mathcal{E}(y_\star) )</th>
<th>MR ( \mathcal{E}(y_\star) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.081</td>
<td>23</td>
<td>5.247</td>
<td>44</td>
<td>11</td>
<td>5.521</td>
<td>44</td>
<td>4.25e-7</td>
<td>2.42e-3</td>
<td>2.27e-4</td>
<td>8.31e-9</td>
</tr>
<tr>
<td>4</td>
<td>2.426</td>
<td>26</td>
<td>27.88</td>
<td>138</td>
<td>15</td>
<td>124.1</td>
<td>44</td>
<td>1.69e-6</td>
<td>9.12e-3</td>
<td>1.49e-4</td>
<td>8.93e-9</td>
</tr>
<tr>
<td>5</td>
<td>7.039</td>
<td>42</td>
<td>1018</td>
<td>752</td>
<td>17</td>
<td>3542</td>
<td>44</td>
<td>3.34e-6</td>
<td>1.82e-2</td>
<td>3.68e-4</td>
<td>3.53e-5</td>
</tr>
<tr>
<td>6</td>
<td>20.96</td>
<td>56</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>54.19</td>
<td>64</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>120.0</td>
<td>72</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td>223.7</td>
<td>75</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>414.4</td>
<td>76</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>11</td>
<td>743.7</td>
<td>81</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Performance with respect to the grid size (Table 4)

The first parameter that we vary is the grid level \( l \). The regularization parameter is \( \gamma = 10^{-6} \), the approximation tolerance \( \varepsilon = 10^{-6} \), and the fractional orders are \( \alpha = 0.5 \), \( \beta_1 = \beta_2 = 1.5 \). The results are shown in Table 4.

In this example, the number of Minres iterations is moderate, but the TT ranks grow quickly, making the computations prohibitively expensive for grids larger than \( l = 5 \).

To estimate the error in the solution, we compute the reference solution \( y_\star \) using the second Schur Complement scheme with a smaller threshold \( \varepsilon_\star = 10^{-8} \). As an important indicator for the incomplete observation and control, we also demonstrate the deviation of the solution \( y \) from the prescribed data \( \bar{y} \) on the observation domain.

A drawback of the Schur Complement scheme which we observe is the growth of the error, by a factor 3 on average from level to level. This is because the condition number of the system matrix grows like \( (2l)^\beta = (2^{1.5})l \approx 3^l \), and so does the error when we compute the approximate inverse matrix in (21). To verify that the error comes from approximation of \( Q \), not from other steps of the scheme, we solve (21) with a higher accuracy (say, \( 10^{-8} \)), while all other steps in (20) are performed with the same \( \varepsilon = 10^{-6} \). For \( l = 9 \), it gives \( \|ySC2-y_\star\| / \|y_\star\| \approx 1.58 \times 10^{-5} \) in about 11 minutes.

Performance with respect to the regularization parameter (Table 5)

When the solution is controlled on a part of the domain, the influence of the regularization parameter \( \gamma \) becomes more interesting numerically. As previously, we fix the grid level to \( l = 7 \) and set \( \varepsilon = 10^{-6} \). The reference solution is computed with the threshold \( \varepsilon_\star = 10^{-8} \). We omit the experiment with Minres, since it would require very large TT ranks.

In this test, the part \( M_1 A^{-1} M_3 (\gamma M_2)^{-1} M_3^\top \) of the Schur complement matrix in (20) is indefinite. For small \( \gamma \) the whole Schur complement may become indefinite, which makes the calculations difficult. Therefore, we are limited by the range of \( \gamma \) we can experiment with. However, the scheme is quite reliable for \( \gamma \geq 10^{-7} \). The solution error is almost stable, and the complexity grows moderately. The discrepancy is proportional to \( \sqrt{\gamma} \), as expected.

Performance with respect to \( \alpha \) and \( \beta \) (Fig. 7)

Since the order of a fractional derivative is a continuous quantity, it is interesting to test the algorithms for a range of orders. In particular, in the first test we vary \( \alpha \in [0.1, 1] \) and
Table 5: Incomplete data test. CPU times (sec.) and accuracy of the SC2 method vs. regularization parameter $\gamma$

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>time</th>
<th>rank</th>
<th>$\mathcal{E}(y)$</th>
<th>$\mathcal{E}(\tilde{y})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-1}$</td>
<td>37.66</td>
<td>44</td>
<td>1.65e-6</td>
<td>9.95e-1</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>41.62</td>
<td>47</td>
<td>1.01e-5</td>
<td>9.61e-1</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>47.02</td>
<td>52</td>
<td>3.67e-5</td>
<td>9.66e-3</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>54.65</td>
<td>64</td>
<td>4.40e-5</td>
<td>3.62e-2</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>74.23</td>
<td>68</td>
<td>6.42e-5</td>
<td>9.66e-3</td>
</tr>
</tbody>
</table>

For $\beta_1 = \beta_2 \in [1.1, 2]$. We show two-dimensional plots of computation times, TT ranks and errors in Fig. [1].

The TT ranks and CPU times grow towards lines $\alpha = 1$ and $\beta = 1$. To explain this, recall that the matrix $C_\alpha$ approaches identity for $\alpha \to 0$ and turns into the standard first-order difference at $\alpha = 1$, and the condition number $\text{cond}(C_\alpha)$ grows monotonously for $\alpha \in (0, 1]$. However, the Riemann-Liouville matrix $L_\beta$ behaves differently: at $\beta = 1$, the matrix is equal to the scaled second-order difference, $L_1 = \frac{h}{2}L_2$. When $\beta > 1$, the matrix $L_\beta$ depends continuously on $\beta$, so the condition number should have an extremum on $[1, 2]$, which is apparently a minimum near $\beta = 1.2$. Large conditioning makes the problem harder to solve when $\beta$ approaches 1. Even more difficult are the cases $\beta < 1$ and $\alpha > 1$: the matrices $C_\alpha$ and $L_\beta$ become indefinite, and the tensor product solver struggles.

Nevertheless, for parabolic cases $\alpha \in (0, 1]$ and $\beta \in (1, 2]$, our scheme is fairly robust. The bottom left plot in Fig. [1] shows that the relative error sits at the level $10^{-4} - 10^{-5}$ for the entire range of orders of differentiation. The deviation from the target vector (Fig. [1], bottom right) is also reasonable: the closer the orders are to zero, the closer the operators are to identities, and hence they impose a less severe constraint to the optimization of the distance $\|y - \tilde{y}\|$. 

**Performance with respect to $\beta_1$ and $\beta_2$ (Fig. [2])**

We now fix $\alpha = 0.5$, and vary $\beta_1, \beta_2$ within the range $[1.1, 2]$. The layout of Fig. [2] is the same as in Fig. [1] only the CPU times are now presented without the logarithmic scale, since they do not vary as strongly. Again, both CPU times and TT ranks increase towards 1. As opposed to the picture w.r.t. $\alpha$ and $\beta$, there is an anisotropy w.r.t. $\beta_1$ and $\beta_2$, since the target function $\tilde{y}(x_1, x_2)$ is not symmetric w.r.t. $x_1$ and $x_2$. Both error indicators behave similarly to those in the previous figure.

**Experiment with variable coefficients (Table 6)**

A more general FDE involves variable coefficients. In particular, a two-dimensional example from [87] reads

$$
\left( C_\alpha D_1^{\alpha} p_+ + \frac{RL}{0} D_{x_1}^{\beta_1} - \frac{RL}{0} D_{x_2}^{\beta_1} - q_+ - \frac{RL}{0} D_{x_2}^{\beta_2} - q_- \right) y(x_1, x_2, t) + u(x_1, x_2, t) = 0,
$$

where $p_+ = \Gamma(1.2)x_1^{\beta_1}, p_- = \Gamma(1.2)(2 - x_1)^{\beta_1}, q_+ = \Gamma(1.2)x_2^{\beta_2}, q_- = \Gamma(1.2)(2 - x_2)^{\beta_2}$ are the coefficient functions, and the orders are $\alpha = 0.5, \beta_1 = \beta_2 = 1.8$. Notice that the left and right Riemann-Liouville derivatives enter the equation with different coefficients. After the Grünwald-Letnikov discretization, the matrix $A$ reads

$$
A = C_\alpha \otimes I_{n_1 n_2} - I_{n_1} \otimes (P_+ T_{\beta_1} + P_- T_{\beta_1}^\top) \otimes I_{n_2} - I_{n_2} \otimes I_{n_1} \otimes (Q_+ T_{\beta_2} + Q_- T_{\beta_2}^\top),
$$
where $T_\beta$ is from [6], $P_+, P_-, Q_+, Q_-$ are diagonal matrices with the grid values of $p_+, p_-, q_+, q_-$, respectively. We also consider a time-dependent desired state

$$\bar{y}(x_1, x_2, t) = 10 \cos(10x_1) \sin(x_1x_2)(1 - \exp(-5t)).$$

All other parameters are the same as in the previous test.

The results are shown in Table 6. Since the coefficient admits a low-rank structure, the behavior of the methods is qualitatively the same as in the constant coefficient case. The CPU times are larger due to larger condition numbers of the matrices involved.

5.3. Three-dimensional problem, incomplete data

To verify the applicability of our technique to a significantly larger problem, we solve the problem (1c) in a three-dimensional space domain plus time. We set $\Omega = [0, 1]^3$, $\alpha = 0.5$, $\beta_1 = \beta_2 = \beta_3 = 1.5$, and $f = 0$. The target function is given by

$$\bar{y}(x_1, x_2, x_3, t) = e^{-64r^2}, \quad r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2.$$ 

Boundary and initial conditions are zeros. On discretization level $l$ we use the uniform $2^l \times 2^l \times 2^l \times 2^l$ grid. The observation and control domains are $\Omega_{\bar{y}} = \Omega_u = [\frac{3}{8}, \frac{5}{8}]^3$, which is about 1.5% of the whole volume. The regularization and stopping tolerances are set to $\gamma = \varepsilon = 10^{-6}$. 


In this example, the MINRES method would be prohibitive computationally due to the large TT ranks, so we investigate only the SC2 approach. In Table 7 we show the performance w.r.t. the grid size. Again, the reference solution $y_*$ is obtained by the SC2 method with accuracy $\varepsilon_* = 10^{-8}$. We observe that the behavior is qualitatively the same as for the 2D problem with incomplete data. We see that the TT ranks stabilize, and the CPU time tends to a linear growth with $l$, but the accuracy of the solution deteriorates proportionally to the condition number of the system matrix, because so does the accuracy of the matrix inversion in the Schur complement. Nevertheless, at moderate grids (e.g. 7) the results are satisfactory, considering the fact that the full problem of size $2^{35}$ with a dense matrix is intractable.

In Fig. 3 we show volumetric plots of the solution, control and Lagrange multiplier vectors at the final time $T = 1$, computed at the grid level $l = 5$. We see a good agreement with the target solution. An interesting feature is the anisotropic structure of the control. The 1.5-order fractional derivative possesses some properties of the convection first-order operator. In particular, a positive force is exerted on the left side of the center (the peak of the Gaussian function): the (inverse) fractional operator moves it to the right, towards the centered Gaussian function.

6. Conclusions and outlook

We have presented numerical algorithms for the optimization of an objective function subject to a fractional differential equation constraint. For this we discussed the discretization of
the differential equation via the well-known Grünwald-Letnikov finite difference method. Using a classical Lagrangian approach we obtained a saddle point system of vast dimensionality representing the first order optimality conditions. We showed that these systems have an inherent tensor product structure. For the efficient solution of these systems we discussed three possible approaches. Two of these methods are variations of a Schur complement approach, and the third is a preconditioned Krylov subspace solver. As the storage requirements are too large for all practically relevant scenarios when using a more fundamental approach, we then introduced algorithms working with a compressed storage format. Namely we utilized a tensor train method, which was also amendable for a tensorized Minres solver as well as alternating tensor solvers. We then illustrated the performance of these solvers on various setups and showed their competitiveness for vast system dimensions.

For future research in this field, we believe the study of nonlinear problems such as the phase-field equations described above will be of great value. Furthermore, the study of FDEs with uncertain parameters is a challenging and interesting topic. In this setting one can again separate the problem into low-rank objects, as the presented techniques can additionally achieve low-rank solutions even if the spatial dimension is not in tensor form, e.g. discretization via finite elements on possibly irregular domains. We aim to base our approaches for these applications on recent results presented in [9, 7].
Figure 3: 3D test. $y$ (top left), $\bar{y}$ (top right), $u$ (bottom left) and $p$ (bottom right) at the final time.

Appendix. Implementation of the TT solvers

The AMEn Algorithm \cite{2} for solution of linear systems is provided by the procedure amen_solve2.m in the TT-Toolbox, and for the fast approximation of products $Ay$ and $A^\top y$ within MINRES we used the procedure amen_mv.m.

Despite the progress of the tensor methods, they still require some parameter tuning for better performance. We used the Frobenius norm of the error between two consecutive iterations as an error measure. This is specified by passing the parameters ‘\texttt{trunc\_norm}’, ‘\texttt{fro}’
To address the ill conditioning of $S$ in (18), for the SC1 method we take $\text{local\_restart} = 100$, $\text{max\_full\_size} = 2500$ and $\text{tol\_exit} = 5\varepsilon$. The first two parameters increase the accuracy of the solver for (29), and the third parameter removes unnecessary iterations near a tolerance of $\varepsilon$ by stopping at $5\varepsilon$. Since the TT ranks of $A^{-1}$ are rather large, we set $\text{kickrank} = 10$ for (21). This allows to increase the ranks faster and have fewer iterations. The Schur complement $S$ in (20) is also ill-conditioned, and we set $\text{local\_restart} = 100$ for this stage. Finally, in the preconditioning stage of the MINRES method, we start the AMEn algorithm using the given right-hand side as the initial guess, as $\tilde{A}$ is reasonably close to the identity matrix.

References


