Max Planck Institute Magdeburg
Preprints

Sergey Dolgov, John W. Pearson, Dmitry V. Savostyanov,
Martin Stoll

Fast tensor product solvers for optimization problems with fractional differential equations as constraints

MPIMD/14-25 **January 9, 2015**

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.

Imprint:

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

While the study of derivatives of arbitrary order is a long-standing subject area [25], 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 [12] and viscoelasticity [38, 73]. The applications include electrical circuits [32, 60], electro-analytical chemistry [71], biomechanics [23], and image processing [76].

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 [42, 43, 58, 59, 60]. Adomian decomposition should be mentioned as a popular semi-analytical approach [1], although its use is limited. Recently, (discontinuous) finite element schemes for FDEs have also received considerable attention [13, 48, 75]. Since the fractional differential operators are, in fact, integral operators, the matrix of the corresponding linear system is usually 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, following the approach described in [51, 37] and recently applied to fractional calculus in [62, 10].

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 [30, 72] for introductions to the field). Optimal control problems for FDEs have previously been studied in literature such as [2, 3, 45, 46, 52, 61]. However, they were mostly considered 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.

Another approach that has recently been studied by Burrage *et al.* is to consider a matrix function approach to solve the discretized system (see [11] for details). Our work here is motivated by some recent results in [59] where the discretization via finite differences is considered in a purely algebraic framework. derivative, which in turn leads to a tensor structured equation.

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 [14, 25, 27, 44, 56, 57] 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 [56] 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 α with $n-1 < \alpha < n$, $n \in \mathbb{N}$, is defined as the following integral

$$ {}^{\mathrm{C}}_{a}D^{\alpha}_{t}\, f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(s)ds}{(t-s)^{\alpha-n+1}}, $$

assuming that it is convergent (see [15, 25, 42, 56, 65] for more details). Based on the discussion in [59], the Caputo derivative is frequently used for the derivative with respect to time. The left-sided Riemann-Liouville derivative of real order α with $n-1 < \alpha < n$, is defined by

$$ {}^{\mathrm{RL}}_{a}D^{\alpha}_{t}\, f(t) = \frac{1}{\Gamma(n-\alpha)} \left(\frac{d}{dt}\right)^n \int_a^t \frac{f(s)ds}{(t-s)^{\alpha-n+1}}, $$

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

$$ {}^{\mathrm{RL}}_{t}D^{\alpha}_{b}\, f(t) = \frac{(-1)^n}{\Gamma(n-\alpha)} \left(\frac{d}{dt}\right)^n \int_t^b \frac{f(s)ds}{(s-t)^{\alpha-n+1}}, $$

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

$$ {}^{\mathrm{R}}D^{\alpha}_{t}\, f(t) = \frac{1}{2}\left({}^{\mathrm{RL}}_{a}D^{\alpha}_{t}\, f(t) + {}^{\mathrm{RL}}_{t}D^{\alpha}_{b}\, f(t)\right). $$

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.

2.2 Model problems

In this section we introduce 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 which links the state to the control is given by the following FDE

$$\left({}_{0}^{\mathrm{C}}D_{t}^{\alpha} - {}^{\mathrm{R}}D_{x}^{\beta}\right) y(x,t) + u(x,t) = f(x,t), \tag{1a}$$

$$\left({}_{0}^{\mathrm{C}}D_{t}^{\alpha} - {}^{\mathrm{R}}D_{x_1}^{\beta_1} - {}^{\mathrm{R}}D_{x_2}^{\beta_2}\right) y(x_1,x_2,t) + u(x_1,x_2,t) = f(x_1,x_2,t), \tag{1b}$$

$$\left({}_{0}^{\mathrm{C}}D_{t}^{\alpha} - {}^{\mathrm{R}}D_{x_1}^{\beta_1} - {}^{\mathrm{R}}D_{x_2}^{\beta_2} - {}^{\mathrm{R}}D_{x_3}^{\beta_3}\right) y(x_1,x_2,x_3,t) + u(x_1,x_2,x_3,t) = f(x_1,x_2,x_3,t) \tag{1c}$$

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 $\operatorname{supp}(\bar{y}) = Q_{\bar{y}}$ and $\operatorname{supp}(u) = Q_u$ 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

$$J(y,u) := \frac{1}{2}\left\|y - \bar{y}\right\|_{L_2(Q_{\bar{y}})}^2 + \frac{\gamma}{2}\left\|u\right\|_{L_2(Q_u)}^2, \tag{2}$$

where γ 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 [9, 28] (details on the numerical treatment of FDEs can be found in [42, 43, 56, 58, 59]). We consider $\Omega = [0,1]^d$ and assume that Ω_u and $\Omega_{\bar{y}}$ are also cubes. Each function is discretized by collocation on a uniform tensor product grid in space and time

$$\begin{aligned} x_k(i_k) &= i_k h_k, \quad i_k = 1,\dots,n_k, \quad h_k = \frac{1}{n_k+1}, \quad k = 1,2,3, \\ t(m) &= m\tau, \quad m = 1,\dots,n_t, \quad \tau = \frac{T}{n_t}. \end{aligned} \tag{3}$$

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 conditions, $y(x_1,\dots,x_d,t) = 0$ if $x_k \leq 0$, $x_k \geq 1$ or $t \leq 0$. If the grid sizes are the same in all spatial directions, we denote $h = h_1 = h_2 = h_3$.

2.3 Grünwald-Letnikov formula

A classical method for the discretization of FDEs is based on the formula of Grünwald and Letnikov. In particular, for the spatial derivative we use its shifted version [42, 43]

$${}^{\mathrm{RL}}_{0}D_{x}^{\beta}\, y(x) \approx \frac{1}{h^{\beta}} \sum_{i=0}^{n} g_{\beta,i}\, y(x - (i-1)h) \tag{4}$$

The coefficients $g_{\beta,i}$ are defined by

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

and can be computed efficiently using the recurrent formula [56]: $g_{\beta,0} = 1$, $g_{\beta,i} = \left(1 - \frac{\beta+1}{i}\right) g_{\beta,i-1}$ 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

$$ {}^{\mathrm{RL}}_{0}D^{\beta}_{x}\, y \quad \to \quad h^{-\beta} \underbrace{\begin{bmatrix} g_{\beta,1} & g_{\beta,0} & 0 & \dots & \dots & \dots & 0 \\ g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & \ddots & & & \vdots \\ g_{\beta,3} & g_{\beta,2} & g_{\beta,1} & g_{\beta,0} & \ddots & & \vdots \\ \vdots & \ddots & g_{\beta,2} & g_{\beta,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & g_{\beta,0} & 0 \\ \vdots & \ddots & \ddots & \ddots & g_{\beta,2} & g_{\beta,1} & g_{\beta,0} \\ g_{\beta,n} & g_{\beta,n-1} & \dots & \dots & g_{\beta,3} & g_{\beta,2} & g_{\beta,1} \end{bmatrix}}_{T_\beta} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}}_{y} . \tag{5}$$

The matrix T_β is a Toeplitz matrix, which we discuss later in more detail. As in [59] we approximate the spatial derivative of order $1 \le \beta \le 2$ 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

$$L_\beta := \frac{1}{2}\left(T_\beta + T_\beta^\top\right).$$

The discretization in time is done analogously, using the Grünwald–Letnikov formula [56, 65]

$$ {}^{\mathrm{C}}_{0}D^{\alpha}_{t}\, y(t) \approx \frac{1}{\tau^\alpha} \sum_{m=0}^{n_t} g_{\alpha,m} y(t - m\tau), \tag{6}$$

and the Caputo derivative leads to a Toeplitz matrix C_α of the lower triangular form

$$C_\alpha = \tau^{-\alpha} \begin{bmatrix} g_{\alpha,0} & 0 & \dots & \dots & \dots & \dots & 0 \\ g_{\alpha,1} & g_{\alpha,0} & \ddots & & & & \vdots \\ g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} & \ddots & & & \vdots \\ \vdots & \ddots & g_{\alpha,1} & g_{\alpha,0} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & g_{\alpha,1} & g_{\alpha,0} & 0 \\ g_{\alpha,n_t} & g_{\alpha,n_t-1} & \dots & \dots & g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} \end{bmatrix} . \tag{7}$$

2.4 Problem structure

Since the spatial grids introduced on Ω, $\Omega_{\bar{y}}$ and Ω_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 tensor-product notation,

$$\begin{aligned} {}^{\mathrm{R}}D_{x_1}^{\beta_1} + {}^{\mathrm{R}}D_{x_2}^{\beta_2} \quad &\to \quad L = L_{\beta_1} \otimes I_{n_2} + I_{n_1} \otimes L_{\beta_2}, \\ {}^{\mathrm{R}}D_{x_1}^{\beta_1} + {}^{\mathrm{R}}D_{x_2}^{\beta_2} + {}^{\mathrm{R}}D_{x_3}^{\beta_3} \quad &\to \quad L = L_{\beta_1} \otimes I_{n_2} \otimes I_{n_3} + I_{n_1} \otimes L_{\beta_2} \otimes I_{n_3} + I_{n_1} \otimes I_{n_2} \otimes L_{\beta_3}, \end{aligned} \tag{8}$$

for $d = 2$ and $d = 3$, respectively. Here and later I_n denotes an identity matrix of size n.

Assuming that L denotes the discretization of spatial derivatives (8), we obtain the discretization of the FDEs as follows

$$Ay + M_3 u = g, \qquad A = C_\alpha \otimes I_n - I_{n_t} \otimes L, \tag{9}$$

where n denotes the total number of space mesh points, $M_3 u$ 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 $n = n_1 n_2$ for a two-dimensional problem and $n = n_1 n_2 n_3$ 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 Ω_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

$$J(y, u) = \frac{1}{2} (y - \bar{y})^\top M_1 (y - \bar{y}) + \frac{\gamma}{2} u^\top M_2 u, \tag{10}$$

where M_1 and M_2 contain the quadrature weights used to evaluate the norm. We use a trapezoidal rule, that coincides with a 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 $\Omega \backslash \Omega_{\bar{y}}$. 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

$$M_1 = \tau h_1 \cdots h_d \cdot \operatorname{diag}(\theta_{\bar{y}}), \quad \text{and} \quad M_2 = \tau h_1 \cdots h_d \cdot I_{N'}. \tag{11}$$

A standard Lagrangian approach results in the following discrete functional which needs to be minimized:

$$\Lambda(y, u, p) = J(y, u) + p^\top V^\top (Ay + M_3 u - g) \tag{12}$$

with p corresponds to the discretized adjoint variable p. The matrix V is used to associate p with a grid function (see [9]), and is written similarly to (11), $V = \tau h_1 \cdots h_d \cdot I_N$.

To obtain the required optimality conditions, we differentiate Λ with respect to y, u and p. This leads to the following first order, Karush-Kuhn-Tucker (KKT), system:

$$\underbrace{\begin{bmatrix} M_1 & 0 & A^\top V \\ 0 & \gamma M_2 & M_3^\top V \\ V^\top A & V^\top M_3 & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} y \\ u \\ p \end{bmatrix} = \begin{bmatrix} M_1 \bar{y} \\ 0 \\ V^\top g \end{bmatrix} =: \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}, \tag{13}$$

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

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

Note that 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.,

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

(see [8]). 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 (13). 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 (13) 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 [64] is well established. For standard saddle point problems in particular this is a method of choice, and is considered state of the art when the problem is of very high dimension (see [8, 22]). Such a Krylov solver proceeds by building up a Krylov subspace

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

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

As the convergence can sometimes be very slow the problem is modified using a preconditioning matrix $\mathcal{P}$, i.e.,

$$\mathcal{A} y = f \quad \Leftrightarrow \quad \mathcal{P}^{-1} \mathcal{A} y = \mathcal{P}^{-1} f.$$

We again refer to [8, 22] for a detailed overview of preconditioners for saddle point problems. In this paper, we follow a result presented in [47], where it was shown that a good strategy when developing preconditioners for a saddle point system,

$$\begin{bmatrix} \Phi & \Psi^T \\ \Psi & 0 \end{bmatrix} y = f,$$

is based on efficient and cheap to evaluate approximations $\widetilde{\Phi}$ and $\widetilde{S}$ of the $(1,1)$-block Φ and the (negative) Schur complement $S := \Psi\Phi^{-1}\Psi^\top$, respectively. This leads to preconditioners of the form

$$\mathcal{P} = \begin{bmatrix} \widetilde{\Phi} & 0 \\ 0 & \widetilde{S} \end{bmatrix} \quad \text{or} \quad \mathcal{P} = \begin{bmatrix} \widetilde{\Phi} & 0 \\ \Psi & -\widetilde{S} \end{bmatrix}.$$

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

$$S = AM_1^{-1}A^\top + \frac{1}{\gamma}M_3M_2^{-1}M_3^\top. \tag{14}$$

The authors have recently obtained robust Schur complement approximations using a matching approach [53, 54, 70] that in our case leads to

$$\widetilde{S} = \widetilde{A}M_1^{-1}\widetilde{A}^\top, \tag{15}$$

with $\widetilde{A} = -A + \widetilde{I}$ and $\widetilde{I} = I_{n_t} \otimes \frac{1}{\sqrt{\gamma}}I_n$. The term $\widetilde{I}$ is chosen so that the outer term in the above Schur complement approximation accurately reflects the second term of the exact Schur complement, that is $\widetilde{I}M_1^{-1}\widetilde{I}^\top \approx \frac{1}{\gamma}M_3M_2^{-1}M_3^\top$.

Before discussing the efficient solution of the block $\widetilde{A}$, which is needed for the efficient evaluation of the Schur complement preconditioner, 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 (14) is represented by our approximation (15). We measure this by considering the eigenvalues of the preconditioned Schur complement $\widetilde{S}^{-1}S$, or (following [55]), by examining the Rayleigh quotient

$$R := \frac{v^\top Sv}{v^\top \widetilde{S}v},$$

for any non-zero $v \in \mathbb{R}^N$.

Theorem 1 *Consider the FDE-constrained control problem (1)–(2) with Grünwald-Letnikov discretizations (5), (7) on the same cubic domain $\Omega = \Omega_{\bar{y}} = \Omega_u$ for the observation, control and system state. Then it holds that $\lambda(\widetilde{S}^{-1}S) \in [\frac{1}{2}, 1)$.*

Proof. From $\Omega = \Omega_{\bar{y}} = \Omega_u$ it follows that $M_1 = M_2 = M_3 = \tau h^d I_N$. Following [53], we write

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

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

$$\frac{1}{2}(a_1 - a_2)^\top (a_1 - a_2) \geq 0 \quad \Leftrightarrow \quad a_1^\top a_1 + a_2^\top a_2 \geq \frac{1}{2}(a_1 + a_2)^\top (a_1 + a_2) \quad \Leftrightarrow \quad R \geq \frac{1}{2}.$$

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

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

It has been shown in [10, Lemma 1] that the matrix $I_{n_t} \otimes L$ is negative semi-definite – we can therefore write that $v^\top \left(I_{n_t} \otimes (L + L^\top) \right) v = 2 v^\top \left(I_{n_t} \otimes L \right) v < 0$, using the symmetry of L.

The matrix $C_\alpha + C_\alpha^\top$ is symmetric positive definite for $\alpha < 1$. This can be argued by the strict diagonal dominance as follows. It is clear that the coefficient $g_{\alpha,k}$ is positive for $k = 0$ and negative for all $k = 1, 2, \ldots$, if $\alpha < 1$, so we need to show that $\sum_{k=0}^{n_t} g_{\alpha_k} > 0$. This follows from the binomial expansion $(1 + z)^\alpha = \sum_{k=0}^{\infty} \binom{\alpha}{k} z^k$: applying this (convergent) sequence at $z = -1$ and using the properties of g_{α_k} gives $\sum_{k=0}^{n_t} g_{\alpha_k} > \sum_{k=0}^{\infty} g_{\alpha_k} = \sum_{k=0}^{\infty} \binom{\alpha}{k} (-1)^k = 0$. Hence, we have that $v^\top \left((C_\alpha + C_\alpha^\top) \otimes I_n \right) v > 0$, using the standard property of Kronecker product matrices that the eigenvalues of $K_1 \otimes K_2$ are equal to those of K_1 multiplied by those of K_2 [41, Chapter 13].

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

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

□

The general case of incomplete observation or control is analytically much more challenging, but there are heuristic techniques [54] to augment the matrices M_1 and M_3 and define fairly good (from computational experience) approximations $\widetilde{\Phi}$ and $\widetilde{S}$.

3.2 First Schur Complement

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

$$S = A M_1^{-1} A^\top + M_3 \left(\gamma M_2 \right)^{-1} M_3^\top.$$

It proceeds by solving the following three systems

$$\begin{aligned} -Sp &= f_3 - AM_1^{-1} f_1 - M_3 \left(\gamma M_2\right)^{-1} f_2 \\ \left(\gamma M_2\right) u &= f_2 - M_3^\top p \\ M_1 y &= f_1 - A^\top p \end{aligned} \tag{16}$$

The disadvantage of this approach is that the invertibility of both observation and control matrix is required for the existence of S. However, when this is indeed the case, the matrices M_1 and M_2 are often simply scaled identities, and their inversion does not lead to computational difficulties. 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

We now allow M_1 and M_2 to have different sizes, so that M_3 becomes 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:

$$\mathcal{A} = \begin{bmatrix} I & -M_1 A^{-1} M_3 & M_1 \\ 0 & \gamma M_2 & 0 \\ 0 & 0 & A \end{bmatrix} \begin{bmatrix} 0 & 0 & M_1 A^{-1} M_3 \left(\gamma M_2\right)^{-1} M_3^\top + A^\top \\ 0 & I & \left(\gamma M_2\right)^{-1} M_3^\top \\ I & A^{-1} M_3 & 0 \end{bmatrix}. \tag{17}$$

The matrix A specifies the underlying PDE model, and is also invertible. Using this factorization the solution of (13) can be computed as follows. We first cast the left factor of the Schur complement to the right-hand side via solving

$$A\tilde{f}_3 = f_3, \quad \left(\gamma M_2\right) \tilde{f}_2 = f_2, \quad A g_2 = M_3 \tilde{f}_2$$

and computing $\tilde{f}_1 = f_1 + M_1 g_2 - M_1 \tilde{f}_3$. Note that if $f_2 = 0$ as in (13), it is trivially observed that $g_2 = 0$. Now we solve the following systems:

$$\begin{aligned} \left(M_1 A^{-1} M_3 \left(\gamma M_2\right)^{-1} M_3^\top + A^\top\right) p &= \tilde{f}_1, \\ \left(\gamma M_2\right) u &= f_2 - M_3^\top p, \\ Ay &= f_3 - M_3 u. \end{aligned} \tag{18}$$

Since the matrix M_2 is often easily invertible, i.e., it is a scaled identity for our problem, and the matrix A has Kronecker-product structure, 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

$$\hat{A} s = \operatorname{vec}\left(M_3 \left(\gamma M_2\right)^{-1} M_3^\top\right), \quad s = \operatorname{vec}(S), \tag{19}$$

where vec$(\cdot)$ stretches matrix entries to a vector. Now the first matrix in (18) reads $\left(M_1 S + A^\top\right)$.

In tensor product representations, the storage complexity of $\hat{A}$ is not much higher than that of A. A certain difficulty arises from the storage of s: 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 (19) 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 (19) is now a low-rank matrix, which yields smaller storage requirements for s to achieve the same accuracy.

4 Numerical algorithms

We now wish to discuss solvers for the matrix systems (13)–(19). While for the one-dimensional PDE the problem has similarities to saddle point problems involving matrix equations of Sylvester-type [68], the higher-dimensional setup requires the use of more specialized tensor solvers. In particular we discuss the tensor-train format introduced in [49]. We first introduce the tensor-train decomposition and then discuss possible solvers.

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, that we now introduce.

The TT format is derived by applying the low-rank approximation recurrently [49]. Given a d-index array $\mathbf{y} = [\mathbf{y}(i_1, \ldots, i_d)]$, with indices varying in ranges $i_k = 1, \ldots, n_k$, $k = 1, \ldots, d$, we may *reshape* a tensor $\mathbf{y}$ 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

$$\overline{i_2 \ldots i_d} = i_2 + (i_3 - 1)n_2 + \cdots + (i_d - 1)n_2 n_3 \cdots n_{d-1}, \tag{20}$$

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

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

The first factor U_1 is of moderate dimension, and can be stored as $\mathbf{y}^{(1)}_{\alpha_1}(i_1) = U_1(i_1, \alpha_1)$, where $\alpha_1 = 1, \ldots, r_1$. The remaining matrix $\Sigma_1 V_1^\top$ depends on indices α_1 and $\overline{i_2 \ldots i_d}$.

Now we *regroup* these indices as follows:

$$Y_2(\overline{\alpha_1 i_2}, \overline{i_3 \dots i_d}) = \Sigma_1(\alpha_1, \alpha_1) V_1^\top(\alpha_1, \overline{i_2 \dots i_d}),$$

and compute the next SVD,

$$Y_2 \approx U_2 \Sigma_2 V_2^\top, \quad \text{where} \quad U_2 \in \mathbb{R}^{r_1 n_2 \times r_2}, \quad V_2 \in \mathbb{R}^{n_3 \cdots n_d \times r_2}. \tag{21}$$

Again, U_2 can be reshaped to a moderately-sized 3D tensor $\mathbf{y}^{(2)}_{\alpha_1,\alpha_2}(i_2) = U_2(\overline{\alpha_1 i_2}, \alpha_2)$, and the decomposition continued for $\Sigma_2 V_2^\top$. Finally, we arrive at the TT format:

$$\mathbf{y}(i_1, \dots, i_d) \approx \sum_{\alpha_1,\dots,\alpha_{d-1}=1}^{r_1,\dots,r_{d-1}} \mathbf{y}^{(1)}_{\alpha_1}(i_1) \mathbf{y}^{(2)}_{\alpha_1,\alpha_2}(i_2) \cdots \mathbf{y}^{(d-1)}_{\alpha_{d-2},\alpha_{d-1}}(i_{d-1}) \mathbf{y}^{(d)}_{\alpha_{d-1}}(i_d), \tag{22}$$

with the total storage estimate $\mathcal{O}(dnr^2)$, where $r \gtrsim r_k$ are called *TT ranks*, and $n \gtrsim n_k$.

A similar construction is introduced for discretized operators in high dimensions. Given a matrix $A = \left[A(\overline{i_1 \dots i_d}, \overline{j_1 \dots j_d})\right] \in \mathbb{R}^{(n_1 \cdots n_d) \times (n_1 \cdots n_d)}$, we decompose it as

$$A(\overline{i_1 \dots i_d}, \overline{j_1 \dots j_d}) = \sum_{\beta_1,\dots,\beta_{d-1}=1}^{R_1,\dots,R_{d-1}} \mathbf{A}^{(1)}_{\beta_1}(i_1, j_1) \mathbf{A}^{(2)}_{\beta_1,\beta_2}(i_2, j_2) \cdots \mathbf{A}^{(d)}_{\beta_{d-1}}(i_d, j_d), \tag{23}$$

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 (22), 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. $r_k(f) = R_k(A) r_k(y)$, $k = 1, \dots, d-1$. 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. (21), can be implemented without high-dimensional tensors appearing. Only QR and SVD decompositions of $nr \times r$ matrices are involved, cf. Section 4.2. The total complexity of this procedure is $\mathcal{O}(dnr^3)$, 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 (20) 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*,

$$i = \overline{i_1 \dots i_l} = i_1 + 2(i_2 - 1) + \cdots + 2^{l-1}(i_l - 1), \qquad i_s \in \{1, 2\}, \quad s = 1, \dots, l.$$

As a result, a *vector* y is reshaped to an l-dimensional *tensor* $\mathbf{y}$, i.e. $\mathbf{y}(i_1, \dots, i_l) = y(i)$, and the TT approximation can be applied to $\mathbf{y}$. The resulting TT format was called the *Quantized* TT (QTT) [35], cf. the term *quantizer* in video compression. If the TT ranks of $\mathbf{y}$ are moderate, we may claim a logarithmic reduction of the total storage, $\mathcal{O}(lr^2) = \mathcal{O}(\log n)$.

For many elementary functions and operators, their TT formats can be written analytically, for example, the discretized Laplace operator [34], the sine, exponential and

polynomial functions, sampled on uniform grids in one [35, 50] and many dimensions [16, 36].

A very important class of matrices, admitting low-rank QTT approximations, are Toeplitz matrices [33]. Given a vector of Toeplitz coefficients $[g_{\alpha,k}]$ (see Section 2.2) in the QTT format with the rank bound r, the Toeplitz matrices C_α, L_β can be analytically constructed in the QTT format with rank bound $2r$. This gives us theoretical support that at least the input data in our problem (13) is well-representable in tensor formats.

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 [70] that the vectors $y, u, p \in \mathbb{R}^{nn_t}$ for space-time saddle point problems can be written as

$$\begin{aligned} y &= \operatorname{vec}(Y) = \operatorname{vec}([y_1, \ldots, y_{n_t}]), \\ u &= \operatorname{vec}(U) = \operatorname{vec}([u_1, \ldots, u_{n_t}]), \\ p &= \operatorname{vec}(P) = \operatorname{vec}([p_1, \ldots, p_{n_t}]). \end{aligned}$$

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

$$(-C_\alpha \otimes I_n + I_{n_t} \otimes L) \operatorname{vec}(Y) = \operatorname{vec}(-I_n Y C_\alpha^\top + L Y I_{n_t}^\top),$$

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 $Y = Y_1 Y_2^\top$, $U = U_1 U_2^\top$, and $P = P_1 P_2^\top$ 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 defined by

$$R_Y = W_Y V_Y^\top \quad \text{with} \quad W_Y \in \mathbb{R}^{n \times r_1},\ V_Y \in \mathbb{R}^{n_t \times r_1}, \tag{24}$$

$$R_U = W_U V_U^\top \quad \text{with} \quad W_U \in \mathbb{R}^{n \times r_2},\ V_U \in \mathbb{R}^{n_t \times r_2}, \tag{25}$$

$$R_P = W_P V_P^\top \quad \text{with} \quad W_P \in \mathbb{R}^{n \times r_3},\ V_P \in \mathbb{R}^{n_t \times r_3}, \tag{26}$$

where $r_i \ll n$. Every Krylov scheme now performs matrix-vector multiplications of $\mathcal{A}$ and the initial residual given in (24)–(26). It is easy to see that this involves the

following multiplications:

$$\text{(first block-row)} \begin{bmatrix} M_{1x}W_Y & L^\top W_P & -W_P \end{bmatrix} \begin{bmatrix} V_Y^\top \\ V_P^\top \\ V_P^\top C_\alpha \end{bmatrix}, \tag{27}$$

$$\text{(second block-row)} \begin{bmatrix} \gamma M_{2x}W_U & M_{3x}^\top W_P \end{bmatrix} \begin{bmatrix} V_U^\top \\ V_P^\top \end{bmatrix}, \tag{28}$$

$$\text{(third block-row)} \begin{bmatrix} LW_Y & -W_Y & M_{3x}W_U \end{bmatrix} \begin{bmatrix} V_Y^\top \\ V_Y^\top C_\alpha^\top \\ V_U^\top \end{bmatrix}, \tag{29}$$

where we denote $M_1 = I_{n_t} \otimes M_{1x}$, $M_2 = I_{n_t} \otimes M_{2x}$, $M_3 = I_{n_t} \otimes M_{3x}$, with matrices M_{ix}, $i = 1, 2, 3$ corresponding to the spatial domains $\Omega_{\bar{y}}, \Omega_u$. Since we always consider the full time interval $[0, T]$ in both observation and control domains, this decomposition is possible.

The expressions above show that a matrix-vector product in matrix form can be written as the product of a left factor and a right factor. We discuss this here in some more detail. The expression (29) represents the third equation of the KKT conditions. It is also clear that the two factors have an increased rank compared to the original factors, for instance V_U and W_U. Hence a truncation of all the factors after the matrix-vector products, via a truncated SVD or a QR decomposition, is used to construct new factors

$$\left\{\tilde{W}_P, \tilde{V}_P\right\} = \mathcal{T}_\varepsilon \left(\begin{bmatrix} LW_Y & -W_Y & M_{3x}W_U \end{bmatrix}, \begin{bmatrix} V_Y^\top \\ V_Y^\top C_\alpha^\top \\ V_U^\top \end{bmatrix} \right).$$

The construction of such a truncation function $\mathcal{T}_\varepsilon$ for the matrix case is further described in [40, 70]. We have now shown that it is possible to maintain the initial low-rank structure throughout a Krylov method. We now briefly discuss the tensor case.

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 allows to rewrite any classical iterative method, such that it keeps all vectors in the tensor format, and performs only structured operations [4, 17, 40]. Let us denote the compression (or *truncation*) procedure from a vector y to a vector $\widetilde{y} \approx y$ as

$$\tilde{y} = \mathcal{T}_\varepsilon(y),$$

where by ε we denote the truncation accuracy in the Frobenius norm. In particular, the TT-MINRES algorithm can be written as shown in Algorithm 1.

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.

Require: Right-hand side f, initial vector y in the TT format, matrix $\mathcal{A}$ as a `MatVec` procedure $g = \mathcal{T}_\varepsilon(\mathcal{A}y)$, preconditioner $\mathcal{P}$ (possibly identity), accuracy ε.
Ensure: Improved solution y.

Start: compute $v_2 = \mathcal{T}_\varepsilon(f - \mathcal{A}y)$, $z_1 = \mathcal{T}_\varepsilon(\mathcal{P}z_2)$, $\gamma_1 = \sqrt{\langle z_1, v_2\rangle}$.
Initialize $c_1 = c_2 = \gamma_0 = 1$, $s_1 = s_2 = 0$, $\eta = \gamma_1$, $v_1 = w_1 = w_2 = [0, \ldots, 0]^\top$.
Iterations:
for $j = 1, 2, \ldots, m$ **do**
  $z_1 = z_1/\gamma_1$, $v_3 = \mathcal{T}_\varepsilon(\mathcal{A}z_1)$.
  $\delta = \langle z_1, v_3\rangle$, $v_3 = \mathcal{T}_\varepsilon\left(v_3 - \frac{\delta}{\gamma_1}v_2 - \frac{\gamma_1}{\gamma_0}v_1\right)$. {Orthogonalize the Krylov vector}
  $z_2 = \mathcal{T}_\varepsilon(\mathcal{P}v_3)$, $\gamma_2 = \sqrt{\langle z_2, v_3\rangle}$.
  $\alpha_0 = c_1\delta - c_0 s_1 \gamma_1$, $\alpha_1 = \sqrt{\alpha_0^2 + \gamma_2^2}$, $\alpha_2 = s_1\delta + c_0 c_1 \gamma_1$, $\alpha_3 = s_0\gamma_1$.
  $c_0 = c_1$, $c_1 = \alpha_0/\alpha_1$, $s_0 = s_1$, $s_1 = \gamma_2/\alpha_1$.
  $w_3 = \frac{1}{\alpha_1}\mathcal{T}_\varepsilon\left(z_1 - \alpha_3 w_1 - \alpha_2 w_2\right)$. {Orthogonalize the preconditioned vector}
  $y = \mathcal{T}_\varepsilon(y + c_1\eta w_3)$. {Correct the solution}
  $\eta = -s_1\eta$, $\gamma_0 = \gamma_1$, $\gamma_1 = \gamma_2$, $v_1 = v_2$, $v_2 = v_3$, $z_1 = z_2$, $w_1 = w_2$, $w_2 = w_3$.
  if $|\eta| < \varepsilon\sqrt{\langle b, \mathcal{P}b\rangle}$ **then** Stop.
end for

Algorithm 1: TT-MINRES

Preconditioning

We have now established the use of a low-rank or tensor Krylov method and have previously discussed block-preconditioners. It is now crucial to evaluate the preconditioners $\mathcal{P}$ given by the block-diagonal matrix/tensor in Section 3.1.

In the matrix case, the inverse of $\tilde{S}$ can be approximated by employing low-rank methods for Sylvester equations to approximately solve for $\tilde{A}$ and $\tilde{A}^\top$. Note that both of these operations mean approximately solving Sylvester-type equations, and computing one matrix multiplication.

The solution of these matrix equations is of crucial importance in many areas of science and engineering. While direct methods such as the Bartels–Stewart algorithm [24] are suitable for moderate matrix sizes, larger problems require the use of iterative schemes. Additionally the storage demand for the solution matrix is often vast, and hence low-rank methods have become a standard tool to approximate the solution of Sylvester equations [68]. The efficient use of alternating direction implicit (ADI) methods was established over the last decade [5, 6, 7, 69]. 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 [26] 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 [21, 67] for Lyapunov equations and later adapted for the case of Sylvester equations [68]. 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 $(L - \frac{1}{\sqrt{\gamma}} I_n)$ in

$$C_\alpha \otimes I_n - I_{n_t} \otimes (L - \frac{1}{\sqrt{\gamma}} I_n).$$

Additionally, the sequence requires the inverse or approximate inverse of both C_α and $(L - \frac{1}{\sqrt{\gamma}} I_n)$. Since they are Toeplitz matrices, we can employ fast iterative solvers using circulant preconditioners, following [10].

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.

4.3 Alternating solvers

It is often observed that Krylov vectors may require much larger TT ranks than the solution of the problem, and hence we now introduce methods that provide better handles on the rank-growth of the approximate solution. *Alternating* iterative tensor algorithms avoid this problem by directly seeking the elements of the tensor format for the solution. The Alternating Least Squares method (see [39] and references there) was developed to fit given data to a low-rank model. The *Density Matrix Renormalization Group* (DMRG) algorithm [74] was initially proposed for solution of ground state eigenvalue problems in quantum physics, and then extended to linear systems [19, 29, 31].

Given a TT format (22) for the approximate solution y, we may collect its first, resp. last, k TT factors into the *interface matrices*,

$$\begin{aligned} Y^{(1:k)}(\overline{i_1 \dots i_k}, \alpha_k) &= \sum_{\alpha_1, \dots, \alpha_{k-1}=1}^{r_1, \dots, r_{k-1}} \mathbf{y}^{(1)}_{\alpha_1}(i_1) \mathbf{y}^{(2)}_{\alpha_1, \alpha_2}(i_2) \cdots \mathbf{y}^{(k)}_{\alpha_{k-1}, \alpha_k}(i_k), \\ Y^{(k:d)}(\alpha_{k-1}, \overline{i_k \dots i_d}) &= \sum_{\alpha_k, \dots, \alpha_{d-1}=1}^{r_k, \dots, r_{d-1}} \mathbf{y}^{(k)}_{\alpha_{k-1}, \alpha_k}(i_k) \cdots \mathbf{y}^{(d)}_{\alpha_{d-1}}(i_d), \end{aligned} \tag{30}$$

where $Y^{(1:k)} \in \mathbb{R}^{n_1 \cdots n_k \times r_k}$ and $Y^{(k:d)} \in \mathbb{R}^{r_{k-1} \times n_k \cdots n_d}$. We then comprise the *frame matrix*,

$$Y_{\neq k} = Y^{(1:k-1)} \otimes I_{n_k} \otimes \left(Y^{(k+1:d)} \right)^\top \in \mathbb{R}^{n_1 \cdots n_d \times r_{k-1} n_k r_k}. \tag{31}$$

Note that the column size of the frame matrix is exactly equal to the number of elements in the TT block $\mathbf{y}^{(k)}$. Therefore, the TT format (22) 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, $y^{(k)}(\overline{\alpha_{k-1} i_k \alpha_k}) = \mathbf{y}^{(k)}_{\alpha_{k-1}, \alpha_k}(i_k)$.

Now let us consider $Ay = f$ as an overdetermined system on $y^{(k)}$, provided the TT format (22) is inserted instead of y, i.e. $A Y_{\neq k} y^{(k)} = f$. A simple way to resolve this system is to project it onto the frame matrix. We solve

$$A_k y^{(k)} = f_k, \qquad A_k = Y_{\neq k}^\top A Y_{\neq k}, \quad f_k = Y_{\neq k}^\top f. \tag{32}$$

Using the *orthogonalizations* of the TT blocks via the QR decompositions of their matrix reshapings, we can always make the frame matrix orthogonal [66], that is $Y_{\neq k}^\top Y_{\neq k} = I$. This ensures the stability of the problem, i.e. $\mathrm{cond}(A_k) \leq \mathrm{cond}(A)$. Iterating for all dimensions $k = 1, \dots, d$ (hence the name "alternating"), we obtain the simple *one-block* DMRG, or Alternating Linear Scheme (ALS) [29] 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 [63]: 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 gives the DMRG algorithm in its initial, *two-block* version [74]. Instead of one block $y^{(k)}$, we update both $y^{(k)}$ and $y^{(k+1)}$ at each step. We consider the two-block frame matrix

$$Y_{\neq k,k+1} = Y^{(1:k-1)} \otimes I_{n_k} \otimes I_{n_{k+1}} \otimes \left(Y^{(k+2:d)}\right)^\top \in \mathbb{R}^{n_1 \cdots n_d \times r_{k-1} n_k n_{k+1} r_{k+1}}, \quad (33)$$

and solve effectively the two-dimensional problem

$$A_{k,k+1} y^{(k,k+1)} = f_{k,k+1}, \qquad A_{k,k+1} = Y_{\neq k,k+1}^\top A Y_{\neq k,k+1}, \quad f_{k,k+1} = Y_{\neq k,k+1}^\top f. \quad (34)$$

The new *superblock* $y^{(k,k+1)}$ is then reshaped to a matrix $Y^{\langle k,k+1\rangle} \in \mathbb{R}^{r_{k-1} n_k \times n_{k+1} r_{k+1}}$, and the low-rank decomposition $Y^{\langle k,k+1\rangle} \approx U \Sigma V^\top$ is computed. The new rank $r'_k = \mathrm{rank}_\varepsilon(Y^{\langle k,k+1\rangle})$ is likely to differ from r_k. After the SVD is computed, we use its factors to rewrite $\mathbf{y}^{(k)}$ and $\mathbf{y}^{(k+1)}$, and replace $r_k := r'_k$. This maintains the orthogonality of the frame matrices automatically.

This method has become the state-of-the-art in the simulation of quantum states of spin chains due to its impressively fast convergence. It is enough to start from a random low-rank initial guess and perform each SVD with the target threshold ε: the ranks are determined adaptively to this tolerance. However, the outlined quantum state problem is the extreme eigenvalue problem for a Hermitian matrix. For linear systems with non-symmetric matrices, even this two-block DMRG method may converge to an incorrect solution.

Another way to increase TT ranks is to *enrich* TT blocks explicitly. Suppose we are iterating $k = 1, \dots, d$, then, after the solution of (32), we may expand

$$\mathbf{y}^{(k)}(i_k) := \begin{bmatrix} \mathbf{y}^{(k)}(i_k) & \mathbf{z}_k^{(k)}(i_k) \end{bmatrix}, \quad \mathbf{y}^{(k+1)}(i_{k+1}) := \begin{bmatrix} \mathbf{y}^{(k+1)}(i_{k+1}) \\ \mathbf{0} \end{bmatrix}, \quad (35)$$

where $\mathbf{z}_k^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times \rho_k}$ is some auxiliary tensor, and the zero-block in $\mathbf{y}^{(k+1)}$ has the corresponding sizes $\rho_k \times n_{k+1} \times r_{k+1}$. This step does not change the whole solution y. However, when we proceed to the next block $\mathbf{y}^{(k+1)}$, the interface matrix $Y^{(1:k)}$ and

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 or iteration limit is not hit **do**
2: Orthogonalize y and z s.t. $Y^{(k:d)}$ and $Z^{(k:d)}$ are orthogonal for $k = 2, \dots, d$.
3: **for** $k = 1, \dots, d$ **do**
4: Build and solve (32) for the TT block $y^{(k)}$.
5: Update the residual block $z^{(k)} = \left(Z^{(1:k-1)} \otimes I_{n_k} \otimes \left(Z^{(k+1:d)}\right)^\top\right)^\top (f - Ay)$.
6: **if** $k < d$ **then**
7: Build the enrichment $z_k^{(k)} = \left(Y^{(1:k-1)} \otimes I_{n_k} \otimes \left(Z^{(k+1:d)}\right)^\top\right)^\top (f - Ay)$.
8: Perform the expansion (35).
9: Orthogonalize $\mathbf{y}^{(k)}$ and $\mathbf{z}^{(k)}$ s.t. $Y^{(1:k)}$ and $Z^{(1:k)}$ are orthogonal.
10: **end if**
11: **end for**
12: **end while**

Algorithm 2: AMEn method

the frame matrix $Y_{\neq k+1}$ carries $\mathbf{z}_k^{(k)}$, so the Galerkin reduction (32) in the step $k+1$ is performed to a wider basis than in the ALS method. This can not only increase the TT rank by ρ_k, but also facilitate the convergence, if we select the augmentation $\mathbf{z}_k^{(k)}$ properly.

The idea [20] 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

$$y(\overline{i_1 \dots i_d}) + hz(\overline{i_1 \dots i_d}) = \begin{bmatrix}\mathbf{y}^{(1)}(i_1) & \mathbf{z}^{(1)}(i_1)\end{bmatrix} \begin{bmatrix}\mathbf{y}^{(2)}(i_2) & \\ & \mathbf{z}^{(2)}(i_2)\end{bmatrix} \dots \begin{bmatrix}\mathbf{y}^{(d)}(i_d) \\ h\mathbf{z}^{(d)}(i_d)\end{bmatrix}.$$

Note that the first factor here has the same form as the enrichment of $\mathbf{y}^{(k)}$ in (35); therefore, (35) performs the *first step* of the TT format addition. This is sufficient, since in the next step we solve (32) and recover correct z-related entries in $\mathbf{y}^{(k+1)}$ that *minimize the energy function* if $A = A^\top > 0$. Due to this property, the new method was called AMEn (alternating minimal energy).

To compute $\mathbf{z}_k^{(k)}$ 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 $\rho_1, \dots, \rho_{d-1}$ and update it towards $f - Ay$ by the *secondary* ALS iteration solving $\|z - (f - Ay)\|_2 \to \min$. The entire procedure is summarized in Algorithm 2. Notice also that after the enrichment (35) we need to orthogonalize $\mathbf{y}^{(k)}$ 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 $k = 1, \dots, d$ (see [18, 19, 20] for details).

5 Numerical results

We solve the problem (13) in the QTT format for different inputs and parameters, and compare three procedures: the first Schur Complement scheme (16), where the AMEn Algorithm 2 is applied to solve linear systems; the Second Schur Complement scheme (18), again with the AMEn solver; and the MINRES Algorithm 1 with the preconditioner (15). Below we refer to these methods as "SC1", "SC2" and "MR", respectively. In the MR approach, the linear systems with the matrices $\tilde{A}$ and $\tilde{A}^\top$ are also solved using the AMEn method. The three components y, u and p and the blocks in the matrix (13) 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 $y_\star$, we denote

$$\mathcal{E}(y_\star) = \frac{\|y - y_\star\|_2}{\|y_\star\|_2}. \tag{36}$$

The examples were implemented on a base of the TT-Toolbox package[1] 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.

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.,

$$x_1(i) = ih, \quad x_2(j) = jh, \quad t(k) = k\tau,$$
$$i, j = 1, \ldots, 2^l, \quad k = 1, \ldots, 2^{2l}, \qquad h = \frac{1}{2^l + 1}, \quad \tau = h^2.$$

The target function is

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

We vary the grid level l, the regularization parameter γ and the tensor approximation tolerance ε. Note that we use the same threshold ε both for the tensor approximation and also as the stopping tolerance for our numerical schemes.

[1] http://github.org/oseledets/TT-Toolbox, version Nov 10, 2014

Table 1: Complete data test. Left: CPU times (sec.), TT ranks and iterations vs. grid level l. Right: discrepancies between solutions computed by SC2, MINRES, Full and SC1 schemes.

l	SC1		SC2		MR			Full	SC2	MR	Full
	time	rank	time	rank	time	rank	iter	time	$\mathcal{E}(y_{SC1})$		
3	0.596	10	1.396	9	2.980	13	9	0.552	1.47e-6	2.24e-7	2.51e-7
4	0.489	12	1.532	14	4.823	20	11	7.695	2.45e-6	3.11e-7	2.90e-7
5	1.021	18	4.103	21	11.45	26	15	232.3	7.07e-6	5.97e-7	5.64e-7
6	2.280	22	11.69	27	24.70	36	15	7513	5.06e-5	8.63e-7	8.36e-7
7	5.061	25	27.61	36	83.99	49	15	–	2.00e-4	9.14e-7	–
8	10.46	28	55.45	41	245.7	62	15	–	8.16e-4	1.18e-6	–
9	18.90	29	99.94	45	984.9	78	17	–	2.41e-3	2.04e-6	–
10	28.03	29	175.1	51	3494	100	17	–	1.03e-2	5.27e-6	–
11	47.49	29	257.8	55	33005	255	25	–	2.18e-2	2.52e-5	–

Performance with respect to the grid size

We fix $\gamma = 10^{-6}$, $\varepsilon = 10^{-6}$, and vary the grid size in the range $l = 3, \dots, 11$. The results are shown in Table 1. It can be seen that the iteration numbers for MINRES are rather 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 approach. 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 (13) is solved via classical MINRES with the full storage of all vectors. Toeplitz matrices in A were multiplied by vectors via the FFT, and the linear systems with $\tilde{A}$ and $\tilde{A}^\top$ from (15) were solved using the BICGSTAB method. We observe a good agreement of the solutions, but the CPU time grows dramatically and prevents calculations 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 γ. The grid level is $l = 7$, and $\varepsilon = 10^{-6}$. We can see that the computed state approaches the desired state further when γ gets smaller. Notice that the CPU times of all methods decrease. It is reasonable that (13) becomes easier to solve: since all domains coincide, in the limit $\gamma \to 0$ we would just copy $y = \bar{y}$, i.e. solve the linear system with the identity matrix. 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 γ.

Table 2: Complete data test. Left: CPU times (sec.) and discrepancies with the observation data. Right: discrepancies with SC1.

γ	SC1		SC2		MR		SC2	MR
	time	$\mathcal{E}(\bar{y})$	time	$\mathcal{E}(\bar{y})$	time	$\mathcal{E}(\bar{y})$	$\mathcal{E}(y_{SC1})$	$\mathcal{E}(y_{SC1})$
10^{-2}	23.91	9.33e-1	31.56	9.33e-1	4884	9.33e-1	4.53e-5	4.12e-5
10^{-4}	14.47	3.31e-1	26.97	3.31e-1	707.0	3.31e-1	1.39e-4	6.94e-6
10^{-6}	5.226	9.53e-2	27.72	9.53e-2	88.38	9.53e-2	1.76e-4	9.51e-7
10^{-8}	1.354	5.27e-3	29.51	5.29e-3	17.18	5.27e-3	4.23e-4	1.18e-6
10^{-10}	1.217	5.75e-5	31.26	4.28e-4	5.531	5.75e-5	4.24e-4	9.91e-7
10^{-12}	1.189	1.57e-6	29.43	3.78e-4	2.450	5.75e-7	3.78e-4	1.48e-6

Table 3: Complete data test. Left: CPU times (sec.) of three methods vs. tensor approximation tolerance ε. Right: discrepancies of SC2 and MINRES with SC1.

ε	SC1	SC2	MR	$\frac{\|y_{SC2}-y_{SC1}\|}{\|y_{SC1}\|}$	$\frac{\|y_{MR}-y_{SC1}\|}{\|y_{SC1}\|}$
10^{-2}	0.728	1.471	3.400	2.50e-1	7.06e-3
10^{-4}	1.449	4.899	9.869	1.38e-2	1.01e-4
10^{-6}	4.962	27.53	75.07	1.99e-4	9.11e-7
10^{-8}	17.26	163.6	504.1	1.18e-6	8.57e-9
10^{-10}	37.34	880.2	2846	3.76e-9	6.27e-10

Performance with respect to the approximation threshold (Table 3)

Finally, we fix $l = 7$, $\gamma = 10^{-6}$, and vary ε 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. Note that ε is here the tolerance both for the tensor truncation and for the stopping criterion of the iterative solvers. The discrepancies demonstrate almost perfect linear dependence on ε.

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.

Performance with respect to the grid size

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 $\alpha = 0.5$, $\beta_1 = \beta_2 = 1.5$.

In this example, the MINRES method converges very slowly, such that we could not

Table 4: Incomplete data test. Left: CPU times (sec.), TT ranks and iterations vs. grid level l. Right: solution errors.

l	SC2		MR			Full	SC2		MR	Full
	time	rank	time	rank	iter	time	$\mathcal{E}(y_\star)$	$\mathcal{E}(\bar{y})$	$\mathcal{E}(y_\star)$	$\mathcal{E}(y_\star)$
3	1.081	23	5.247	44	11	5.521	4.25e-7	2.42e-3	2.27e-4	8.31e-9
4	2.426	26	27.88	138	15	124.1	1.69e-6	9.12e-3	1.49e-4	8.93e-9
5	7.039	42	1018	752	17	3542	3.34e-6	1.82e-2	3.68e-4	3.53e-5
6	20.96	56		–		–	1.33e-5	2.98e-2	–	–
7	54.19	64		–		–	4.29e-5	3.62e-2	–	–
8	120.0	72		–		–	3.21e-4	3.76e-2	–	–
9	223.7	75		–		–	1.28e-3	3.76e-2	–	–
10	414.4	76		–		–	4.25e-3	3.78e-2	–	–
11	743.7	81		–		–	1.05e-2	3.94e-2	–	–

proceed for grids larger than $l = 5$. We observe low iteration numbers but the rank increase is too dramatic for further mesh-levels.

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. The results are shown in Table 4.

A weakness of the Schur Complement scheme is the growth of the error, asymptotically by a factor 3 from level to level. This is because the condition number of the system matrix grows like $(2^l)^\beta$, and so does the error when we compute the approximate inverse matrix in (19). This can be verified by solving (19) with a higher accuracy (say, 10^{-8}), while all other steps in (18) are performed with the same $\varepsilon = 10^{-6}$. For $l = 9$, it gives $\frac{\|y_{SC2} - y_\star\|}{\|y_\star\|} \approx 1.58\text{e-}5$ in 667 seconds. So, the loss of accuracy can be removed by using a smaller tolerance. The computation time increases, but still remains on a reasonable level.

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 γ 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, and instead only investigate SC2.

In this test, the part $M_1 A^{-1} M_3 \left(\gamma M_2\right)^{-1} M_3^\top$ of the Schur complement matrix in (18) is indefinite. For small γ the whole Schur complement may become indefinite, which makes the calculations difficult. Therefore, we are limited by the range of γ 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 decays asymptotically proportional to $\sqrt{\gamma}$, as expected.

Table 5: Incomplete data test. Left: CPU times (sec.) of the SC2 method vs. regularization parameter γ. Right: solution errors.

γ	time	rank	$\mathcal{E}(y_\star)$	$\mathcal{E}(\bar{y})$
10^{-1}	37.66	44	1.65e-6	9.95e-1
10^{-2}	41.62	47	1.01e-5	9.61e-1
10^{-4}	47.02	52	3.67e-5	3.48e-1
10^{-6}	54.65	64	4.40e-5	3.62e-2
10^{-7}	74.23	68	6.42e-5	9.66e-3

Performance with respect to α and β (Fig. 1)

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

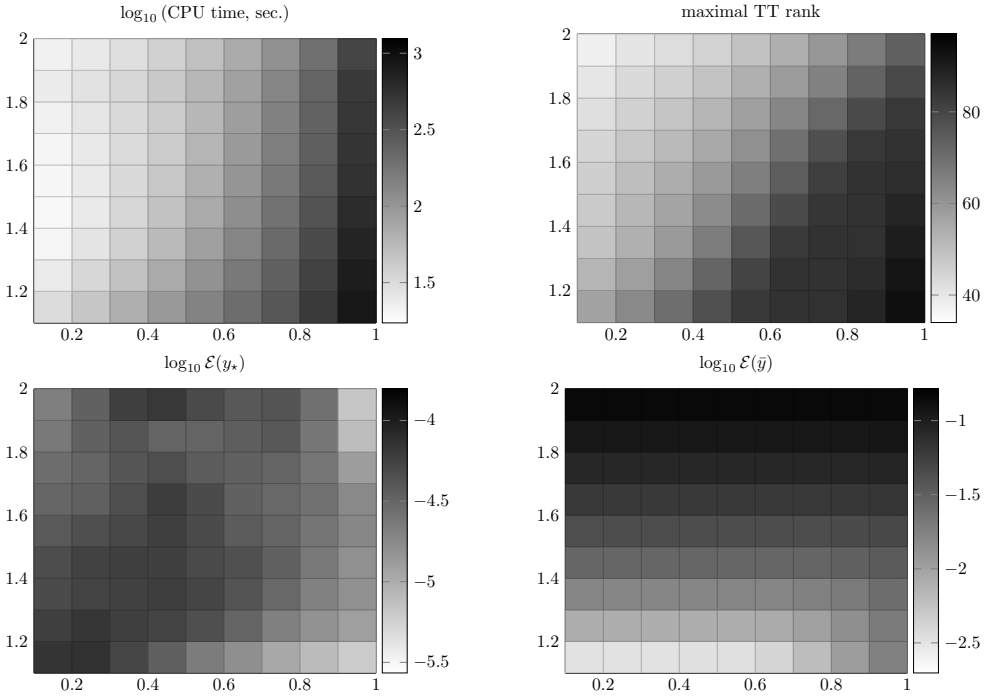
It is interesting that TT ranks and CPU times increase not with the total differentiation order, but towards 1 in both directions. The condition number of the Caputo matrix C_α grows monotonously with $\alpha \in (0, 1]$. The Caputo matrix is close to the identity when α is small and turns to the standard first-order difference at $\alpha = 1$. However, the Riemann-Liouville matrix L_β 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_β depends continuously on β, so the condition number is minimal near $\beta = 1.2$. This inconsistency makes the problem harder to solve when β approaches 1. Even more difficult are the cases $\beta < 1$ and $\alpha > 1$: the matrices C_α and L_β 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 error remains 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 $\|y - \bar{y}\|$.

Performance with respect to β_1 and β_2 (Fig. 2)

We now fix $\alpha = 0.5$, and vary β_1, β_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. α and β, there is an anisotropy w.r.t. β_1 and β_2, since the target function $\bar{y}(x_1, x_2)$ is not symmetric w.r.t. x_1 and x_2. Both error indicators behave similarly to those in the previous figure.

Figure 1: Incomplete data test. CPU times, TT ranks and errors vs. α (x axis) and β (y axis).



5.3 Three-dimensional problem, incomplete data

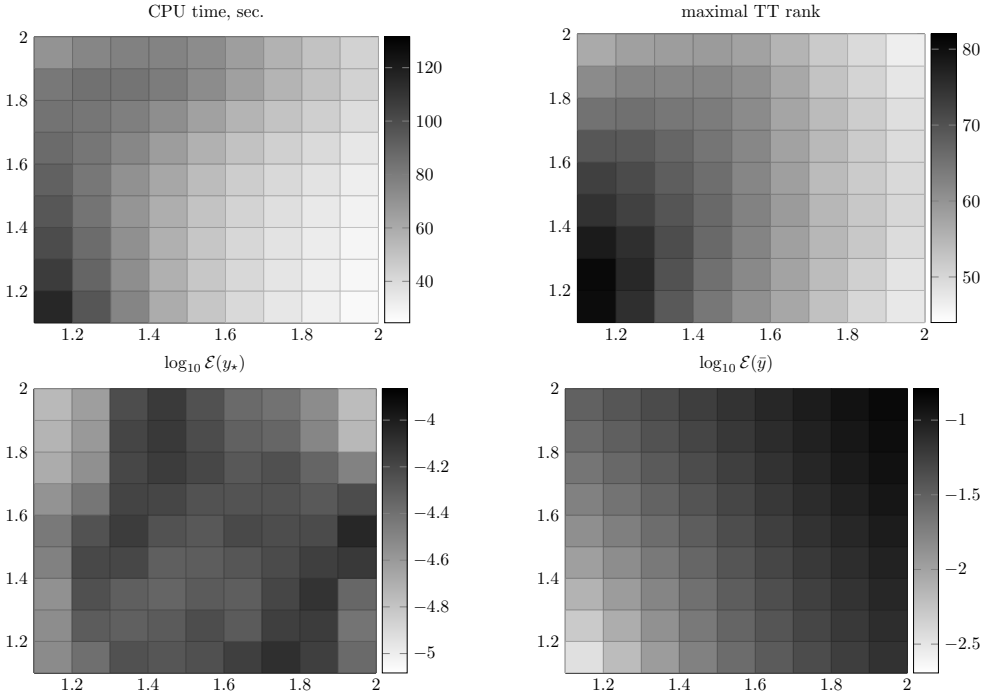
To verify the applicability of our technique to a significantly larger problem, we solve the problem (1c) in a three-dimensional space domain plus time. We set $\Omega = [0,1]^3$, $\alpha = 0.5$, $\beta_1 = \beta_2 = \beta_3 = 1.5$, and $f = 0$. The target function is given by

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

On discretization level l we use the uniform $2^l \times 2^l \times 2^l \times 2^{2l}$ grid. The observation and control domains are $\Omega_{\bar{y}} = \Omega_u = \left[\frac{3}{8}, \frac{5}{8}\right]^3$, with the regularization and stopping tolerances given by $\gamma = \varepsilon = 10^{-6}$.

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 6 we show the performance w.r.t. the grid size. Again, the reference solution $y_\star$ is obtained by the SC2 method with accuracy $\varepsilon_\star = 10^{-8}$. 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

Figure 2: Incomplete data test. CPU times, TT ranks and errors vs. β_1 (x axis) and β_2 (y axis).



does the accuracy of the matrix inversion in the Schur complement. Nevertheless, at moderate grid levels (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): the (inverse) fractional operator moves it to the right, towards the centered Gaussian distribution.

6 Conclusions

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

Table 6: 3D test. Left: CPU times (sec.) and TT ranks of the SC2 method vs. grid level l. Right: solution errors.

l	time	rank	$\mathcal{E}(y_\star)$	$\mathcal{E}(\bar{y})$
3	1.253	21	4.58e-7	1.05e-2
4	5.425	43	2.22e-6	8.02e-3
5	19.96	71	9.43e-6	6.60e-3
6	61.11	83	7.62e-5	8.11e-3
7	135.5	85	4.84e-4	9.05e-3
8	302.8	86	2.27e-3	9.38e-3
9	583.6	86	9.92e-3	1.20e-2
10	941.6	86	5.54e-2	4.58e-2
11	1132	85	2.11e-1	1.73e-1

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.

Appendix. Implementation of the TT solvers

The AMEn Algorithm 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 made using these methods, tensor product algorithms 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 `'trunc_norm','fro'` to `amen_solve2`. To address the ill conditioning of S in (16), for the SC1 method we take `local_restart` $= 100$, `max_full_size` $= 2500$ and `tol_exit` $= 5\varepsilon$. the first two parameters increase the accuracy of the solver for (32), and the third parameter removes unnecessary iterations near a tolerance of ε by stopping at 5ε. Since the TT ranks of A^{-1} are rather large, we set `kickrank` $= 10$ for (19). This allows to increase the ranks faster and have fewer iterations. The Schur complement S in (18) is also ill-conditioned, and we set `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

[1] G. Adomian, *Solving Frontier problems of Physics: The decomposition method*, Kluwer Academic Publishers, 1994.

[2] O. P. Agrawal, *A general formulation and solution scheme for fractional optimal control problems*, Nonlinear Dynam., 38 (2004), pp. 323–337.

[3] T. Akbarian and M. Keyanpour, *A new approach to the numerical solution of fractional order optimal control problems*, Appl. Appl. Math., 8(2) (2013), pp. 523–534.

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

[5] U. Baur and P. Benner, *Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic*, Computing, 78 (2006).

[6] P. Benner and P. Kürschner, *Computing real low-rank solutions of Sylvester equations by the factored ADI method*, MPI Magdeburg Preprint MPIMD/13-05, May 2013.

[7] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*, J. Computat. Appl. Math., 233 (2009), pp. 1035–1045.

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

[9] M. Benzi, E. Haber, and L. Taralli, *A preconditioning technique for a class of PDE-constrained optimization problems*, Adv. Comput. Math., 35 (2011), pp. 149–173.

[10] T. Breiten, V. Simoncini, and M. Stoll, *Fast iterative solvers for fractional differential equations*, Submitted, (2014).

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

[12] M. Caputo and F. Mainardi, *Linear models of dissipation in anelastic solids*, Rivista del Nuovo Cimento, 1 (1971), pp. 161–198.

[13] W. Deng and J. S. Hesthaven, *Local discontinuous galerkin methods for fractional diffusion equations*, ESAIM: Math. Model. Num., 47 (2013), pp. 1845–1864.

[14] K. Diethelm, *The analysis of fractional differential equations: an application-oriented exposition using differential operators of Caputo type*, Lecture Notes in Mathematics, Springer, 2004.

[15] K. Diethelm, N. J. Ford, A. D. Freed, and Y. Luchko, *Algorithms for the fractional calculus: a selection of numerical methods*, Comput. Method Appl. M., 194 (2005), pp. 743–773.

[16] S. Dolgov and B. Khoromskij, *Two-level QTT-Tucker format for optimized tensor calculus*, SIAM J. on Matrix An. Appl., 34 (2013), pp. 593–623.

[17] S. V. Dolgov, *TT-GMRES: solution to a linear system in the structured tensor format*, Russ. J. Numer. Anal. Math. Model., 28 (2013), pp. 149–172.

[18] ——, *Tensor product methods in numerical simulation of high-dimensional dynamical problems*, PhD thesis, University of Leipzig, 2014.

[19] S. V. Dolgov and I. V. Oseledets, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739.

[20] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J Sci. Comput., 36 (2014), pp. A2248–A2271.

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

[22] H. Elman, D. Silvester, and 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.

[23] A. Freed and K. Diethelm, *Fractional calculus in biomechanics: a 3D viscoelastic model using regularized fractional-derivative kernels with application to the human calcaneal fat pad*, Biomech. Model. Mechanobiol., 5 (2006), pp. 203–215.

[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[25] R. Gorenflo and F. Mainardi, *Fractional calculus: integral and differential equations of fractional order*, in Fractals and Fractional Calculus in Continuum Mechanics, A. Carpinteri and F. Mainardi, eds., Springer, 1997, pp. 223–276.

[26] L. Grasedyck and W. Hackbusch, *A multigrid method to solve large scale Sylvester equations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 870–894.

[27] R. Hilfer, P. Butzer, U. Westphal, J. Douglas, W. Schneider, G. Zaslavsky, T. Nonnemacher, A. Blumen, and B. West, *Applications of fractional calculus in physics*, vol. 5, World Scientific, 2000.

[28] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Mathematical Modelling: Theory and Applications, Springer-Verlag, New York, 2009.

[29] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713.

[30] K. Ito and K. Kunisch, *Lagrange multiplier approach to variational problems and applications*, vol. 15 of Advances in Design and Control, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008.

[31] E. Jeckelmann, *Dynamical density–matrix renormalization–group method*, Phys. Rev. B, 66 (2002), p. 045114.

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

[33] V. Kazeev, B. Khoromskij, and E. Tyrtyshnikov, *Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity*, SIAM J. Sci. Comput., 35 (2013), pp. A1511–A1536.

[34] V. A. Kazeev and B. N. Khoromskij, *Low-rank explicit QTT representation of the Laplace operator and its inverse*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 742–758.

[35] B. N. Khoromskij, $\mathcal{O}(d \log n)$*–Quantics approximation of* N*–*d *tensors in high-dimensional numerical modeling*, Constr. Approx., 34 (2011), pp. 257–280.

[36] B. N. Khoromskij and I. V. Oseledets, *DMRG+QTT approach to computation of the ground state for the molecular Schrödinger operator*, Preprint 69, MPI MIS, Leipzig, 2010.

[37] ——, *Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs*, Comput. Methods Appl. Math., 10 (2010), pp. 376–394.

[38] R. Koeller, *Applications of fractional calculus to the theory of viscoelasticity*, J. Appl. Mech., 51(2) (1984), pp. 299–307.

[39] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

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

[41] A. J. Laub, *Matrix Analysis for Scientists and Engineers*, Society for Industrial and Applied Mathematics (SIAM), 2005.

[42] M. Meerschaert and C. Tadjeran, *Finite difference approximations for fractional advection–dispersion flow equations*, J. Comput. Appl. Math., 172 (2004), pp. 65–77.

[43] ——, *Finite difference approximations for two-sided space-fractional partial differential equations*, Appl. Numer. Math., 56 (2006), pp. 80–90.

[44] K. S. Miller and B. Ross, *An Introduction to the Fractional Calculus and Fractional Differential Equations*, John Wiley & Sons, 1993.

[45] G. M. Mophou, *Optimal control of fractional diffusion equation*, Comput. Math. Appl., 61 (2011), pp. 68–78.

[46] G. M. Mophou and G. M. N'Guérékata, *Optimal control of a fractional diffusion equation with state constraints*, Comput. Math. Appl., 62 (2011), pp. 1413–1426.

[47] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[48] R. H. Nochetto, E. Otárola, and A. J. Salgado, *A pde approach to fractional diffusion in general domains: a priori error analysis*, arXiv preprint arXiv:1302.0698, (2013).

[49] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[50] ——, *Constructive representation of functions in low-rank tensor formats*, Constr. Approx., 37 (2013), pp. 1–18.

[51] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Linear algebra for tensor problems*, Computing, 85 (2009), pp. 169–188.

[52] N. Özdemir and D. Avci, *Optimal control of a linear time-invariant space-time fractional diffusion process*, J. Vib. Control, 20 (2014).

[53] J. W. Pearson, M. Stoll, and A. J. Wathen, *Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 1126–1152.

[54] J. W. Pearson and A. J. Wathen, *A new approximation of the Schur complement in preconditioners for PDE-constrained optimization*, Numer. Linear Algebra Appl., 19 (2012), pp. 816–829.

[55] ——, *Fast iterative solvers for convection-diffusion control problems*, Electron. Trans. Numer. Anal., 40 (2013), pp. 294–310.

[56] 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.

[57] ——, *Fractional Differential Equations*, Academic Press, 1999.

[58] ——, *Matrix approach to discrete fractional calculus*, Fract. Calc. Appl. Anal., 3 (2000), pp. 359–386.

[59] I. Podlubny, A. Chechkin, T. Skovranek, Y. Chen, and B. Vinagre Jara, *Matrix approach to discrete fractional calculus II: partial fractional differential equations*, J. Comput. Phys., 228 (2009), pp. 3137–3153.

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

[61] M. R. Rapaić and Z. D. Jeličić, *Optimal control of a class of fractional heat diffusion systems*, Nonlinear Dynam., 62(1–2) (2010), pp. 39–51.

[62] J. A. Roberts, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Superfast solution of linear convolutional Volterra equations using QTT approximation*, J. Comput. Appl. Math., 260 (2014), pp. 434–448.

[63] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM J. Num. Anal., 51 (2013), pp. 1134–1162.

[64] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.

[65] S. Samko, A. Kilbas, and O. Marichev, *Fractional Integrals and Derivatives*, Gordon and Breach Science Publishers Yverdon, 1993.

[66] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Annals of Physics, 326 (2011), pp. 96–192.

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

[68] ——, *Computational methods for linear matrix equations*, tech. rep., Università di Bologna, March 2013.

[69] D. C. Sorensen and A. C. Antoulas, *The Sylvester equation and approximate balanced reduction*, Linear Algebra Appl., 351–352 (2002), pp. 671–700.

[70] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, to appear in SIAM J. Sci. Comp.

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

[72] F. Tröltzsch, *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, American Mathematical Society, 2010.

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

[74] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356.

[75] Q. Xu and J. S. Hesthaven, *Stable multi-domain spectral penalty methods for fractional partial differential equations*, J. Comput. Phys., 257 (2014), pp. 241–258.

[76] P. Yi-Fei, *Application of fractional differential approach to digital image processing*, Journal of Sichuan University (Engineering Science Edition), 39(3) (2007), pp. 124–132.

Figure 3: 3D test. y (top left), $\bar{y}$ (top right), u (bottom left) and p (bottom right) at the final time.