Fast Solution Methods for Convex Fractional Differential Equation Optimization

Spyridon Pougkakiotis*, John W. Pearson*, Santolo Leveque*, and Jacek Gondzio*

*School of Mathematics, University of Edinburgh

Technical Report ERGO 19–014

August 16, 2019

Abstract

In this paper, we present numerical methods suitable for solving convex Fractional Differential Equation (FDE) optimization problems, with potential box constraints on the state and control variables. First we derive powerful multilevel circulant preconditioners, which may be embedded within Krylov subspace methods, for solving equality constrained FDE optimization problems. We then develop an Alternating Direction Method of Multipliers (ADMM) framework, which uses preconditioned Krylov solvers for the resulting sub–problems. The latter allows us to tackle FDE optimization problems with box constraints, posed on space–time domains, that were previously out of the reach of state–of–the–art preconditioners. Discretized versions of FDEs involve large dense linear systems. In order to overcome this difficulty, we design a recursive linear algebra, which is based on the fast Fourier transform (FFT), and our proposed multilevel circulant preconditioners are suitable for approximating discretized FDEs of arbitrary dimension. Focusing on time–dependent 2–dimensional FDEs, we manage to keep the storage requirements linear, with respect to the grid size $N$, while ensuring an order $N \log N$ computational complexity per iteration of the Krylov solver. We implement the proposed method, and demonstrate its scalability, generality, and efficiency, through a series of experiments over different setups of the FDE optimization problem.

1 Introduction

Optimization problems with partial differential equations (PDEs) as constraints have received a great deal of attention within the applied mathematics and engineering communities, due in particular to their wide applicability across many fields of science. In addition to PDE constraints, one may also allow the use of fractional
derivatives in order to model processes that could not otherwise be modeled using integer derivatives. In fact, there is a wide and increasing use of fractional differential equations (FDEs) in the literature. Among other processes, FDEs have been used to model viscoelasticity (e.g. [32]), chaotic systems [53], turbulent flow, or anomalous diffusion (e.g. [6]). In particular, since the fractional operator is non–local, problems with non–local properties are usually well–modeled using FDEs (see [44], for an extended review).

Availability of closed form solutions for FDEs is rare, and hence various numerical schemes for solving them have been developed and analyzed in the literature (see [39, 40, 41] for finite difference, and [21, 15] for finite element methods). Most of the aforementioned numerical schemes typically produce dense matrices, making the solution or even the storage of FDE–constrained optimization problems extremely difficult for fine grids. Naturally, this behavior is even more severe in the case of multidimensional FDEs. In light of the previous, employing standard direct approaches for solving such problems requires $O(N^3)$ operations and $O(N^2)$ storage, where $N$ is the overall number of grid points. Iterative methods with general purpose preconditioners also suffer from similar issues.

Various specialized solution methods have been proposed in the literature, aiming at lowering the computational and storage cost of solving such problems. One popular and effective approach is to employ tensor product solvers. Such specialized methods have been proposed for solving high–dimensional FDE–constrained inverse problems with great success, even for very fine discretizations (see for example [17, 29] and the references therein). While these solvers are highly scalable (with respect to the grid size), to date they have been tailored solely to problems with specific cost functionals and without additional algebraic constraints. Another popular approach is based on the observation that multidimensional FDEs possess a multilevel Toeplitz structure. It is well known that such matrices can be very well approximated by multilevel circulant matrices (see for example [13, 12, 34]), which in turn can be inverted or applied to a vector in only $O(N \log N)$ operations using the fast Fourier transform (FFT) (e.g. [52]). The idea is to apply a Krylov subspace solver, supported by a circulant preconditioner, in order to solve the optimality conditions of the problem. One is able to redesign the underlying linear algebra, in order to achieve an $O(N \log N)$ iteration complexity for the Krylov solver, with $O(N)$ overall storage requirements (see for example [36, 35, 34]). While such solution methods are certainly more general, as compared to tensor product solvers, they remain rather sensitive in terms of the underlying structure. In particular, no such method has been proposed for the solution of more general FDE optimization problems, for instance those which include box constraints on the state and control variables. We highlight that a time–independent problem, with box constraints on the control, is studied in [19], and the authors attempt to solve it using a Limited–memory Broyden–Fletcher–Goldfarb–Shanno (L–BFGS) method.

In this paper we attempt to solve large–scale optimization problems, with FDE constraints posed over a space–time domain, for different choices of cost function-
als to be minimized, and possibly with additional box constraints imposed on the state and control variables. We propose a multilevel circulant preconditioner, suitable for approximating multilevel Toeplitz matrices, by extending the preconditioner proposed in [13]. We then present an optimization method, which combines an Alternating Direction Method of Multipliers (ADMM) with a bespoke iterative solver for the resulting matrix systems. We employ ADMM to separate the equality from the inequality constraints. As a consequence, it is possible to apply powerful circulant preconditioners to solve the equality constrained ADMM sub-problems. In particular, the method is matrix-free, that is for a general FDE optimization problem with inequality constraints, it requires only \(O(N)\) storage, and as expected, the iteration complexity of the Krylov solver is limited to \(O(N \log N)\) computations. In light of the previous, the proposed methodology is very general, and allows us to solve problems that have not been tackled in the literature. In particular, we are able to solve inverse problems with 2-dimensional time-dependent FDE constraints, and possible box constraints on the control as well as the state variables. The misfit, between the solution state and a desired state, is measured either in the \(L^2\) or in the \(H^1\)–norm. The solution of space–time FDE optimization problems measuring the \(H^1\)–norm misfit within the cost functional, or with additional state and/or control box constraints, has not to our knowledge been tackled using existing preconditioned iterative solvers.

This paper is structured as follows. In Section 2, we present the FDE–constrained optimization problem under consideration. Subsequently, in Section 3, a general multilevel circulant preconditioner is derived, suitable for approximating multilevel Toeplitz matrices, while demonstrating that such a preconditioner is effective for the problem at hand. In Section 4, we propose flexible and efficient solution methods for the aforementioned FDE–constrained optimization problem. Finally, in Section 5, we present some numerical results, and in Section 6, we derive our conclusions.

## 2 The FDE–constrained Optimization Model

In this section, we present the FDE–constrained optimization problem studied in this paper, and provide details as to the FDE discretization used. Then, we highlight some important properties of the resulting discretized matrices.

We define the *Caputo derivative* of a function \(f(t)\) defined on \(t \in [t_0, t_1]\), of real order \(\alpha\) such that \(n - 1 < \alpha < n\) with \(n \in \mathbb{N}\), as follows:

\[
\frac{C}{t_0} D_\alpha^t f(t) = \frac{1}{\Gamma(n-\alpha)} \int_{t_0}^t \frac{d^n f(s)}{ds^n} \frac{ds}{(t-s)^{\alpha-n+1}},
\]

assuming convergence of the above [16, 40, 44]. We also define the left–sided and right–sided *Riemann–Liouville derivatives* of a function \(f(x)\) defined on \(x \in [x_0, x_1]\),
of real order $\beta$ such that $n - 1 < \beta < n$ with $n \in \mathbb{N}$, as
\[
\begin{align*}
\frac{RL}{x_0 D_x^\beta} f(x) &= \frac{1}{\Gamma(n - \beta)} \frac{d^n}{dx^n} \int_{x_0}^x \frac{f(s) \, ds}{(x - s)^{\beta - n + 1}}, \\
\frac{RL}{x D_{x_1}^\beta} f(x) &= \frac{(-1)^n}{\Gamma(n - \beta)} \frac{d^n}{dx^n} \int_{x}^{x_1} \frac{f(s) \, ds}{(s - x)^{\beta - n + 1}},
\end{align*}
\]
respectively. From this, we define the symmetric Riesz derivative as follows [17, 44]:
\[
\frac{R}{x^2} D_x^\beta f(x) = \frac{1}{2} \left( \frac{RL}{x_0 D_x^\beta} f(x) + \frac{RL}{x D_{x_1}^\beta} f(x) \right).
\]
(2.1)

We highlight that Caputo derivatives are frequently used for discretization of FDEs in time, given initial conditions, with Riemann–Liouville derivatives correspondingly considered for spatial derivatives, given boundary conditions. We consider the minimization problem:
\[
\begin{align*}
\min_{y, u} & \ J(y, u) \\
\text{s.t.} & \ (D_t^\alpha y(x_1, x_2, t) + u(x_1, x_2, t) = g(x_1, x_2, t), \\
y_a \leq y \leq y_b, \ u_a \leq u \leq u_b,
\end{align*}
\]
(2.2)
where the fractional differential equation and additional algebraic constraints are given on the space–time domain $\Omega \times (0, T)$, where $\Omega \subset \mathbb{R}^2$ has boundary $\partial \Omega$, and the spatial coordinates are given by $x = [x_1, x_2]^T$. We note that $y_a$, $y_b$, $u_a$, and $u_b$ may take the form of constants, or functions in spatial and/or temporal variables. We also impose an initial condition $y(x, 0) = y_0(x)$ at $t = 0$, and the Dirichlet condition $y = 0$ on $\partial \Omega \times (0, T)$. We assume that the orders of differentiation satisfy $0 < \alpha < 1, 1 < \beta_1 < 2, 1 < \beta_2 < 2$.

The cost functional $J(y, u)$ measures the misfit between the state variable $y$ and a given desired state $\tilde{y}$ in some given norm, and also measures the ‘size’ of the control variable $u$. In this paper we consider two different cost functionals $J(y, u)$, one corresponding to $L^2$–norms measuring both the misfit between state and desired state, and the control variable:
\[
\begin{align*}
J(y, u) &= J_{L^2}(y, u) = \frac{1}{2} \int_0^T \int_{\Omega} (y - \tilde{y})^2 \, dx \, dt + \frac{\gamma}{2} \int_0^T \int_{\Omega} u^2 \, dx \, dt,
\end{align*}
\]
and the second measuring the misfit between state and desired state in the $H^1$-norm:
\[
\begin{align*}
J(y, u) &= J_{H^1}(y, u) = \frac{1}{2} \int_0^T \int_{\Omega} [\langle y - \tilde{y} \rangle^2 + |\nabla (y - \tilde{y})|^2] \, dx \, dt + \frac{\gamma}{2} \int_0^T \int_{\Omega} u^2 \, dx \, dt.
\end{align*}
\]
Here $\gamma > 0$ denotes a regularization parameter on the control variable. We note that other variants for $J(y, u)$ are possible, including measuring the control variable in the $H^1$–norm, as well as alternative weightings within the cost functionals. We also emphasize that it is perfectly reasonable to consider such problems involving FDEs in one or three spatial dimensions (or indeed higher dimensions), rather than in two dimensions as in (2.2), and the methodology in this paper could be readily
tailored to such problems.

Upon discretization, we consider the non-shifted Grünwald–Letnikov formula [17, 44, 47] to approximate the Caputo derivative in time:

\[
\frac{C}{t_0} D_t^\alpha y(t) \approx \frac{1}{\tau^\alpha} \sum_{k=0}^{n-1} g_k^\alpha y(t - k\tau),
\]

(2.3)

where \( g_k^\alpha = \frac{\Gamma(k - \alpha)}{\Gamma(-\alpha)(k + 1)} \) may be computed recursively via \( g_k^\alpha = (1 - \frac{\alpha + 1}{k}) g_{k-1}^\alpha, \ k = 1, 2, ..., \nu, \) with \( g_0^\alpha = 1 \) and \( \nu \in \mathbb{N} \). This leads to the Caputo derivative matrix for all grid points in the time variable:

\[
\mathcal{C}_\alpha = \frac{1}{\tau^\alpha} \begin{bmatrix}
g_0^\alpha & 0 & \cdots & \cdots & 0 \\
g_1^\alpha & g_0^\alpha & \cdots & \vdots \\
g_2^\alpha & g_1^\alpha & g_0^\alpha & \cdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
g_{\nu t - 1}^\alpha & \cdots & \cdots & g_2^\alpha & g_1^\alpha & g_0^\alpha 
\end{bmatrix},
\]

(2.4)

For the (left-sided) spatial derivative, we use the shifted Grünwald–Letnikov formula [39, 41, 44]:

\[
\frac{\text{RL}}{x_0} D_x^\beta y(x) \approx \frac{1}{h^\beta} \sum_{k=0}^{n} g_k^\beta y(x - (k - 1)h),
\]

(2.5)

leading to the matrix

\[
\mathcal{L}_{\beta,1} = \frac{1}{h^\beta} \begin{bmatrix}
g_1^\beta & g_0^\beta & 0 & \cdots & \cdots & 0 \\
g_2^\beta & g_1^\beta & g_0^\beta & \cdots & \vdots \\
g_3^\beta & \cdots & \cdots & \cdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
g_{\nu t}^\beta & \cdots & \cdots & g_2^\beta & g_1^\beta & g_0^\beta 
\end{bmatrix},
\]

whereby using the formula (2.1) leads to the following Riemann–Liouville derivative matrix for the symmetrized Riesz derivative:

\[
\mathcal{L}_{\beta} = \frac{1}{2} \left( \mathcal{L}_{\beta,1} + \mathcal{L}_{\beta,1}^T \right).
\]

(2.6)

Using all the previous definitions, we can write the discretized version of the FDE constraint within (2.2) as:

\[
Ay + u = g,
\]

(2.7)

where \( y, u, g \) represent the discretized variants of \( y, u, g \), and:

\[
A = \mathcal{C}_\alpha \otimes I_{n_{x_1} \times n_{x_2}} - I_n \otimes (\mathcal{L}_{\beta_1} \otimes I_{n_{x_2}} + I_{n_{x_1}} \otimes \mathcal{L}_{\beta_2}).
\]

(2.8)
We note at this point that FDE–constrained optimization problems, and preconditioners for the resulting matrix systems, have been considered previously (see for instance [14, 17, 29, 54]): in the majority of previous work the cost functional \( J_{L^2}(y,u) \) is considered, and no additional algebraic constraints on the state and control variables are imposed. Recently, preconditioners for FDE–constrained optimization were considered for problems with algebraic and sparsity constraints in the time-independent setting [19]. Two major contributions of this work are to present solvers for problems involving a more sophisticated cost functional \( J_{H^1}(y,u) \), and problems with additional constraints on \( y \) and \( u \), with fractional derivatives imposed in both space and time variables. These developments illustrate the rather broad applicability of the approach presented in this paper. For simplicity of exposition, in the rest of the paper we assume that \( h_{x_1} = h_{x_2} = h \), where \( h_{x_i} \) is the discretization step in the respective spatial direction, noting that the method readily generalizes to problems where this is not the case.

In the following proposition, we summarize some well–known properties of the fractional binomial coefficient sequence that arises above when constructing the matrices \( C^\alpha \) and \( L^\beta \) (see for example [28], page 397, or [40, 50]):

**Proposition 2.1.** Let \( 0 < \alpha < 1 \) and \( 1 < \beta < 2 \), with \( g^\alpha_k \), \( g^\beta_k \) as in (2.3), (2.5). Then, we have that:

\[
g^\alpha_0 > 0, \quad g^\alpha_k < 0, \quad \forall \quad k \geq 1,
\]

\[
\sum_{k=0}^{n_t} g^\alpha_k > 0, \quad \forall \quad n_t \geq 1, \quad (2.9)
\]

and

\[
g^\beta_0 = 1, \quad g^\beta_1 = -\beta, \quad g^\beta_2 > g^\beta_3 > \ldots > 0,
\]

\[
\sum_{k=0}^{\infty} g^\beta_k = 0, \quad \sum_{k=0}^{n} g^\beta_k < 0, \quad \forall \quad n \geq 1.
\]

Proposition 2.1 will become useful in showing that the preconditioner derived in the following section has some desirable properties.

### 3 Toeplitz Matrices – Circulant Preconditioners

In this section, we propose a multilevel circulant preconditioner, suitable for approximating multilevel Toeplitz matrices, and then examine the quality of such a preconditioner for the problem at hand. Firstly, let us define the relevant terminology that will be used in the rest of this paper.

**Definition 3.1.** Given a matrix \( T \in \mathbb{R}^{n \times n} \), we say that \( T \) is Toeplitz (or level–1 Toeplitz), if it is constant along all of its diagonals, i.e.:

\[
T = \begin{bmatrix}
t_0 & t_{-1} & \cdots & t_{-n+1} \\
t_1 & t_0 & \cdots & t_{-n+2} \\
\vdots & \vdots & \ddots & \vdots \\
t_{n-1} & t_{n-2} & \cdots & t_0
\end{bmatrix}.
\]
Similarly, we say that $T \in \mathbb{R}^{(n_1 \cdot n_2) \times (n_1 \cdot n_2)}$ is a level–2 Toeplitz matrix, if it consists of Toeplitz blocks that repeat in a Toeplitz manner, that is:

$$T = \begin{bmatrix} T_0 & T_{-1} & \cdots & T_{-n_2+1} \\ T_2 & T_0 & \cdots & T_{-n_2+2} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n_2-1} & T_{n_2-2} & \cdots & T_0 \end{bmatrix},$$

where $T_i \in \mathbb{R}^{n_1 \times n_1}$ is Toeplitz, for all $i \in \{-n_2+1, \ldots, n_2-1\}$. Given the previous, we can recursively define level–$k$ Toeplitz matrices, for any $k \geq 1$.

**Definition 3.2.** Given a matrix $C \in \mathbb{R}^{n \times n}$, we say that $C$ is circulant if its columns consist of cyclic permutations of its first column vector $c$, with offset equal to the column’s index, that is:

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

We define level–$k$ circulant matrices as in the Toeplitz case.

Toeplitz and multilevel Toeplitz matrices appear in numerous models of scientific computing problems. In particular, they are used to model the numerical solution of certain partial, integral, or fractional differential equations, problems in time series analysis, as well as in signal processing (see for example [2, 51, 45, 36], and the references therein). For example, computing the asymptotic spectral distribution of discretized matrices arising from numerical approximations of various continuous problems, can be achieved through generalized Toeplitz sequences, using the general theory that has been developed in [24] and [23].

An active area of research is that of solving a huge–scale systems of linear equations, $Ax = b$, where the matrix $A$ has some specific structure, such as Toeplitz, multilevel Toeplitz, or it can be written as a combination of Toeplitz and other structured matrices. While fast direct approaches for solving such systems have been proposed in the literature (see for example [5, 37]), they tend to be either unstable or problem specific. A more popular approach is to employ some iterative method to solve the system, assisted by an appropriately designed preconditioner, to ensure that the iterative method achieves fast convergence (as in [13, 12, 36, 35, 34, 8, 11, 9, 7, 49, 31, 10]).

Given a level–1 Toeplitz matrix $T \in \mathbb{R}^{n \times n}$, we employ the circulant approximation proposed for the first time in [12]. More specifically, we define the optimal circulant approximation of $T$, as the solution of the following optimization problem:

$$C_1(T) = \min_{C \in \mathcal{C}_n} \|C - T\|_F,$$
where $C_n$ is the set of all $n \times n$ circulant matrices, and $\| \cdot \|_F$ the Frobenius norm. It turns out that the previous optimization problem admits the following closed form solution:

$$c_i = \frac{(n - i) \cdot t_i + i \cdot t_{n+i}}{n}, \quad i \in \{0, \ldots, n - 1\}.$$ 

Other approximations are possible, however, we focus on the previously presented one. Nevertheless, the reader is referred to [11, 49, 10], for alternative level–1 circulant approximations. In the following theorem, we summarize some of the well-known properties of $C_1(T)$. The reader is referred to [13, 49] (and the references therein) for a detailed derivation of this result.

**Theorem 3.3.** Let $T$ be a Toeplitz matrix, with elements representing the Fourier coefficients of a generating function $f$. If these coefficients are absolutely summable, the eigenvalues of $[C_1(T)]^{-1}T$ are clustered around 1. Furthermore, if $T$ is symmetric and positive definite, then so is $C_1(T)$.

Let us now follow the developments in [13], to derive a level–2 circulant approximation of a level–2 Toeplitz matrix $T$. To do so, we firstly note that any circulant matrix can be diagonalized by the Fourier transform. More specifically, for any level–1 circulant matrix $C \in C_n$, we have that:

$$C = F_n^* \Lambda F_n,$$

where $F_n$ is the discrete Fourier transform of size $n$ and $\Lambda$ is a diagonal matrix containing the eigenvalues of $C$. In particular, $\Lambda$ can be computed efficiently as $\Lambda = F_n c$, where $c$ is the vector containing the first column of $C$. We proceed by taking the level–1 circulant approximation of each of the Toeplitz blocks of $T$. We then obtain the matrix:

$$\tilde{C}(T) = \begin{bmatrix}
C_1(T_0) & C_1(T_{-1}) & \cdots & C_1(T_{-n+1}) \\
C_1(T_1) & C_1(T_0) & \cdots & C_1(T_{-n+2}) \\
& & \ddots & \vdots \\
C_1(T_{n-1}) & C_1(T_{n-2}) & \cdots & C_1(T_0)
\end{bmatrix}.$$ 

Using the diagonalization property of circulant matrices, we can re–write the previous matrix as:

$$\tilde{C}(T) = (I_{n_2} \otimes F_n^*) \begin{bmatrix}
\Lambda_0 & \Lambda_{-1} & \cdots & \Lambda_{-n+1} \\
\Lambda_1 & \Lambda_0 & \cdots & \Lambda_{-n+2} \\
& & \ddots & \vdots \\
\Lambda_{n-1} & \Lambda_{n-2} & \cdots & \Lambda_0
\end{bmatrix} (I_{n_2} \otimes F_{n_1}),$$

where $\otimes$ denotes the Kronecker product, $F_n$ is the discrete Fourier transform of size $n$, and $\Lambda_i$ is a diagonal matrix containing the eigenvalues of the $i$–th circulant block. By defining the following permutation matrix:

$$P(v^1, \ldots, v^{n_2})^T = (v^1_1, \ldots, v^{n_2}_1, \ldots, v^1_{n_1}, \ldots, v^{n_2}_{n_1})^T,$$  

(3.1)
where \( v^i = (v^i_1, v^i_2, \ldots, v^i_{n_1})^T, \ i \in \{1, \ldots, n_2\}, \) we can re-write:

\[
\hat{C}(T) = (I_{n_2} \otimes F_{n_1}) P_T \begin{bmatrix}
D_0 & 0_{n_2 \times n_2} & \cdots & 0_{n_2 \times n_2} \\
0_{n_2 \times n_2} & D_1 & \cdots & 0_{n_2 \times n_2} \\
\vdots & \vdots & \ddots & \vdots \\
0_{n_2 \times n_2} & 0_{n_2 \times n_2} & \cdots & D_{n_1-1}
\end{bmatrix} P(I_{n_2} \otimes F_{n_1}).
\]

It is easy to see that, since \( T \) is level–2 Toeplitz, we are guaranteed that each of the blocks \( D_i \in \mathbb{R}^{n_2 \times n_2} \) is level–1 Toeplitz. For the final step in the construction of the level–2 circulant approximation of \( T \), we apply the level–1 circulant approximation to each matrix \( D_i \), and using once again the diagonalization property of circulant matrices, we can write the final level–2 approximation as:

\[
C_2(T) = (I_{n_2} \otimes F_{n_2}^*) P_T (I_{n_1} \otimes F_{n_2}^*) \Lambda (I_{n_1} \otimes F_{n_2}) P(I_{n_2} \otimes F_{n_1}),
\]

where \( \Lambda \in \mathbb{R}^{(n_1 \cdot n_2) \times (n_1 \cdot n_2)} \) is a diagonal matrix, containing the eigenvalues of \( C_2(T) \). It is now straightforward to define recursively the level–\( k \) circulant approximation \( C_k(T) \) of a level–\( k \) Toeplitz matrix \( T \). The method is summarized in Algorithm 1.

At this point, we have presented how to construct a level–\( k \) circulant preconditioner, for an arbitrary level–\( k \) Toeplitz matrix. The preconditioner can be computed efficiently in \( O(N \log N) \) operations, where \( N = n_1 \cdots n_k \). The storage requirements are \( O(N) \) since we only need to store the eigenvalues of the resulting matrix. This is because we work with multilevel circulant matrices with identical levels, and hence, all such matrices are diagonalized by the same matrix, say \( \hat{F} \). Given the previous, it is obvious that one can multiply two level–\( k \) circulant matrices in \( O(N) \) operations, simply by multiplying their respective diagonal eigenvalue matrices. Similarly, addition is performed in \( O(N) \) operations.

It is a well–known fact that any multilevel Toeplitz or circulant matrix can be multiplied by a vector expeditiously, that is, in \( O(N \log N) \) operations, using the fast Fourier transform (see for example [52]). The detailed algorithms for achieving this, for a general level–\( k \) Toeplitz or circulant matrix, are provided in the Appendices. We summarize the computational and storage cost of the presented recursive linear algebra in Table 1, where we assume that \( N = n_1 \cdots n_k \) and \( k = O(1) \).

### Table 1: Summary of Computational and Storage Complexity

| Structure       | Operation | Computations | Storage |
|-----------------|-----------|--------------|---------|
| Level–\( k \) Circulant | \( Cx \) | \( O(N \log N) \) | \( O(N) \) |
| Level–\( k \) Circulant | \( C^{-1}x \) | \( O(N \log N) \) | \( O(N) \) |
| Level–\( k \) Circulant | \( C^{(1)}C^{(2)} \) | \( O(N) \) | \( O(N) \) |
| Level–\( k \) Circulant | \( C^{(1)} + C^{(2)} \) | \( O(N) \) | \( O(N) \) |
| Level–\( k \) Toeplitz | \( Tx \) | \( O(N \log N) \) | \( O(N) \) |
| Level–\( k \) Circulant | Construct \( C_k(T) \) | \( O(N \log N) \) | \( O(N) \) |
Algorithm 1 Level-$k$ Circulant Preconditioner

Input: $T \in \mathbb{R}^{(n_1 \cdots n_k) \times (n_1 \cdots n_k)}$.

if $(k = 1)$ then
    Compute $C_1(T)$.
    return $\Lambda = F_{n_k} c$, where $c$ the first column of $C_1(T)$.
end if

- Compute the level-$(k - 1)$ circulant approximation for each of the $2 \cdot n_k - 1$ blocks of $T$, to obtain:

$$
\hat{C}(T) = (I_{n_k} \otimes \tilde{F}^*) \left[ \begin{array}{cccc}
A_{k-1}(T_0) & A_{k-1}(T_1) & \cdots & A_{k-1}(T_{n_k-1}) \\
A_{k-1}(T_1) & A_{k-1}(T_0) & \cdots & A_{k-1}(T_{n_k-2}) \\
\vdots & \vdots & \ddots & \vdots \\
A_{k-1}(T_{n_k-1}) & A_{k-1}(T_{n_k-2}) & \cdots & A_{k-1}(T_0) \\
\end{array} \right] (I_{n_k} \otimes \tilde{F}),
$$

where $\tilde{F} \in \mathbb{R}^{(n_1 \cdots n_{k-1}) \times (n_1 \cdots n_{k-1})}$ is an appropriate matrix, diagonalizing $C_{k-1}(T_i)$, $\forall i \in \{-n_k + 1, \ldots, n_k - 1\}$.

- Apply permutation $P$, defined in (3.1), by adjusting the dimensions given in its definition, to obtain:

$$
\hat{C}(T) = (I_{n_k} \otimes \tilde{F}^*) P^T \left[ \begin{array}{cccc}
D_0 & 0_{n_k \times n_k} & \cdots & 0_{n_k \times n_k} \\
0_{n_k \times n_k} & D_1 & \cdots & 0_{n_k \times n_k} \\
\vdots & \vdots & \ddots & \vdots \\
0_{n_k \times n_k} & 0_{n_k \times n_k} & \cdots & D_{n_1 \cdots n_{k-1} - 1} \\
\end{array} \right] P(I_{n_k} \otimes \tilde{F}),
$$

where $D_i \in \mathbb{R}^{n_k \times n_k}$ is level-1 Toeplitz, for all $i \in \{0, \ldots, (n_1 \cdots n_{k-1} - 1)\}$.
- Compute the level-1 circulant approximation for each $D_i$, to obtain the final preconditioner:

$$
C_k(T) = (I_{n_k} \otimes \tilde{F}^*) P^T (I_{n_1 \cdots n_{k-1}} \otimes F_{n_k}^*) \Lambda (I_{n_1 \cdots n_{k-1}} \otimes F_{n_k}) P(I_{n_k} \otimes \tilde{F}),
$$

where $\Lambda \in \mathbb{R}^{(n_1 \cdots n_k) \times (n_1 \cdots n_k)}$ is diagonal; i.e. containing the eigenvalues of $C_k(T)$.

return $\Lambda$


We now focus on the discretized FDE given in (2.7). By multiplying this equation on both sides by $\psi = \min\{\tau^\alpha, h^\beta_1, h^\beta_2\}$, we have:

$$
By + \psi u = \psi g,
$$

where $y, u, g$ represent the discretized variants of $y, u, g$, $B = \psi A$, with $A$ defined as in (2.8), and $\tau, h$ the time and spatial steps. We observe that matrix $A$ (and hence $B$) enjoys a level-3 Toeplitz structure. In particular, each block of $A$ ($B$) enjoys a quadrantal symmetric block Toeplitz structure (such matrices are analyzed for example in [8]).

In light of the previous, we can represent all non-repeating elements of the matrix
$B$, using the following notation:

$$[B]_{\kappa,\lambda,\mu} = t_{\kappa,\lambda,\mu}, \quad |\kappa| < n_t, \quad |\lambda| < n_{x_1}, \quad |\mu| < n_{x_2},$$

where $n_t$ are the discretization points in time, while $n_{x_1}, n_{x_2}$ the discretization points in the respective spatial direction. Let us also define the following scalars:

$$\xi_1 = \frac{\psi}{h^\beta_1}, \quad \xi_2 = \frac{\psi}{h^\beta_2}, \quad \xi_3 = \frac{\psi}{\tau^\alpha}, \quad (3.2)$$

which are obviously bounded above by 1. In order to analyze the effectiveness of the proposed level–3 circulant preconditioner for $B$, we prove that the generating function of matrix $B$ is absolutely summable. In turn, we use this to invoke other well-known results from the literature (see [34], for the case $k = 2$):

**Lemma 3.4.** The sequence $t_{\kappa,\lambda,\mu}$, produced by the generating function of the level–3 Toeplitz matrix $B$, is absolutely summable, i.e.:

$$\sum_{\kappa=-\infty}^{\infty} \sum_{\lambda=-\infty}^{\infty} \sum_{\mu=-\infty}^{\infty} |t_{\kappa,\lambda,\mu}| < \infty.$$

**Proof.** We note that the proof depends on the structure of the matrix $B$ (more specifically the structure of matrices $C_\alpha$ in (2.4) and $L_\beta$ in (2.6)). Let us re-write the matrix $B$ as:

$$B = \psi g_\alpha \otimes I_{n_{x_1} \times n_{x_2}} - I_{n_t} \otimes \tilde{L},$$

where

$$\tilde{L} = \psi (L_\beta_1 \otimes I_{n_{x_2}} + I_{n_{x_1}} \otimes L_\beta_2).$$

We start from the lowest level. Using the definition of $L_\beta$, we can see that:

$$\sum_{\mu=-\infty}^{\infty} |[L_\beta]|_{\mu} = \frac{2}{h^\beta} \left( \frac{1}{2} |g_1^{\beta}| + \frac{|g_2^{\beta} + g_0^{\beta}|}{2} + \frac{1}{2} \sum_{\mu=3}^{\infty} |g_\mu^{\beta}| \right)$$

$$\leq \frac{1}{h^\beta} \sum_{\mu=0}^{\infty} |g_\mu^{\beta}| = \frac{2}{h^\beta} \cdot |\beta|, \quad (3.3)$$

where we used Proposition 2.1. We can now analyze the second level, by examining $\tilde{L}$. By exploiting the structure of $\tilde{L}$ and using (3.2), (3.3), we have:

$$\sum_{\lambda=-\infty}^{\infty} \sum_{\mu=-\infty}^{\infty} |[\tilde{L}]_{\lambda,\mu}| = \psi \left( \sum_{\lambda=-\infty}^{\infty} |[L_\beta_1]|_{\lambda} + \sum_{\mu=-\infty}^{\infty} |[L_\beta_2]|_{\mu} \right)$$

$$\leq (2 \cdot \xi_1) \cdot |\beta_1| + (2 \cdot \xi_2) \cdot |\beta_2|. \quad (3.4)$$

Finally, using (3.3), (3.4), the definition of $g_\alpha$, as well as Proposition 2.1, we have:

$$\sum_{\kappa=-\infty}^{\infty} \sum_{\lambda=-\infty}^{\infty} \sum_{\mu=-\infty}^{\infty} |t_{\kappa,\lambda,\mu}| \leq \xi_3 \sum_{\kappa=-\infty}^{\infty} |g_0^\alpha| + \sum_{\lambda=-\infty}^{\infty} \sum_{\mu=-\infty}^{\infty} |[\tilde{L}]_{\lambda,\mu}|$$

$$\leq (2 \cdot \xi_3) \cdot g_0^\alpha + (2 \cdot \xi_1) \cdot |\beta_1| + (2 \cdot \xi_2) \cdot |\beta_2|. \quad (3.5)$$

Upon noting that the right–hand side is bounded, the proof is complete. \qed
Using the results presented in [34, 8, 35], we can derive the following theorem:

**Theorem 3.5.** Let $B = \psi A$. Then, for every $\epsilon > 0$, there exist constants $N_{x_2}, N_{x_1}, N_t$, such that for all $n_{x_2} > N_{x_2}, n_{x_1} > N_{x_1}, n_t > N_t$,

$$B - C_3(B) = U + V,$$

where

$$\text{rank}(U) = O(n_{x_2}n_{x_1} + n_{x_1}n_t + n_tn_{x_2}),$$

and

$$\|V\|_2 < \epsilon.$$

**Proof.** The proof follows the developments in [34, 8, 35]. For ease of presentation, we only provide an outline of it. In particular, by examining Algorithm 1, we have that:

$$\tilde{C}(B) = C_1(\psi C_\alpha) \otimes I_{n_{x_1}n_{x_2}} - I_{n_t} \otimes \psi(C_2(\mathcal{L}_{\beta_1} \otimes I_{n_{x_2}}) + C_2(I_{n_{x_1}} \otimes \mathcal{L}_{\beta_2})).$$

However, from [34, Lemma 2], we know that any level–2 Toeplitz matrix $T$, satisfying the absolutely summable condition proved in Lemma 3.4, also satisfies the following relation:

$$C_2(T) - T = U_1 + V_1, \quad \text{rank}(U_1) = O(n_1 + n_2), \quad \|V\| \leq \epsilon, \quad \forall \ n_1 \geq N_1(\epsilon), \ n_2 \geq N_2(\epsilon),$$

for suitably chosen constants $N_1(\epsilon), N_2(\epsilon)$. In light of the previous, we can see that for all $\epsilon > 0$, there exist $N_{x_2}, N_{x_1}, N_t$, such that for all $n_{x_2} > N_{x_2}, n_{x_1} > N_{x_1}, n_t > N_t$, we have:

$$\tilde{C}(B) - B = U_2 + V_2,$$

where $\text{rank}(U_2) = O(n_tn_{x_1} + n_tn_{x_2})$ and $\|V_2\| \leq \epsilon > 0$. The previous is derived by simple algebra manipulations. The reader is referred to [35] for a similar result. The authors there investigate a preconditioner that is based on the level–1 circulant approximation proposed in [10]. Finally, following the developments in [34, Lemma 2], one can show that:

$$C_3(B) - \tilde{C}(B) = U_3 + V_3,$$

where $\text{rank}(U_3) = O(n_{x_1}n_{x_2})$ and $\|V_3\| \leq \epsilon$. Combining all the previous workings completes the proof.

Following [35, Remark 4.1], assuming that $k = O(1)$, we can recursively extend the result of Theorem 3.5 to the level–$k$ case, using induction. In other words, the developments discussed in this paper can be extended trivially to higher dimensional FDEs. It is important to note here the result in [48], where it is proved that, in the general case, any multilevel circulant preconditioner for multilevel Toeplitz matrices is not a superlinear preconditioner. Superlinear preconditioners are important, in that they allow preconditioned conjugate gradient–like methods to converge in a constant number of iterations, independently of the size of the problem. In light of that, it comes as no surprise that a preconditioner like the one in Thereom 3.5 does not asymptotically capture all of the eigenvectors of the approximated matrix.
Nevertheless, the preconditioner approximates the matrices under consideration sufficiently well, while giving rise to a very low computational and storage cost. For a general theoretical framework suitable for interpreting and producing similar results as in Theorem 3.5, the reader is referred to [23] (and the references therein), where approximating classes of sequences are defined and studied.

4 Solution Methods

In this section, we present solution methods for the optimization problem (2.2), where a time–dependent, 2–dimensional FDE acts as a constraint. Applying a discretize–then–optimize approach, as in [17] for example, leads to the following discretized convex quadratic programming problem:

\[
\min_{y,u} \left( \frac{1}{2} (y - \bar{y})^T M_1 (y - \bar{y}) + \gamma \frac{1}{2} u^T M_2 u \right)
\]

\[\text{s.t. } By + \psi u = \psi g, \quad (4.1)\]

where \( B \in \mathbb{R}^{(n_x \times n_u \times n_t) \times (n_x \times n_u \times n_t)} \) is a level–3 Toeplitz matrix, \( M_1, M_2 \) are positive definite matrices, arising from the trapezoidal rule applied to the respective continuous integrals in the time variable, \( \bar{y} \in \mathbb{R}^{(n_x \times n_y \times n_t) \times 1} \) is the discretized version of the desired state \( \bar{y} \), and \( y_a, y_b, u_a, u_b \) arise from the discretization of functions \( y_a(x_1, x_2, t), y_b(x_1, x_2, t), u_a(x_1, x_2, t), u_b(x_1, x_2, t) \), that model lower and upper bound restrictions on the state and control variables. We note here that the theory (more specifically Theorem 3.5) holds only for the scaled version of the FDE constraints, and hence we have equivalently replaced (2.7) with the equality constraints in (4.1). Observe that we only solve problems in which the inequality constraints are simple box constraints. Nevertheless, it is worth mentioning that more general inequality constraints can be dealt with, using the proposed approach.

We highlight that an active research area in the PDE-constrained optimization community is to build numerical schemes such that the optimality conditions arising from the discretize–then–optimize method coincide with the optimize–then–discretize approach, where optimality conditions are first derived on the continuous level and then discretized.

4.1 Equality Constrained Model

Initially, we assume that the problem has no inequality constraints. In other words, all the variables are free, i.e. we want to solve:

\[
\min_{y,u} \left( \frac{1}{2} (y - \bar{y})^T M_1 (y - \bar{y}) + \gamma \frac{1}{2} u^T M_2 u \right)
\]

\[\text{s.t. } By + \psi u = \psi g. \quad (4.2)\]

In this case, we can form the Lagrangian of (4.2), as:

\[\mathcal{L}(y, u, p) = \frac{1}{2} (y - \bar{y})^T M_1 (y - \bar{y}) + \gamma \frac{1}{2} u^T M_2 u + p^T (By + \psi (u - g)).\]
Equating the gradient of the previous function to the zero vector, gives the first–order optimality conditions of (4.2):

\[
\begin{bmatrix}
M_1 & 0 & B^T \\
0 & \gamma M_2 & \psi I \\
B & \psi I & 0
\end{bmatrix}
\begin{bmatrix}
y \\
u \\
p
\end{bmatrix} =
\begin{bmatrix}
M_1 \bar{y} \\
0 \\
\psi g
\end{bmatrix}.
\] (4.3)

The saddle point matrix in (4.3) is invertible, since the matrix \([B \psi I]\) is of full rank, while \(M_1 \succ 0\) and \(M_2 \succ 0\) (see [17]). We are going to solve two variations of problem (4.2). In the first problem, we assume that \(J(y, u) = J_{L^2}(y, u)\) is minimized. In the second problem, we replace the previous objective function with \(J(y, u) = J_{H^1}(y, u)\). The choice of space \((L^2\) or \(H^1)\) reflects the properties of the solution. In particular, the problem in the \(H^1\) space is expected to provide less rapidly changing solutions, since it also takes into consideration the magnitude of the gradient of the solution. Both problems are \(L^2\)–regularized on the control variable. We note here that our developments can directly be extended to the case of \(H^1\)–regularization (for more on \(H^1\)–regularized problems, the reader is referred to [1, 38] and the references therein).

4.1.1 \(L^2\)–norm

Applying the trapezoidal rule to the objective \(J_{L^2}(y, u)\), we obtain:

\[
M = M_1 = M_2 = \begin{bmatrix}
I_{(n-1) \cdot n_{x_1} \cdot n_{x_2}} & 0 \\
0 & \frac{1}{2} I_{n_{x_1} \cdot n_{x_2}}
\end{bmatrix},
\]

which is applied to vectors arising from every time–step, apart from the initial time \(t = 0\). We explore two possible ways of solving (4.2) in this case. On the one hand, we can construct a positive definite preconditioner for the matrix in (4.3), and employ MINRES (see [43]) to solve the system. On the other hand, since the Hessian of the objective is positive definite and easily invertible, we are able to form the normal equations of (4.3) and solve them using Conjugate Gradient (CG) method (see [30]). As for MINRES, we use a preconditioned variant of CG (PCG).

**PCG**

Firstly, we have to decide on which variables to pivot, in order to form the normal equations. We choose here to pivot on \(u\) and \(p\), due to better numerical properties of the resulting normal equations. More specifically, from the third block equation of (4.3), we have:

\[
u = g - \frac{1}{\psi} B \bar{y}.
\]

Substituting the previous into the second block equation of (4.3), we can further obtain:

\[p = -\frac{\gamma}{\psi} M_2 g + \frac{\gamma}{\psi^2} M_2 B \bar{y},\]

and the resulting normal equations (after re–scaling) are:

\[(\psi^2 M_1 + \gamma B^T M_2 B) \bar{y} = \psi^2 M_1 \bar{y} + \psi \gamma B^T M_2 g.\] (4.4)
Now, $M(=M_1 = M_2)$ can be very well approximated by a circulant matrix, and we use the optimal circulant approximation presented in the previous section to do that, i.e. $C_1(M) \approx M$. To approximate $B$ and $B^T$, we employ the level–3 circulant approximation presented in the previous section. Then, the resulting preconditioner for the matrix of (4.4) is:

$$P_{CG}^{L^2} = \psi^2 C_1(M_1) + \gamma (C_3(B)^T C_1(M_2) C_3(B)). \quad (4.5)$$

**Theorem 4.1.** For every $\epsilon > 0$, there exist constants $N_{x_2}, N_{x_1}, N_t$, such that for all $n_{x_2} > N_{x_2}$, $n_{x_1} > N_{x_1}$, $n_t > N_t$,

$$\psi^2 M_1 + \gamma B^T M_2 B - \left( \psi^2 C_1(M_1) + \gamma (C_3(B)^T C_1(M_2) C_3(B)) \right) = U + V,$$

where

$$\text{rank}(U) = O(n_{x_2} n_{x_1} + n_{x_1} n_t + n_t n_{x_2}),$$

and

$$\|V\|_2 < \epsilon.$$

**Proof.** Using Theorem 3.5, the proof follows directly the developments in [34, Theorem 1].

Based on our discussion in the previous section, we see that the preconditioner in (4.5) can be formed in $O(N \log N)$ operations, where $N = n_{x_2} \cdot n_{x_1} \cdot n_t$ and requires $O(N)$ storage. Similarly, $(P_{CG}^{L^2})^{-1}$ can be applied to a vector in $O(N \log N)$ operations. On the other hand, the matrix related to the normal equations in (4.4) is never explicitly formed. Instead, it is treated as an operator (that is, we are only allowed to perform matrix–vector products with it). The latter is achieved in $O(N \log N)$ operations using Algorithm 10. We should mention that the preconditioner approximates a product of multilevel Toeplitz matrices by a product of their individual multilevel circulant approximations. This is a reasonable extension, which has been studied for example in [34]. For a formal theoretical investigation of approximating products of multilevel Toeplitz matrices, the reader is referred to [18] (e.g. Proposition 4).

**MINRES**

Alternatively, we can solve the block system (4.3) using preconditioned MINRES. While MINRES is suitable for solving saddle point systems, it requires that a positive definite preconditioner is used. To construct such a preconditioner, we first derive a saddle point preconditioner for (4.2), that is equivalent to (4.5). In particular, we have:

$$\tilde{P} = \begin{bmatrix}
C_1(M_1) & 0 & C_3(B)^T \\
0 & \gamma C_1(M_2) & \psi I \\
C_3(B) & \psi I & 0
\end{bmatrix}.$$
Let us now derive a positive definite preconditioner for \( \tilde{P} \). It is well-known (see \([42, 3, 20]\)) that a very good approximation of \( \tilde{P} \) reads as follows:

\[
P_{\text{MINRES}}^{L_2} = \begin{bmatrix}
C_1(M_1) & 0 & 0 \\
0 & \gamma C_1(M_2) & 0 \\
0 & 0 & (\psi^2(\gamma C_1(M_2))^{-1} + C_3(B)(C_1(M_1))^{-1}C_3(B)^T)
\end{bmatrix}.
\]

(4.6)

Based on the developments in \([33, 42]\), we know that \((P_{\text{MINRES}}^{L_2})^{-1}\tilde{P}\) will only have three distinct eigenvalues as long as \(\tilde{P}\) is invertible, namely, \(1, \frac{1}{2}(1 + \sqrt{5}), \frac{1}{2}(1 - \sqrt{5})\). Hence, we expect that \(P_{\text{CG}}^{L_2}\) will perform slightly better than \(P_{\text{MINRES}}^{L_2}\), however, they are both expected to be comparable in practice. As before, \(P_{\text{MINRES}}^{L_2}\) can be computed and applied expeditiously. A counterpart to Theorem 4.1 can also be given here for \(P_{\text{MINRES}}^{L_2}\), however, we omit it for ease of presentation.

### 4.1.2 \(H^1\)-norm

The only difference in this case, as compared to the \(L^2\)-norm model, is the resulting matrix \(M_1\), after the discretization. More specifically, we have that:

\[
M_1 = M_2 + \begin{bmatrix}
K & 0 & \cdots & 0 \\
0 & K & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & \frac{1}{2}K
\end{bmatrix},
\]

with \(M_2\) defined as in the previous paragraph, \(K = I_{n_x} \otimes K_{x_2} + K_{x_1} \otimes I_{n_{x_2}}\) and:

\[
K_{x_i} = \frac{1}{h^2} \begin{bmatrix}
2 & -1 & 0 & \cdots & 0 \\
-1 & 2 & -1 & \cdots & 0 \\
0 & -1 & 2 & \ddots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & -1 & 2
\end{bmatrix},
\]

where \(h\) is the distance between two adjacent interior points in the discretization of the respective spatial dimension. It is obvious that \(M_1\) is very well approximated by a level–3 Toeplitz matrix. In particular, each block of \(M_1\) is level–2 Toeplitz. Hence, it can be very well approximated by a level–3 circulant matrix, using the previously presented multilevel circulant preconditioner. As in the \(L^2\)-norm case, we investigate two solution methods. The first one employs PCG for solving \((4.4)\). The preconditioner in this case is constructed similarly to \((4.5)\), by replacing its first term by \(C_3(M_1)\). As before, the second method employs the preconditioned MINRES method for solving \((4.3)\). Again, we proceed similarly to the \(L^2\)-norm case, replacing \(C_1(M_1)\) by \(C_3(M_1)\) where it appears.

### 4.2 Box–Constrained Model

Let us now focus on solution methods for \((4.1)\). Notice that \((4.1)\) is a linearly constrained convex quadratic problem. There are various powerful optimization
methods for solving such problems. However, we are dealing with huge-scale matrix systems upon discretization. As a consequence, we cannot afford to store the resulting matrices without exploiting their structure. In light of that, most of the available optimization methods suitable for solving (4.1) would fail. For example, if we were to solve the problem using an Interior Point Method (IPM) (see for example [26]), we would have to introduce a logarithmic barrier function in the objective, making the Hessian of the resulting IPM sub-problems extremely ill-conditioned and badly approximated by a circulant matrix. In other words, the previously presented matrix-free preconditioners would no longer be effective and we would have to resort to general purpose preconditioners, that require the problem matrices to be fully stored. On the other hand, any Active-Set type of method (e.g. [22]) would produce sub-problems over a subset of the columns of the problem’s constraint matrix. As a consequence, the resulting sub-problems would not be guaranteed to possess any structure. Other methods, such as penalty (e.g. [1]) or augmented Lagrangian methods (e.g. [46]) would also face similar problems.

In order to overcome this issue, we propose the use of an Alternating Direction Method of Multipliers (ADMM) (see Section 5 in [4] and the references therein), which separates the equality from the inequality constraints, thus allowing us to exploit the structure found in the equality constraints. We should mention here that while ADMM allows us to retain the underlying structure of the problem, it comes at a cost. It is well-known (see e.g. [4]) that ADMM leads to relatively slow convergence and hence is not suitable for finding very accurate solutions. Nevertheless, a 4-digit accurate solution can be generally found in reasonable CPU time. On the other hand, due to the discretization error, we expect that such a solution should be satisfactory as long as the grid is not too fine, in which case the ADMM error should exceed the discretization error.

We begin by re-writing problem (4.1), after introducing some auxiliary variables $z_y, z_u ∈ \mathbb{R}^{n_y × n_x × n_t}$:

$$\begin{align*}
\min_{y,u} & \left( \frac{1}{2} (y - \bar{y})^T M_1 (y - \bar{y}) + \gamma u^T M_2 u \right) \\
\text{s.t.} & \quad By + \psi u = \psi g,
\end{align*}$$

(4.7)

Notice that (4.7) is equivalent to (4.1). The FDE constraints have been scaled by the constant $ψ$, which is required from the theory (see Theorem 3.5). By doing this, we ensure that the elements of the matrix $B$ are of order 1 ($O(1)$). As a result, the discretized control in the FDE constraints is multiplied by $ψ$. In ADMM such a scaling translates to a scaled step of the dual variables corresponding to the FDE constraints. In order to improve the balance of the algorithm, we multiply the constraints linking $u$ with its copy variables $z_u$ by $ψ$, thus scaling all the dual multipliers corresponding to these constraints. Next, we define the augmented
Lagrangian function corresponding to (4.7):

\begin{equation}
L_\delta(y,u,z_y,z_u,p,w_y,w_u) = \frac{1}{2}(y - \bar{y})^T M_1 (y - \bar{y}) + \frac{\gamma}{2} u^T M_2 u
+ p^T (B y + \psi(u - g)) + u_y^T (y - z_y) + \psi w_u^T (u - z_u)
+ \frac{1}{2\delta} \left( \|B y + \psi(u - g)\|^2_2 + \|y - z_y\|^2_2 + \psi^2 \|u - z_u\|^2_2 \right),
\end{equation}

(4.8)

where \(p, w_y, \) and \(w_u\) are the dual variables, each corresponding to the appropriate equality constraint of (4.7). An ADMM applied to model (4.7) is given in Algorithm 2. We omit specific details of the algorithm. The reader is referred to [4] for a proof of convergence of Algorithm 2, as well as a detailed overview of ADMM.

**Algorithm 2 (2–Block) ADMM**

**Input:** \(N = n_y \cdot n_u \cdot n_t\). Let \(y^0, u^0, z^0_y, z^0_u, p^0, w^0_y, w^0_u \in \mathbb{R}^N, \delta > 0.\)

for \((j = 0, 1, \ldots)\) do

\begin{align}
(y_{j+1}, u_{j+1}) &= \arg \min_{y,u} \{L_\delta(y,u,z_y^j,z_u^j,p^j,w_y^j,w_u^j)\} \tag{4.9a} \\
(z_y^{j+1}, z_u^{j+1}) &= \arg \min_{z_y \in [y_a, y_b], z_u \in [u_a, u_b]} \{L_\delta(y_{j+1}, u_{j+1}, z_y, z_u, p^j, w_y^j, w_u^j)\} \tag{4.9b} \\
p^{j+1} &= p^j + \frac{1}{\delta} (B y^{j+1} + \psi(u^{j+1} - g)) \tag{4.9c} \\
(w_y^{j+1}, w_u^{j+1}) &= \left( w_y^j + \frac{1}{\delta}(y^{j+1} - z_y^{j+1}), w_u^j + \frac{\psi}{\delta}(u^{j+1} - z_u^{j+1}) \right) \tag{4.9d}
\end{align}

end for

One can easily observe that the most challenging step of Algorithm 2, is that of solving (4.9a). The optimality conditions of (4.9a), at some iteration \(j\), read as follows:

\begin{equation}
\begin{bmatrix}
M_1 + \frac{1}{\delta}(B^T B + I) \\
\frac{\psi}{\delta} B \\
\frac{\psi}{\delta} B^T \\
\gamma M_2 + \frac{2\psi^2}{\delta} I
\end{bmatrix}
\begin{bmatrix}
y \\
u
\end{bmatrix} = \begin{bmatrix}
\eta_1 \\
\eta_2
\end{bmatrix},
\end{equation}

(4.10)

where

\[
\eta_1 = M_1 \bar{y} - B^T p^j - w_u^j + \frac{1}{\delta}(\psi \bar{B}^T g + z_y^j),
\]

and

\[
\eta_2 = -\psi p^j - \psi w_u^j + \frac{\psi^2}{\delta}(g + z_u^j).
\]

Solving (4.10) directly is not a good idea in our case, since its coefficient matrix is neither expected to be cheap to work with, nor too well–conditioned. Instead, we notice that we can equivalently change the order of steps (4.9b) and (4.9c) in Algorithm 2. This follows from the fact that the solution of (4.9b) does not depend
on \( p \). Then, we can merge steps (4.9a) and (4.9c) to obtain a more flexible system. More specifically, to take (4.9c) into account, we substitute

\[
p = p^j + \frac{1}{\delta}(By + \psi(u - g)),
\]

into (4.10), and the joint optimality conditions of (4.9a) and (4.9c) can then be written as follows:

\[
\begin{bmatrix}
M_1 + \frac{1}{\delta}I & 0 & B^T \\
0 & \gamma M_2 + \frac{\psi^2}{\delta}I & \psi I \\
B & \psi I & -\delta I
\end{bmatrix}
\begin{bmatrix}
y \\ u \\ p
\end{bmatrix}
= \begin{bmatrix}
M_1\tilde{y} - w^j_y + \frac{1}{\delta}z^j_y \\
-\psi w^j_u + \frac{\psi^2}{\delta} z^j_u \\
\psi g - \delta p^j
\end{bmatrix}. \tag{4.11}
\]

At this point, we have to decide how to solve (4.11). One can notice that the blocks of its coefficient matrix retain the Toeplitz–like structure we encountered in Section 4.1. Hence, all the previously proposed methodologies may also be applied here. In particular, depending on the instance under consideration (e.g. inequality constraints on both state and control, with \( L^2/H^1 \) norm, inequality constraints only on state variable, etc.), we can decide whether we want to solve (4.11) using PCG or MINRES, alongside a suitable (block) level–3 circulant preconditioner. In the case of PCG, we also have to decide on which variables to pivot in order to form the normal equations with the best numerical properties. We note that the developments in Section 4.1 can trivially be extended to solve systems like (4.11). For simplicity of exposition, we present here only one way of solving system (4.11), by forming the normal equations and then employing PCG to solve the resulting system. In particular, from the second block equation of this system, we have:

\[
u = \left(\frac{\psi^2}{\delta} \left(\gamma M_2 + \frac{\psi^2}{\delta} I\right) - \delta I\right)^{-1} \left(\psi p - \psi w^j_u + \frac{\psi^2}{\delta} z^j_u\right).
\]

Combining the previous with the third block equation of (4.11) gives:

\[
p = \left(\frac{\psi^2}{\delta} \left(\gamma M_2 + \frac{\psi^2}{\delta} I\right) - \delta I\right)^{-1} \left(B y + \phi\right),
\]

where

\[
\phi = -\psi g + \delta p^j - \psi \left(\gamma M_2 + \frac{\psi^2}{\delta} I\right)^{-1} \left(\psi w^j_u - \frac{\psi^2}{\delta} z^j_u\right);
\]

and the resulting normal equations read as follows:

\[
\left(M_1 + \frac{1}{\delta}I\right) + B^T \left(\frac{\psi^2}{\delta} \left(\gamma M_2 + \frac{\psi^2}{\delta} I\right) - \delta I\right)^{-1} B \right) y = M_1\tilde{y} - w^j_y + \frac{1}{\delta}z^j_y - B^T \left(\frac{\psi^2}{\delta} \left(\gamma M_2 + \frac{\psi^2}{\delta} I\right) - \delta I\right)^{-1} \phi.
\]

Finally, we should mention that line (4.9b) of Algorithm 2 admits a closed form solution. More specifically, we perform the optimization by ignoring the box constraints and then projecting the solution onto the box.
5 Numerical Results

At this point, we present the problem with which we test our proposed methods. The state and the control are defined on the domain $\Omega = [0, 1]^2$ and we consider $T = [0, 1]$. For some $n \in \mathbb{N}$, the discretized grid contains $n \times n \times n^2$ uniform points, in space and time, that is:

$$x_i^1 = ih, \ x_j^2 = jh, \ t^k = k\tau, \ i, j = 1, \ldots, n, \ k = 1, \ldots, n^2, \ h = \frac{1}{n+1}, \ \tau = h^2.$$

As a desired state function, we follow [17, Section 5.1] and set:

$$\bar{y}(x_1, x_2, t) = 10 \cos(10x_1) \sin(x_1x_2),$$

with homogeneous boundary and initial conditions. Throughout this section, we employ the convention that $n_{x_1} = n_{x_2} = n_t = n_{x_1}^2$, and we only present the overall size of the discretized state vector, that is $N = n_{x_1} \cdot n_{x_2} \cdot n_t = n_{x_1}^4$. As an indicator of convergence of the numerical method, we apply the trapezoidal rule to roughly approximate the discrepancy between the solution for the state and the desired state on the discrete level, in the space under consideration, i.e.:

$$\varepsilon L^2(y - \bar{y}) \approx ||y - \bar{y}||_{L^2},$$

or

$$\varepsilon H^1(y - \bar{y}) \approx ||y - \bar{y}||_{H^1}.$$

We should note that the previous measures approximate the misfit between the state and the desired state of the continuous problem, and hence they are not expected to converge to zero. However, as observed in [17], due to the Dirichlet boundary conditions, we consider only the inner grid points of the discrete model. Hence, one can actually approach $\bar{y}$, as $\gamma \to 0$. On the other hand, much of the activity, for our chosen desired state, occurs near the boundary, and hence an increase in the grid size is expected to result in slight increase in the approximate discrepancy measures.

It is worth mentioning that while we present a recursive linear algebra in Algorithms 3–10, the implementation is focused on the level–3 case for efficiency. The experiments are conducted on a PC with a 2.2 GHz Intel (hexa–) core i7 processor, run under the Windows 10 operating system. The code is written in Matlab R2019a.

5.1 Equality Constrained Model

Firstly, we provide two solution methods for (4.2). The first one is using the Matlab function `pcg`, in order to solve system (4.4). The second one employs `minres`, and solves system (4.3). The application of preconditioners (4.5) and (4.6), respectively, as well as the procedures performing the matrix–vector product of the coefficient matrices, are passed as function handles to the iterative solvers. The iterative procedures are terminated after a tolerance $\epsilon = 10^{-6}$ is reached.
$L^2$–norm misfit minimization:

In this case, we have $J(y,u) = J_{L^2}(y,u)$. Initially, we perform some tests to examine the performance of the two methods, with respect to the grid size. For that, we fix the regularization parameter to $\gamma = 10^{-4}$, and the fractional derivative orders to $\alpha = 0.5$, $\beta_1 = \beta_2 = 1.5$. As stated before, we set $N = n_x \cdot n_{x_2} \cdot n_t$. In Table 2, we summarize the performance of the two solution methods, for varying values of $N$.

From Table 2, one can observe that both PCG using (4.5) as a preconditioner, and MINRES using (4.6), converge very fast, even for fine discretizations. While both methods are affected by the grid size (as we anticipated from the theory in Section 3), the increase in the number of iterations is reasonable. As expected, MINRES requires around twice as many iterations to converge. Nevertheless, MINRES can allow one to solve more general FDE–constrained optimization problems, making it useful to highlight its performance. In the last example the swap memory is activated, due to insufficient physical memory, making the time required to solve it disproportionately large. Finally, the memory requirements become an issue for MINRES only when the state variable is of size $N = 40,960,000$. Given that the experiments are conducted on a personal computer, it becomes obvious that employing matrix–free recursive linear algebra allows us to solve significantly larger problems, as compared to conventional iterative solvers, that usually require the storage of the Cholesky factors of the preconditioner.

Next, we fix $N = 32^4$, $\alpha = 0.5$, and $\beta_1 = \beta_2 = 1.5$. In Table 3, we summarize the runs of the previous methods, for varying values of the regularization parameter $\gamma$. We can observe that both methods can significantly slow down (or accelerate), depending on the regularization parameter. Nevertheless, even for large values of the regularization parameter, the methods achieve convergence in a very reasonable number of iterations, and hence in reasonable CPU time. As state before, one can actually approach $\bar{y}$, as $\gamma \to 0$. This behavior is confirmed in Table 3, since for small values of $\gamma$, we compute a state which is very close to $\bar{y}$, while the Schur complement of the optimality conditions is approaching a scaled identity matrix (allowing for rapid convergence of the iterative methods).

1† means that the solver run out of memory.

---

**Table 2:** Equality constrained problem: $L^2$–norm misfit with varying grid.

| $N$  | $\ell^2(y - \bar{y})$ | PCG | MINRES |
|------|-----------------------|-----|--------|
|      |                       | Time (s) | Iterations | Time (s) | Iterations |
| $8^4$ | $3.79 \times 10^{-1}$ | 0.22 | 11 | 0.53 | 23 |
| $16^4$ | $5.51 \times 10^{-1}$ | 1.97 | 16 | 3.77 | 33 |
| $32^4$ | $6.79 \times 10^{-1}$ | 42.03 | 28 | 82.20 | 55 |
| $50^4$ | $7.29 \times 10^{-1}$ | 343.06 | 42 | 672.75 | 80 |
| $64^4$ | $7.48 \times 10^{-1}$ | 1045.66 | 53 | 2900.70 | 98 |
| $80^4$ | $7.61 \times 10^{-1}$ | 5807.62 | 62 | [††] | [††] |
Table 3: Equality constrained problem: $L^2$–norm misfit with varying regularization.

| $\gamma$ | $\varepsilon^{L^2}(y - \bar{y})$ | PCG | MINRES |
|----------|----------------------------------|-----|--------|
|         |                                  | Time (s) | Iterations | Time (s) | Iterations |
| $10^0$  | $2.31 \times 10^0$               | 127.37 | 91       | 248.43 | 172       |
| $10^{-1}$ | $2.30 \times 10^0$               | 118.47 | 85       | 239.74 | 166       |
| $10^{-2}$ | $2.16 \times 10^0$               | 39.41  | 63       | 191.31 | 132       |
| $10^{-4}$ | $6.79 \times 10^{-1}$            | 42.03  | 28       | 82.20  | 55        |
| $10^{-6}$ | $4.57 \times 10^{-2}$            | 16.01  | 8        | 29.78  | 18        |
| $10^{-8}$ | $5.33 \times 10^{-4}$            | 10.74  | 4        | 18.92  | 10        |

Finally, in Tables 4 and 5, we present the runs of the methods, for varying fractional derivative orders. In particular, in Table 4, we fix $N = 32^4$, $\gamma = 10^{-4}$, $\beta_1 = \beta_2 = 1.5$, and vary the time fractional order $\alpha$. Similarly, in Table 5, we fix $\alpha = 0.5$, and vary the space fractional order $\beta = \beta_1 = \beta_2$. Obviously, changing $\alpha$ affects the condition number of $C_\alpha$ in (2.4), while $\beta$ similarly affects the condition number of $L_\beta$ in (2.6). As a result, we can see that different values of the fractional orders, can result in different number of iterations for convergence of the iterative methods. However, due to the proposed preconditioning strategy, we can see that the performance difference, for the preconditioned methods, is not crucial.

Table 4: Equality constrained problem: $L^2$–norm misfit with varying time fractional order.

| $\alpha$ | $\varepsilon^{L^2}(y - \bar{y})$ | PCG | MINRES |
|----------|----------------------------------|-----|--------|
|         |                                  | Time (s) | Iterations | Time (s) | Iterations |
| 0.1     | $6.78 \times 10^{-1}$            | 39.99 | 26       | 75.93 | 50        |
| 0.3     | $6.79 \times 10^{-1}$            | 41.17 | 27       | 78.82 | 53        |
| 0.5     | $6.79 \times 10^{-1}$            | 42.03 | 28       | 82.20 | 55        |
| 0.7     | $6.80 \times 10^{-1}$            | 44.09 | 29       | 85.57 | 58        |
| 0.9     | $6.87 \times 10^{-1}$            | 53.46 | 36       | 104.09 | 71        |

Table 5: Equality constrained problem: $L^2$–norm misfit with varying space fractional order.

| $\beta$ | $\varepsilon^{L^2}(y - \bar{y})$ | PCG | MINRES |
|---------|----------------------------------|-----|--------|
|         |                                  | Time (s) | Iterations | Time (s) | Iterations |
| 1.1     | $1.21 \times 10^{-1}$            | 21.22 | 12       | 38.21 | 24        |
| 1.3     | $3.35 \times 10^{-1}$            | 28.06 | 17       | 50.77 | 33        |
| 1.5     | $6.79 \times 10^{-1}$            | 42.03 | 28       | 82.20 | 55        |
| 1.7     | $1.15 \times 10^0$               | 67.61 | 47       | 131.14 | 90        |
| 1.9     | $1.62 \times 10^0$               | 102.09 | 74       | 206.14 | 142       |
**$H^1$–norm misfit minimization:**

In this case, we have $J(y, u) = J_{H^1}(y, u)$. As before, we perform some tests to examine the performance of the two iterative methods, with respect to the grid size. For that, we fix $\gamma = 10^{-4}$, $\alpha = 0.5$, $\beta_1 = \beta_2 = 1.5$, and summarize the results in Table 6. We can draw similar observations as in the case $J(y, u) = J_{L^2}(y, u)$. In particular, neither of the methods is significantly affected by the increase of the grid size. On the other hand, the new preconditioned systems are more robust with respect to the regularization parameter $\gamma$. In particular, we fix $N = 32^4$, $\alpha = 0.5$, $\beta_1 = \beta_2 = 1.5$, and summarize the runs of the methods, for varying values of the regularization parameter, in Table 7.

**Table 6:** Equality constrained problem: $H^1$–norm misfit with varying grid.

| $N$  | $\varepsilon_{H^1}(y - \bar{y})$ | PCG | MINRES |
|------|----------------------------------|-----|--------|
|      |                                  |     |        |
| $8^4$ | $4.38 \times 10^{-3}$            | 0.45| 16     |
| $16^4$ | $5.89 \times 10^{-3}$           | 3.77| 23     |
| $32^4$ | $1.45 \times 10^{-2}$           | 64.20| 29    |
| $50^4$ | $3.76 \times 10^{-2}$           | 475.03| 39   |
| $64^4$ | $6.95 \times 10^{-2}$           | 1397.76| 46  |

**Table 7:** Equality constrained problem: $H^1$–norm misfit with varying regularization.

| $\gamma$ | $\varepsilon_{H^1}(y - \bar{y})$ | PCG | MINRES |
|----------|----------------------------------|-----|--------|
|          |                                  |     |        |
| $10^0$   | $1.52 \times 10^3$              | 154.30| 75    |
| $10^{-1}$| $6.76 \times 10^2$             | 109.03| 52    |
| $10^{-2}$| $5.95 \times 10^1$             | 79.59| 37    |
| $10^{-4}$| $1.44 \times 10^{-2}$         | 64.20| 29    |
| $10^{-6}$| $4.91 \times 10^{-5}$         | 62.50| 28    |
| $10^{-8}$| $3.63 \times 10^{-5}$         | 60.51| 27    |

Finally in Tables 8 and 9, we fix $N = 32^4$, $\gamma = 10^{-4}$, and present the runs of the methods for varying fractional derivative orders. In this case, we can observe that change in the fractional orders $\alpha$ and $\beta$ has little effect in the conditioning of the system. This was expected, due to the presence of the matrix $M_1$ arising from the discretization of the $H^1$–norm. Note that the latter matrix is significantly more ill-conditioned, compared to the discretized version of the $L^2$–norm. As a consequence, the conditioning of the system is not so strongly dependent on the fractional orders $\alpha$, $\beta$. Nevertheless, we present the following experiments for completeness.
Table 8: Equality constrained problem: $H^1$–norm misfit with varying time fractional order.

| $\alpha$ | $\varepsilon^2 (y - \bar{y})$ | PCG | MINRES |
|----------|-----------------|------|---------|
|          | Time (s) | Iterations | Time (s) | Iterations |
| 0.1      | $1.44 \times 10^{-2}$ | 62.29 | 28 | 100.45 | 45 |
| 0.3      | $1.45 \times 10^{-2}$ | 64.68 | 29 | 100.70 | 45 |
| 0.5      | $1.45 \times 10^{-2}$ | 64.20 | 29 | 102.92 | 46 |
| 0.7      | $1.47 \times 10^{-2}$ | 66.79 | 30 | 106.56 | 48 |
| 0.9      | $3.33 \times 10^{-2}$ | 70.26 | 32 | 133.96 | 61 |

Table 9: Equality constrained problem: $H^1$–norm misfit with varying space fractional order.

| $\beta$ | $\varepsilon^2 (y - \bar{y})$ | PCG | MINRES |
|---------|-----------------|------|---------|
|         | Time (s) | Iterations | Time (s) | Iterations |
| 1.1     | $1.52 \times 10^{-4}$ | 64.34 | 29 | 92.17 | 41 |
| 1.3     | $1.04 \times 10^{-3}$ | 64.39 | 29 | 92.26 | 41 |
| 1.5     | $1.45 \times 10^{-2}$ | 64.20 | 29 | 102.92 | 46 |
| 1.7     | $2.94 \times 10^{-1}$ | 67.43 | 30 | 115.76 | 52 |
| 1.9     | $5.80 \times 10^0$ | 68.41 | 31 | 132.34 | 60 |

5.2 Box–Constrained Model

Let us now focus our attention on problem (4.1). We equivalently implement a 2–Block ADMM for solving (4.7). The implementation follows exactly the developments in Section 4, by merging steps (4.9a) and (4.9c) in Algorithm 2. In order to solve the resulting system (4.11), we can incorporate the previously presented iterative procedures, by altering the preconditioners accordingly. For ease of presentation, we solve (4.11) using pcg. We note that while various potential acceleration strategies for ADMMs have been studied in the literature (see for example [4, 25]), the focus of the paper is to illustrate the viability of the proposed approach, and hence the simplest possible ADMM scheme is adopted. The termination criteria of the ADMM are summarized as follows:

$$
\left( \|By + \psi(u - g)\|_\infty \leq 10^{-4} \right) \land \left( \|y - z_y\|_\infty \leq 10^{-4} \right) \land \left( \|u - z_u\|_\infty \leq 10^{-4} \right).
$$

In order to avoid unnecessary computations, we have decided not to require a specific tolerance for the dual infeasibility. Instead, we report the dual infeasibility at the accepted optimal point. For simplicity of exposition, for the rest of this section we assume that $J(y, u) = J_{L^2}(y, u)$. The Krylov solver tolerance is set dynamically, based on the accuracy attained at the respective ADMM iteration. Hence, we present the average number of inner iterations in the results to follow. In the rest of this section, we fix the regularization parameter $\gamma = 10^{-4}$, and the fractional derivative orders $\alpha = 0.5$, $\beta_1 = \beta_2 = 1.5$. In the following Tables, we employ the convention...
that the discretized restricting functions are of the form \( y_b = -y_a = c \cdot e_N \) (or \( u_b = -u_a = c \cdot e_N \)), where \( e_N \) is the \( N \)-dimensional vector of ones and \( c \in \mathbb{R}^+ \). Hence, we present only the value of the entries of \( y_a \) (or \( u_a \) respectively).

**Box constraints on the state \( y \):**
Initially, we focus on the case where the state variable is required to stay in a box, while the control is free, that is \( y_a \leq y \leq y_b \), \( -\infty \leq u \leq \infty \). Obviously, using similar arguments as in [17, 19], we can see that an optimal solution in this case is guaranteed to exist. We fix the ADMM penalty parameter \( \delta = 0 \). 4 and perform some experiments with the method for varying grid size, as well as varying inequality bounds on the state \( y \). The results are summarized in Table 10.

**Table 10:** Box–Constrained problem: Inequalities on the state.

| \( N = 8^4 \) | \( y_a \) | \( \varepsilon^{H^1}(y - \bar{y}) \) | Dual Inf. | Iterations | Time (s) |
|---|---|---|---|---|---|
| | \(-5\) | \(3.79 \times 10^{-1} \)\(^2\) | \(2.46 \times 10^{-3}\) | 4 | 28 | 2.70 |
| | \(-4\) | \(4.32 \times 10^{-1}\) | \(1.88 \times 10^{-3}\) | 4 | 34 | 3.49 |
| | \(-3\) | \(6.95 \times 10^{-1}\) | \(2.69 \times 10^{-3}\) | 4 | 31 | 3.15 |
| | \(-1\) | \(1.53 \times 10^0\) | \(2.41 \times 10^{-3}\) | 4 | 25 | 2.63 |

| \( N = 16^4 \) | \( y_a \) | \( \varepsilon^{H^1}(y - \bar{y}) \) | Dual Inf. | Iterations | Time (s) |
|---|---|---|---|---|---|
| | \(-5\) | \(5.51 \times 10^{-1} \)\(^(*)\) | \(3.54 \times 10^{-4}\) | 6 | 42 | 38.36 |
| | \(-4\) | \(6.43 \times 10^{-1}\) | \(7.98 \times 10^{-4}\) | 6 | 37 | 32.71 |
| | \(-3\) | \(8.58 \times 10^{-1}\) | \(8.70 \times 10^{-4}\) | 6 | 35 | 29.19 |
| | \(-1\) | \(1.66 \times 10^0\) | \(7.04 \times 10^{-4}\) | 6 | 30 | 25.42 |

| \( N = 32^4 \) | \( y_a \) | \( \varepsilon^{H^1}(y - \bar{y}) \) | Dual Inf. | Iterations | Time (s) |
|---|---|---|---|---|---|
| | \(-6\) | \(6.79 \times 10^{-1} \)\(^(*)\) | \(3.22 \times 10^{-3}\) | 6 | 38 | 451.91 |
| | \(-5\) | \(7.06 \times 10^{-1}\) | \(4.31 \times 10^{-3}\) | 6 | 64 | 761.03 |
| | \(-4\) | \(8.05 \times 10^{-1}\) | \(2.21 \times 10^{-3}\) | 7 | 70 | 822.34 |
| | \(-1\) | \(1.73 \times 10^0\) | \(5.7 \times 10^{-3}\) | 7 | 76 | 932.49 |

**Box constraints on the control \( u \):**
In this case, we have \( -\infty \leq u \leq \infty \). Again, it is straightforward to show that such a problem admits an optimal solution (see [19]). It is important to note here, that since we introduce box constraints on the control \( u \), the linear system of the ADMM sub–problem (4.11) has a conditioning independent of the regularization parameter \( \gamma \). We fix the ADMM penalty parameter \( \delta = 0.4 \), and perform some experiments with the method for varying grid size, as well as varying inequality bounds on the control \( u \). The results are summarized in Table 11.

**Box constraints on all the variables:**
Finally, we consider the case where \( y_a \leq y \leq y_b \), \( u_a \leq u \leq u_b \). In general, in this

\(^2\)\(^(*)\) means that the solution coincides with the respective equality constrained solution.
Table 11: Box-Constrained problem: Inequalities on the control.

| $u_a$  | $\varepsilon^{H^1}(y - \bar{y})$ | Dual Inf. | Iter. | Iter. | Time (s) |
|--------|---------------------------------|-----------|-------|-------|----------|
|        |                                 |           | PCG   | ADMM  |          |
| $N = 8^4$ |                                |           |       |       |          |
| -300   | $3.79 \times 10^{-1}$ (⋆)      | $1.68 \times 10^{-5}$ | 9     | 49    | 8.62     |
| -200   | $4.77 \times 10^{-1}$          | $3.84 \times 10^{-5}$ | 9     | 42    | 6.75     |
| -100   | $9.29 \times 10^{-1}$          | $8.32 \times 10^{-5}$ | 8     | 37    | 5.58     |
| -50    | $1.38 \times 10^{0}$           | $4.34 \times 10^{-5}$ | 8     | 39    | 5.72     |
| $N = 16^4$ |                                |           |       |       |          |
| -400   | $5.51 \times 10^{-1}$ (⋆)      | $1.02 \times 10^{-3}$ | 13    | 41    | 61.23    |
| -200   | $6.92 \times 10^{-1}$          | $8.34 \times 10^{-4}$ | 12    | 35    | 52.12    |
| -150   | $8.39 \times 10^{-1}$          | $4.04 \times 10^{-4}$ | 13    | 37    | 54.36    |
| -100   | $1.09 \times 10^{0}$           | $5.67 \times 10^{-4}$ | 11    | 31    | 42.35    |
| $N = 32^4$ |                                |           |       |       |          |
| -500   | $6.79 \times 10^{-1}$ (⋆)      | $3.67 \times 10^{-3}$ (⋆) | 10    | 19    | 349.16   |
| -300   | $6.90 \times 10^{-1}$          | $1.99 \times 10^{-3}$ | 12    | 39    | 780.51   |
| -200   | $8.00 \times 10^{-1}$          | $1.51 \times 10^{-3}$ | 12    | 67    | 1321.49  |
| -100   | $1.17 \times 10^{0}$           | $2.32 \times 10^{-3}$ | 11    | 125   | 2435.41  |

In case one is not able to conclude that the problem admits an optimal solution. We mainly present this case for completeness, and to avoid infeasibility we use rather conservative bounds for which we know that an optimal solution exists. We fix the ADMM penalty parameter $\delta = 0.4$, and present the results in Table 12.

Table 12: Box-Constrained problem: Inequalities on all variables.

| $y_a$ | $u_a$  | $\varepsilon^{H^1}(y - \bar{y})$ | Dual Inf. | Iter. | Iter. | Time (s) |
|-------|--------|---------------------------------|-----------|-------|-------|----------|
|       |        |                                 |           | PCG   | ADMM  |          |
| $N = 8^4$ |                                |           |       |       |       |          |
| -5    | -300   | $3.79 \times 10^{-1}$ (⋆)      | $3.29 \times 10^{-4}$ | 7     | 51    | 7.21     |
| -5    | -100   | $9.29 \times 10^{-1}$          | $2.13 \times 10^{-4}$ | 7     | 39    | 5.47     |
| -3    | -250   | $6.95 \times 10^{-1}$          | $2.76 \times 10^{-5}$ | 9     | 72    | 11.96    |
| -1    | -300   | $1.53 \times 10^{0}$           | $1.05 \times 10^{-4}$ | 9     | 70    | 11.16    |
| $N = 16^4$ |                                |           |       |       |       |          |
| -5    | -400   | $5.51 \times 10^{-1}$ (⋆)      | $6.41 \times 10^{-3}$ | 9     | 44    | 49.17    |
| -5    | -200   | $6.92 \times 10^{-1}$          | $4.42 \times 10^{-3}$ | 8     | 42    | 44.65    |
| -2    | -260   | $1.21 \times 10^{0}$           | $1.15 \times 10^{-4}$ | 11    | 76    | 103.53   |
| -1    | -400   | $1.66 \times 10^{0}$           | $1.96 \times 10^{-4}$ | 12    | 66    | 90.58    |
| $N = 32^4$ |                                |           |       |       |       |          |
| -6    | -500   | $6.79 \times 10^{-1}$ (⋆)      | $7.24 \times 10^{-4}$ | 15    | 20    | 470.81   |
| -6    | -200   | $8.00 \times 10^{-1}$          | $1.51 \times 10^{-3}$ | 14    | 67    | 1507.50  |
| -4    | -350   | $8.05 \times 10^{-1}$          | $1.30 \times 10^{-3}$ | 10    | 97    | 1679.87  |
| -1    | -500   | $1.73 \times 10^{0}$           | $3.94 \times 10^{-5}$ | 13    | 117   | 2499.89  |
We should note at this point that the current implementation is a proof of concept. While the method is sufficiently efficient, one should be able to accelerate it significantly. In particular, since we employ a linear algebra based on the FFT, the Krylov subspace methods can be cast directly in the GPU. There are various commercially available GPU implementations of the \texttt{fft} routine, and for large enough problems (assuming sufficient GPU memory), one should be able to gain a significant speedup. On the other hand, throughout the implementation, we only use dense vectors. There are cases where a sparse Fourier transform (see [27]) could deliver similar results, in significantly less time and requiring significantly less memory. Unfortunately, the SFFT library is not yet commercially available. Nevertheless, if efficiency is an issue, one should also consider this option.

6 Conclusions

We have presented numerical methods for solving FDE–constrained optimization problems, with potential box constraints on the state and control variables. In particular, we use the Grünwald–Letnikov finite difference method, and by employing a discretize–then–optimize approach, we solve the resulting problem in the discretized variables. If no box constraints are present, we are required to solve a high–dimensional linear system, representing the optimality conditions of the problem. For that, we design a recursive linear algebra based on FFTs, using which we solve the system through a Krylov subspace solver alongside a multilevel circulant preconditioner. In the presence of box constraints, we reformulate the problem, and additionally employ an ADMM along with the bespoke preconditioners. The ADMM sub–problems retain the structure of the problem, allowing us to use the aforementioned iterative solvers. In both cases, we demonstrate how one can restrict the storage requirements to order of $N$ (where $N$ is the grid size), while requiring only $O(N \log N)$ operations for every iteration of the iterative solver. As a proof of concept, we implement the method, and demonstrate its scalability, efficiency, and generality.

While the paper is structured around FDE optimization problems, we conjecture that the presented method has a significantly wider range of applicability. As a future research direction, we would like to extend the method, in order to solve a significantly larger family of optimization problems, that only enjoy a partial (multilevel–) Toeplitz (or circulant) structure in their coefficient matrices.

Acknowledgements

SP acknowledges financial support from a Principal’s Career Development PhD scholarship at the University of Edinburgh. JWP acknowledges support from a Fellowship of The Alan Turing Institute. SL acknowledges financial support from a School of Mathematics PhD studentship at the University of Edinburgh, and JG acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC) grant EP/N019652/1.
Appendices

We now discuss how one can compute the matrix–vector product of a level–$k$ circulant matrix, say $C$, by an arbitrary vector $x$, in $O(n_1 \cdots n_k \log(n_1 \cdots n_k))$ operations. Recall that we write $C = \tilde{F}^* \Lambda \tilde{F}$, for an appropriate matrix $\tilde{F}$, and we store only the eigenvalues of $C$. Hence, all we need to do is to perform efficiently the multiplication $\tilde{F}x$ (forward multiplication) and $\tilde{F}^* x$ (backward multiplication). For that, we provide Algorithms 3, 4. The final matrix–vector product is summarized in Algorithm 5. The pseudo–code is written using MATLAB indexing notation.

**Algorithm 3** FM (Forward Multiplication)

**Input:** $x \in \mathbb{R}^{(n_1 \cdots n_k)}$, $k$.

```plaintext
if ($k = 1$) then
  $x = F_{n_k}x$, where $F_{n_k}$ the discrete Fourier transform.
else
  for ($i = 1 : n_k$) do
    tmp = $x((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1}))$.
    $x((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1})) = \text{FM}(tmp, k - 1)$.
  end for
  $x = Px$, ($O(n_1 \cdots n_k)$ since $P$ is stored as a vector).
for ($i = 1 : (n_1 \cdots n_{k-1})$) do
  tmp = $x((i - 1) \cdot n_k + 1 : i \cdot n_k)$.
  $x((i - 1) \cdot n_k + 1 : i \cdot n_k) = F_{n_k}^*(tmp)$.
end for
end if
return $x$.
```

**Algorithm 4** BM (Backward Multiplication)

**Input:** $x \in \mathbb{R}^{(n_1 \cdots n_k)}$, $k$.

```plaintext
if ($k = 1$) then
  $x = F_{n_k}^*x$.
else
  for ($i = 1 : (n_1 \cdots n_{k-1})$) do
    tmp = $x((i - 1) \cdot n_k + 1 : i \cdot n_k)$.
    $x((i - 1) \cdot n_k + 1 : i \cdot n_k) = F_{n_k}^*(tmp)$.
  end for
  $x = P^T x$.
for ($i = 1 : n_k$) do
  tmp = $x((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1}))$.
  $x((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1})) = \text{BM}(tmp, k - 1)$.
end for
end if
return $x$.
```
Algorithm 5 Level–k Circulant Operator

Input: $\Lambda, x \in \mathbb{R}^{(n_1 \cdots n_k)}$, $k$.

1. $x = \text{FM}(x, k)$.
2. $x = \Lambda \cdot x$ (component–wise).
3. $x = \text{BM}(x, k)$.

return $x$.

Finally, given a level–$k$ Toeplitz matrix, say $T$, we discuss how to compute the matrix–vector product $Tx$, for an arbitrary vector $x$, in $O(n_1 \cdots n_k \log(n_1 \cdots n_k))$ operations. To do so, we firstly have to represent $T$ as a level–$k$ circulant matrix $C$, of size $(2^k \cdot n_1 \cdots n_k) \times (2^k \cdot n_1 \cdots n_k)$ (noting that we assume $k$ to be $O(1)$). This is achieved using Algorithm 6. We pass this level–$k$ circulant matrix to Algorithm 7, which returns its eigenvalues. The initial vector $x$, must also be appropriately extended as in Algorithm 8. Then, using the new matrix and vector, we perform the multiplication with the previously presented level–$k$ circulant operator. To retrieve the result in the initial dimension, we employ Algorithm 9. We should mention here that $T$ is stored as a vector, containing all the non–repeating elements of the matrix, that is, $T \in \mathbb{R}^{((2 \cdot n_1 - 1) \cdots (2 \cdot n_k - 1)) \times 1}$. The assumption regarding storage is that for a level–1 Toeplitz matrix $T$, we only need $2 \cdot n_1 - 1$ elements. The first $n_1$ are extracted from the first column of $T$, while the remaining $n_1 - 1$ from its first row. The matrix–vector multiplication is summarized in Algorithm 10.

Algorithm 6 CR (Circulant Representation)

Input: $T \in \mathbb{R}^{((2 \cdot n_1 - 1) \cdots (2 \cdot n_k - 1))}$, $k$.

1. $C = 0^{(2^k \cdot n_1 \cdots n_k)}$.
2. if $(k = 1)$ then
   3. $C = [t_1, \ldots, t_{n_k}, 0, t_{2 \cdot n_k - 1}, \ldots, t_{n_k + 1}]$.
   4. else
   5. $nr_C = 2^{k-1} \cdot n_1 \cdots n_{k-1}$.
   6. $nr_T = (2n_1 - 1) \cdots (2n_k - 1)$.
   7. for $(i = 1 : n_k)$ do
      8. $tmp = T((i - 1) \cdot nr_T + 1 : i \cdot nr_T)$.
      9. $C((i - 1) \cdot nr_C + 1 : i \cdot nr_C) = \text{CR}(tmp, k - 1)$.
   10. end for
   11. for $(i = (n_k + 2) : 2 \cdot n_k)$ do
      12. $tmp = T((3 \cdot n_k - i) \cdot nr_T + 1 : (3 \cdot n_k - i + 1) \cdot nr_T)$.
      13. $C((i - 1) \cdot nr_C + 1 : i \cdot nr_C) = \text{CR}(tmp, k - 1)$.
   14. end for
   15. end if

return $C$. 

29
Algorithm 7 CE (Circulant Eigenvalues)

Input: $C \in \mathbb{R}^{(n_1 \cdots n_k)}$, $k$.

if ($k = 1$) then
    $\Lambda = F_{n_k} C$.
else
    $\Lambda = 0_{(n_1 \cdots n_k) \times 1}$.
    for ($i = 1 : n_k$) do
        $tmp = C((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1}))$.
        $\Lambda((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1})) = CE(tmp, k - 1)$.
    end for
    $\Lambda = \Lambda(P)$ (Apply the permutation).
    for ($i = 1 : n_1 \cdots n_{k-1}$) do
        $tmp = C((i - 1) \cdot n_k + 1 : i \cdot n_k)$.
        $\Lambda((i - 1) \cdot n_k + 1 : i \cdot n_k) = F_k tmp$.
    end for
end if
return $\Lambda$.

Algorithm 8 extend (vector expansion)

Input: $x \in \mathbb{R}^{(n_1 \cdots n_k)}$, $k$.

if ($k = 1$) then
    $w = [x; 0_{(n_1 \cdots n_k) \times 1}]$.
else
    for ($i = 1 : n_k$) do
        $tmp = x((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1}))$.
        $w((i - 1) \cdot (2^{k-1} \cdot n_1 \cdots n_{k-1}) + 1 : (2^{k-1} \cdot n_1 \cdots n_{k-1})) = \text{extend}(tmp, k - 1)$.
    end for
end if
return $w$.

Algorithm 9 reduce (vector reduction)

Input: $x \in \mathbb{R}^{(2^k n_1 \cdots n_k)}$, $k$.

if ($k = 1$) then
    $w = x(1 : n_k)$.
else
    for ($i = 1 : n_k$) do
        $tmp = x((i - 1) \cdot (2^{k-1} \cdot n_1 \cdots n_{k-1}) + 1 : i \cdot (2^{k-1} \cdot n_1 \cdots n_{k-1}))$.
        $w((i - 1) \cdot (n_1 \cdots n_{k-1}) + 1 : i \cdot (n_1 \cdots n_{k-1})) = \text{reduce}(tmp, k - 1)$.
    end for
end if
return $w$. 
Algorithm 10 Level–k Toeplitz Operator

**Input:** $T \in \mathbb{R}^{((2^n-1)\cdots(2^n-k-1))}$, $x \in \mathbb{R}^{(n_1\cdots n_k)\times 1}$, $k$.

1. $w = 0_{(2^k\cdot n_1\cdots n_k)}$.
2. $w = \text{extend}(x)$.
3. $C = CR(T, k)$.
4. $\Lambda = \text{CE}(C)$.
5. $w = \text{Level–k Circulant Operator}(\Lambda, w, k)$.
6. return $x = \text{reduce}(w)$.

References

[1] A. Barker, T. Rees, and M. Stoll. A Fast Solver for an $H^1$ Regularized PDE-Constrained Optimization Problem. *Comm. Comp. Phys.*, Vol. 19 (No. 1): 143–167, 2016.

[2] H. Bart, I. Gohberg, and M. A. Kaashoek. *Toeplitz Matrices and Linear Systems*. I. Gohberg (ed.), Toeplitz Centennial. Operator Theory: Advances and Applications, vol. 4. Birkhäuser, Basel, 1982.

[3] M. Benzi, G. H. Golub, and J. Liesen. Numerical Solution of Saddle Point Problems. *Acta Numer.*, Vol. 14 : 1–137, 2005.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, Vol. 3 (No. 1): 1–122, 2010.

[5] J. Bunch. Stability of Methods for Solving Toeplitz Systems of Equations. *SIAM J. Sci. Stat. Comp.*, Vol. 6 (No. 2): 349–364, 1985.

[6] B. A. Carreras, V. E. Lynch, and G. M. Zaslavsky. Anomalous Diffusion and Exit Time Distribution of Particle Tracers in Plasma Turbulence Models. *Phys. Plasma*, Vol. 8 : 5096–5103, 2001.

[7] R. H. Chan. Circulant Preconditioners for Hermitian Toeplitz Systems. *SIAM J. Mat. Anal. Appl.*, Vol. 10 (No. 4): 542–550, 1989.

[8] R. H. Chan and X. Q. Jin. A Family of Block Preconditioners for Block Systems. *SIAM J. Sci. Stat. Comp.*, Vol. 13 (No. 5): 1218–1235, 1992.

[9] R. H. Chan, J. G. Nagy, and R. J. Plemmons. Circulant Preconditioned Toeplitz Least Squares Iterations. *SIAM J. Mat. Anal. Appl.*, Vol. 15 (No. 1): 80–97, 1994.

[10] R. H. Chan and G. Strang. Toeplitz Equations by Conjugate Gradient with Circulant Preconditioner. *SIAM J. Sci. Stat. Comp.*, Vol. 10 (No. 1): 104–119, 1989.
[11] R. H. Chan and C. K. Wong. Best-Conditioned Circulant Preconditioners. *Linear Alg. Appl.*, Vol. 218: 205–211, 1995.

[12] T. F. Chan. An Optimal Circulant Preconditioner for Toeplitz Systems. *SIAM J. Sci. Stat. Comp.*, Vol. 9 (No. 4), 1988.

[13] T. F. Chan and J. A. Olkin. Circulant Preconditioners for Toeplitz-Block Matrices. *Numerical Alg.*, Vol. 6: 89–101, 1994.

[14] S. Cipolla and F. Durastante. Fractional PDE Constrained Optimization: An Optimize-then-Discretize Approach with L-BFGS and Approximate Inverse Preconditioning. *Appl. Numer. Math.*, Vol. 123: 43–57, 2018.

[15] W. Deng. Finite Element Method for the Space and Time Fractional Fokker-Planck Equation. *SIAM J. Numer. Anal.*, Vol. 47: 204–226, 2008.

[16] K. Diethelm, N. J. Ford, A. D. Freed, and Y. Luchko. Algorithms for the Fractional Calculus: A Selection of Numerical Methods. *Comput. Method Appl. Math.*, Vol. 194 (No. 6): 743–773, 2005.

[17] S. Dolgov, J. W. Pearson, D. V. Savostyanov, and M. Stoll. Fast Tensor Product Solvers for Optimization Problems with Fractional Differential Equations as Constraints. *Appl. Math. Comp.*, Vol. 273: 604–623, 2016.

[18] M. Donatelli, C. Garoni, M. Mazza, S. Serra-Capizzano, and D. Sesana. Preconditioned HSS Method for Large Multilevel Block Toeplitz Linear Systems via the Notion of Matrix-Valued Symbol. *Numer. Linear Algebra Appl.*, Vol. 23: 83–119, 2016.

[19] F. Durastante and S. Cipolla. Fractional PDE Constrained Optimization: Box and Sparse Constrained Problems, in Numerical Methods for Optimal Control Problems (M. Falcone, R. Ferretti, L. Grune, W. M. McEneaney, eds.). Springer, 2018.

[20] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Numerical Mathematics and Scientific Computation, Oxford University Press, 2014.

[21] V. J. Ervin, N. Heuer, and J. P. Roop. Numerical Approximation of a Time Dependent, Nonlinear, Space-Fractional Diffusion Equation. *SIAM J. Numer. Anal.*, Vol. 45: 572–591, 2007.

[22] A. Forsgren, P. E. Gill, and E. Wong. Primal and Dual Active-Set Methods for Convex Quadratic Programming. *Math. Prog.*, Vol. 159 (No. 1-2): 469–508, 2016.

[23] C. Garoni, M. Mazza, and S. Serra-Capizzano. Block Generalized Locally Toeplitz Sequences: From the Theory to the Applications. *Axioms*, Vol. 7 (No. 3): 49, 2018.
[24] C. Garoni and S. Serra-Capizzano. The Theory of Generalized Locally Toeplitz Sequences: A Review, an Extension, and a Few Representative Applications. D. Bini, T. Ehrhardt, A. Karlovich, I. Spitkovsky (ed.), Large Truncated Toeplitz Matrices, Toeplitz Operators, and Related Topics. Operator Theory: Advances and Applications, Vol. 259. Birkhäuser, Cham, 2017.

[25] T. Goldstein, B. O’Donoghue, S. Setzer, and R. Baraniuk. Fast Alternating Direction Optimization Methods. SIAM J. Imaging Sci., Vol. 7 (No. 3): 1588–1623, 2014.

[26] J. Gondzio. Interior Point Methods 25 Years Later. European J. of Operational Research, Vol. 218 (No. 3): 587–601, 2013.

[27] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly Optimal Sparse Fourier Transform. STOC, 2012.

[28] M. Hazewinkel. Encyclopedia of Mathematics: A-Integral – Coordinates. Springer, 1995.

[29] G. Heidel, V. Khoromskaia, B. N. Khoromskij, and V. Schulz. Tensor Approach to Optimal Control Problems with Fractional $d$-dimensional Elliptic Operator in Constraints. arxiv:1809.01971, 2018.

[30] R. M. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. J. Res. Nat. Bur. of Stand., Vol. 49 (No. 6): 409–436, 1952.

[31] X. Q. Jin. A Preconditioner for Constrained and Weighted Least Squares Problems with Toeplitz Structure. BIT Num. Math., Vol. 36 (No. 1): 101–109, 1996.

[32] R. C. Koeller. Applications of Fractional Calculus to the Theory of Viscoelasticity. J. Appl. Mech., Vol. 51 (No. 2): 299–307, 1984.

[33] Y. A. Kuznetsov. Efficient Iterative Solvers for Elliptic Finite Element Problems on Nonmatching Grids. Russ. J. Numer. Anal. Math. Modelling, Vol. 10 (No. 3): 187–211, 1995.

[34] S. L. Lei, X. Chen, and X. Zhang. FFT-Based Preconditioners for Toeplitz-Block Least Squares Problems. SIAM J. Numer. Anal., Vol. 30 (No. 6): 1740–1768, 1992.

[35] S. L. Lei, X. Chen, and X. Zhang. Multilevel Circulant Preconditioners for High-Dimensional Fractional Diffusion Equations. East Asian J. Appl. Math., Vol. 6 (No. 2): 109–130, 2016.

[36] S. L. Lei and H. W. Sun. A Circulant Preconditioner for Fractional Diffusion Equations. J. Comp. Phys., Vol. 242 : 715–725, 2013.

[37] N. Levinson. The Wiener RMS Error Criterion in Filter Design and Prediction. J. Math. Phys., Vol. 25 : 261–278, 1947.
[38] A. Mang and G. Biros. Constrained $H^1$-Regularization Schemes for Diffeomorphic Image Registration. *SIAM J. Imaging Sci.*, Vol. 9 (No. 3): 1154–1194, 2016.

[39] M. M. Meerschaert, H.-P. Scheffler, and C. Tadjeran. Finite Difference Methods for Two-Dimensional Fractional Dispersion Equation. *J. Comput. Phys.*, Vol. 211 (No. 1): 249–261, 2006.

[40] M. M. Meerschaert and C. Tadjeran. Finite Difference Approximations for Fractional Advection–Dispersion Flow Equations. *J. Comput. Appl. Math.*, Vol. 172 (No. 1): 65–77, 2004.

[41] M. M. Meerschaert and C. Tadjeran. Finite Difference Approximations for Two-Sided Space-Fractional Partial Differential Equations. *Appl. Numer. Math.*, Vol. 56 (No. 1): 80–90, 2006.

[42] M. Murphy, G. Golub, and A. Wathen. A Note on Preconditioning for Indefinite Linear Systems. *SIAM J. Sci. Comp.*, Vol. 21 (No. 6): 1969–1972, 2000.

[43] C. C. Paige and M. A. Saunders. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM J. Numer. Anal.*, Vol. 12 (No. 4): 617–629, 1975.

[44] I. Podlubny. *Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of their Solution and Some of their Applications*. Mathematics in Science and Engineering, Volume 198 (1st Edition), Elsevier, 1999.

[45] S. S. Reddi. Eigenvector Properties of Toeplitz Matrices and their Application to Spectral Analysis of Time Series. *Signal Processing*, Vol. 7 : 45–56, 1984.

[46] R. T. Rockafellar. Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming. *Math. of Operations Research*, Vol. 1 (No. 2): 97–116, 1976.

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

[48] S. Serra-Capizzano and E. Tyrtyshnikov. Any Circulant-Like Preconditioner for Multilevel Matrices is Not Superlinear. *SIAM J. Matrix Anal. Appl.*, Vol. 21 (No. 2): 431–439, 2000.

[49] E. E. Tyrtyshnikov. Optimal and Superoptimal Circulant Preconditioners. *SIAM J. Mat. Anal. Appl.*, Vol. 13 (No. 2): 459–473, 1992.

[50] H. Wang, K. Wang, and T. Sircar. A Direct $O(N\log^2 N)$ Finite Difference Method for Fractional Diffusion Equations. *J. Comp. Phys.*, Vol. 229 : 8095–8104, 2010.

[51] H. Xiao and W. B. Wu. Covariance Matrix Estimation for Stationary Time Series. *Ann. Statist.*, Vol. 40 (No. 1): 466–493, 2012.
[52] K. Ye and L. H. Lim. Algorithms for Structured Matrix-Vector Product of Optimal Bilinear Complexity. *IEEE ITW*, (doi 10.1109/ITW.2016.7606846), 2016.

[53] G. M. Zaslavsky, D. Stevens, and H. Weitzner. Self-Similar Transport in Incomplete Chaos. *Phys. Rev. E*, Vol. 48 : 1683–1694, 1993.

[54] M.-L. Zeng and G.-F. Zhang. An MHSS–Like Iteration Method for Two–by–Two Linear Systems with Application to FDE Optimization Problems. *J. Comput. Appl. Math.*, Vol. 352 : 368–381, 2019.