Computers and Mathematics with Applications 75 (2018) 1929–1941

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A fast algorithm for solving the space–time fractional diffusion equation☆

Siwei Duo [a], Lili Ju [b], Yanzhi Zhang [a,*]

[a] Department of Mathematics and Statistics, Missouri University of Science and Technology, Rolla, MO 65409-0020, USA
[b] Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA

ARTICLE INFO

Article history:
Available online 17 May 2017

Keywords:
Fractional diffusion equation
Spectral fractional Laplacian
Caputo fractional derivative
Matrix transfer method
Discrete sine transform
Matrix–vector product

ABSTRACT

In this paper, we propose a fast algorithm for efficient and accurate solution of the space–time fractional diffusion equations defined in a rectangular domain. The spatial discretization is done by using the central finite difference scheme and matrix transfer technique. Due to its nonlocality, numerical discretization of the spectral fractional Laplacian $(-\Delta)_s^{\alpha/2}$ results in a large dense matrix. This causes considerable challenges not only for storing the matrix but also for computing matrix–vector products in practice. By utilizing the compact structure of the discrete system and the discrete sine transform, our algorithm avoids to store the large matrix from discretizing the nonlocal operator and also significantly reduces the computational costs. We then use the Laplace transform method for time integration of the semi-discretized system and a weighted trapezoidal method to numerically compute the convolutions needed in the resulting scheme. Various experiments are presented to demonstrate the efficiency and accuracy of our method.

1. Introduction

The space–time fractional diffusion equation, obtained from the standard diffusion equation by replacing the integer-order spatial and temporal derivatives with their fractional counterparts, has received great attention in the literature [1–9]. It has been widely applied to study the anomalous diffusive processes associated with sub-diffusion (fractional in time) and super-diffusion (fractional in space) in many fields [10–12]. However, the nonlocality of the fractional derivatives introduces considerable challenges in both analysis and simulations of the fractional diffusion equation. In this paper, we present an efficient and accurate numerical method to solve the space–time fractional diffusion equation, which has significantly less computational complexity and memory than the existing methods in the literature.

Let $\Omega \subset \mathbb{R}^d$ (for $d = 1, 2$ or 3) denote an open and bounded domain. We consider a space–time fractional diffusion equation of the following form [7,2,13,9]:

$$\partial_t^\gamma u(\mathbf{x}, t) = -\kappa_\alpha(-\Delta)_s^{\alpha/2} u(\mathbf{x}, t) + f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega,\ t > 0, \tag{1.1}$$

$$u(\mathbf{x}, t) = 0, \quad \mathbf{x} \in \partial\Omega,\ t \geq 0, \tag{1.2}$$

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \overline{\Omega}, \tag{1.3}$$

☆ This work was supported by the US National Science Foundation under Grant Numbers DMS-1217000, DMS-1521965 and DMS-1620465.
* Corresponding author.
E-mail addresses: sddy9@mst.edu (S. Duo), ju@math.sc.edu (L. Ju), zhangyanz@mst.edu (Y. Zhang).

http://dx.doi.org/10.1016/j.camwa.2017.04.008
0898-1221/

for $0 < \gamma < 1$ and $\alpha > 0$, where κ_α is the diffusion coefficient, and f denotes a source term. The operator ∂_t^γ is defined as the Caputo fractional derivative of order γ, i.e.,

$$\partial_t^\gamma u(\mathbf{x}, t) = \frac{1}{\Gamma(1-\gamma)} \int_0^t \frac{1}{(t-\tau)^\gamma} \partial_\tau u(\mathbf{x}, \tau)\, d\tau, \quad 0 < \gamma < 1, \tag{1.4}$$

where $\Gamma(\cdot)$ represents the Gamma function. As $\gamma \to 1$, the Caputo derivative in (1.4) gives the first-order temporal derivative ∂_t. The *spectral fractional Laplacian* (also known as the *fractional power of the Dirichlet Laplacian*) is defined via the spectral decomposition [14,13,9,15,7,16,17], i.e.,

$$(-\Delta)_s^{\alpha/2} u(\mathbf{x}) = \sum_{k \in \mathbb{N}^d} c_k\, \lambda_k^{\alpha/2} \varphi_k(\mathbf{x}), \quad \alpha > 0, \tag{1.5}$$

where λ_k and φ_k are the eigenvalues and eigenfunctions of the Laplace operator $-\Delta$ on the bounded domain $\Omega \subset \mathbb{R}^d$ with homogeneous Dirichlet boundary conditions. We would like to specially remark that the operator in (1.5) is different from the fractional Laplacian with extended homogeneous Dirichlet boundary condition (i.e., $u \equiv 0$, for $\mathbf{x} \in \mathbb{R}^d \setminus \Omega$), although these two operators are freely interchanged in some literature. The *fractional Laplacian* $(-\Delta)^{\alpha/2}$ is defined via a pseudo-differential operator with the symbol $-|\xi|^\alpha$ [18,19]:

$$(-\Delta)^{\alpha/2} u(\mathbf{x}) = \mathcal{F}^{-1}[|\xi|^\alpha \mathcal{F}[u](\xi)], \quad \alpha > 0, \tag{1.6}$$

where $\mathcal{F}$ represents the Fourier transform, and $\mathcal{F}^{-1}$ denotes its inverse transform. For $\alpha \in (0, 2)$, the fractional Laplacian can be also defined as the following integral form [19,20,18]:

$$(-\Delta)^{\alpha/2} u(\mathbf{x}) = c_{d,\alpha}\, \text{P.V.} \int_{\mathbb{R}^d} \frac{u(\mathbf{x}) - u(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|^{d+\alpha}}\, d\mathbf{y}, \quad \alpha \in (0, 2), \tag{1.7}$$

where P.V. stands for the principal value integral, and $c_{d,\alpha}$ denotes a normalization constant. From a probabilistic point of view, the fractional Laplacian $(-\Delta)^{\alpha/2}$ with extended homogeneous Dirichlet boundary condition represents an infinitesimal generator of a symmetric α-stable Lévy process that particles are killed upon leaving the domain Ω, while the spectral fractional Laplacian $(-\Delta)_s^{\alpha/2}$ represents an infinitesimal generator of the process that first kills Brownian motion in a bounded domain Ω and then subordinates the killed Brownian motion via a $\frac{\alpha}{2}$-stable subordinator; see [21] for more discussion and comparison of these two operators. To distinguish them, we will include the subscript 's' in the operator $(-\Delta)_s^{\alpha/2}$ to indicate that it is defined via the spectral decomposition.

The nonlocal nature of the spectral fractional Laplacian $(-\Delta)_s^{\alpha/2}$ introduces considerable challenges in solving the space–time fractional diffusion equations (1.1)–(1.3), especially in high spatial dimensions. To the best of our knowledge, numerical studies of (1.1)–(1.3) in the literature are still limited to one- and two-dimensional problems [13,1,7,9,4,5]. To localize the problem, a Dirichlet-to-Neumann mapping approach is proposed in [22,7], where $(-\Delta)_s^{\alpha/2}$ is realized by an operator mapping the Dirichlet boundary condition to a Neumann type condition, and different numerical methods are proposed to solve the extended problems [7,1]. In these methods, one solves a local problem instead of the nonlocal one, however, this comes at the cost of introducing one more spatial dimension to the problem and raises questions about computational efficiency [7,1]. In [4,9], another approach is proposed to first discretize $-\Delta$ by using the finite difference/element methods and obtain its matrix representation A, and then the matrix representation of the spectral fractional Laplacian $(-\Delta)_s^{\alpha/2}$ can be given by $\mathrm{A}^{\alpha/2}$. Usually, $\mathrm{A}^{\alpha/2}$ is a dense matrix, and it is costly to directly compute this exponent. In [9], to avoid directly evaluating $\mathrm{A}^{\alpha/2}$, they rewrite the scheme in terms of the matrix function vector product, $g(\mathrm{A})\mathbf{b}$, for a suitable vector $\mathbf{b}$, and then approximate this product by preconditioned Lanczos methods. Later, various methods are proposed in [23] to improve the efficiency in computing $g(\mathrm{A})\mathbf{b}$. In addition, quadrature approximation methods for discretizing Dunford Taylor integral representation of the inverse operator $(-\Delta)_s^{-\alpha/2}$ can be found in [14,24], which can be applied to solve the fractional diffusion problem.

In this paper, we propose an efficient and accurate numerical method to solve the space–time fractional diffusion equations (1.1)–(1.3) in a rectangular domain. It is based on the matrix transfer technique proposed in [4]. The main merits of our algorithm include that it requires less computational cost and memory, and thus it is more efficient in solving higher dimensional problems. Moreover, our method can be easily implemented in computer codes. The rest of the paper is organized as follows. In Section 2, we present the spatial discretization of (1.1) and propose a fast algorithm for its efficient computation. In Section 3, we introduce the Laplace transform for time integration and obtain the full discretization of the space–time fractional diffusion equation by using the weighted trapezoidal rule. In Section 4, we test the accuracy and efficiency of the proposed algorithm by studying both time-independent and time-dependent problems. Some conclusions and remarks are provided in Section 5.

2. Semi-discretization in space

In this section, we introduce a finite difference method for discretizing the spatial operator $(-\Delta)_s^{\alpha/2}$ and propose a fast algorithm based on the discrete sine transform for its efficient computation. We will start introducing our method for the one-dimensional ($d = 1$) case and then generalize it to the higher dimensions ($d > 1$).

2.1. One-dimensional case

For $d = 1$, we denote x_i (for $0 \le i \le N$) as the uniform grid points and h as the mesh size. Let $u_i = u_i(t)$ represent the numerical approximation of the solution $u(x_i, t)$, for $i \in \mathcal{T}_1$, where $\mathcal{T}_1 = \{i \,|\, 1 \le i \le N-1\}$. Denote the solution vector as $\mathbf{u} = (u_1, u_2, \ldots, u_{N-1})^T$. Then, the second order accurate central finite difference method for the Laplace operator $-\Delta = -d_x^2$ reads:

$$-\Delta_h u(x,t) = \mathrm{A}\mathbf{u}(t),$$

where the tridiagonal matrix $\mathrm{A} = \frac{1}{h^2} G_{(N-1)\times(N-1)}$ with

$$\mathrm{G}_{M\times M} = \begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 2 \end{pmatrix}_{M\times M}, \tag{2.1}$$

for any $M \in \mathbb{N}$. It shows in [4] that if A is a matrix representation of a differential operator T, then the matrix representation of the operator $T^{\alpha/2}$ can be given by $\mathrm{A}^{\alpha/2}$, for $\alpha \in \mathbb{R}$. Following it, we obtain the finite difference approximation of $(-\Delta)_s^{\alpha/2}$ as:

$$(-\Delta)_{s,h}^{\alpha/2} u(x,t) = \mathrm{A}^{\alpha/2}\mathbf{u}(t), \tag{2.2}$$

i.e., $\mathrm{A}^{\alpha/2}$ can be viewed as a matrix representation of the operator $(-\Delta)_s^{\alpha/2}$. Substituting (2.2) into (1.1) yields the semi-discretization in space of the one-dimensional fractional diffusion equation:

$$\partial_t^\gamma \mathbf{u}(t) = -\kappa_\alpha \mathrm{A}^{\alpha/2}\mathbf{u}(t) + \mathbf{f}(t), \tag{2.3}$$

where the vector $\mathbf{f}(t) = (f(x_1,t),\, f(x_2,t),\, \ldots,\, f(x_{N-1},t))^T$.

Since the matrix A is symmetric positive definite, it can be diagonalized as

$$\mathrm{A} = \mathrm{P}\,\Lambda\mathrm{P}^{-1}, \tag{2.4}$$

where Λ is a diagonal matrix composed of the eigenvalues of A, i.e.,

$$\Lambda = \mathrm{diag}[\lambda_1,\, \lambda_2,\, \ldots,\, \lambda_{N-1}], \qquad \lambda_i = \frac{4}{h^2}\sin^2\left(\frac{i\pi}{2N}\right), \quad 1 \le i \le N-1, \tag{2.5}$$

and the matrix $\mathrm{P} \in \mathbb{R}^{(N-1)\times(N-1)}$ is defined with its entry

$$\mathrm{P}_{i,j} = \sin\left(\frac{ij\pi}{N}\right), \quad i,j = 1,2,\ldots,N-1. \tag{2.6}$$

From (2.4), we obtain

$$\mathrm{A}^{\alpha/2} = \mathrm{P}\,\Lambda^{\alpha/2}\,\mathrm{P}^{-1} \tag{2.7}$$

with Λ and P as defined in (2.5) and (2.6), respectively. If N is small, the matrix–vector products in (2.3) can be directly computed via first obtaining $\mathrm{A}^{\alpha/2}$ from the matrix diagonalization in (2.4)–(2.7). However, as pointed out in [4,9,7,14], the computational cost of this approach is extremely high if N is large. Hence, various methods are proposed in [9,23] to improve the efficiency in evaluating the matrix vector product $\mathrm{A}^{\alpha/2}\mathbf{u}$.

Here, we will propose a fast algorithm to exactly compute the matrix vector product in (2.3). From the definition of P in (2.6), we find that the product P$\mathbf{u}$ is equivalent to the discrete sine transform (DST) of the vector $\mathbf{u}$, while $\mathrm{P}^{-1}\mathbf{u}$ can be realized by the inverse DST of $\mathbf{u}$. Therefore, we can rewrite the semi-discretization scheme in (2.3) as

$$\partial_t^\gamma \mathbf{u}(t) = -\kappa_\alpha\, \mathcal{S}_1[\Lambda^{\alpha/2}\mathcal{S}_1^{-1}[\mathbf{u}(t)]] + \mathbf{f}(t), \tag{2.8}$$

where $\mathcal{S}_d$ ($d = 1$, 2, or 3) denotes the d-dimensional DST, and $\mathcal{S}_d^{-1}$ represents its inverse transform. We note that the main idea here is very similar to that used in the popular fast Poisson solver. Thus, the computational cost of evaluating the matrix vector product in (2.3) can be reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log N)$. Moreover, in contrast to the schemes in [9,23], our algorithm exactly computes the matrix–vector product, and thus no additional computational errors are introduced.

2.2. Higher dimensional case

In the following, we will generalize the idea of our fast algorithm from the one dimension to higher dimensions ($d > 1$). Let $N_k \in \mathbb{N}$ represent the number of intervals in the $x^{(k)}$-direction, for $k = 1, 2, \ldots, d$. Denote $x_i^{(k)}$ (for $0 \le i \le N_k$) as the uniform grid points in the $x^{(k)}$-direction with mesh size $h_k > 0$. Denote the index set $\mathcal{T}_d = \{(i_1, i_2, \ldots, i_d) \mid 1 \le i_1 \le N_1 - 1, \ \ldots, \ 1 \le i_d \le N_d - 1\}$. Let $u_{i_1,\ldots,i_d}(t)$ represent the numerical approximation of the solution $u(x_{i_1}^{(1)}, \ \ldots, \ x_{i_d}^{(d)}, \ t)$, for $(i_1, \ \ldots, \ i_d) \in \mathcal{T}_d$. Denote the d-dimensional solution array as $\mathrm{U}(t) = \{u_{i_1,\ldots,i_d}\}_{(N_1-1)\times\cdots\times(N_d-1)}$, and let $\mathbf{u}(t)$ be the vector form of $\mathrm{U}(t)$ by the natural dimension-wise ordering. Then, the central finite difference discretization of the Laplace operator $-\Delta = -\sum_{k=1}^{d} \partial^2_{x^{(k)}}$ can be written as:

$$-\Delta_h u(\mathbf{x}, t) = \left(\sum_{k=1}^{d} \mathrm{I}_1 \otimes \cdots \otimes \mathrm{I}_{k-1} \otimes \mathrm{A}_k \otimes \mathrm{I}_{k+1} \otimes \cdots \otimes \mathrm{I}_d\right)\mathbf{u}(t),$$

where the matrix

$$\mathrm{A}_k = \frac{1}{h_k^2}\mathrm{G}_{(N_k-1)\times(N_k-1)}, \quad k = 1, 2, \ldots, d$$

with G defined in (2.1), $\mathrm{I}_k \in \mathbb{R}^{(N_k-1)\times(N_k-1)}$ represents the identity matrix, and $\otimes$ represents the Kronecker product (or tensor product) of two matrices. Following the same arguments of the one-dimensional case, we can obtain the finite difference discretization of the operator $(-\Delta)_s^{\alpha/2}$ in d-dimension as

$$(-\Delta)_{s,h}^{\alpha/2} u(\mathbf{x}, t) = \left(\sum_{k=1}^{d} \mathrm{I}_1 \otimes \cdots \otimes \mathrm{I}_{k-1} \otimes \mathrm{A}_k \otimes \mathrm{I}_{k+1} \otimes \cdots \otimes \mathrm{I}_d\right)^{\alpha/2} \mathbf{u}(t). \tag{2.9}$$

In order to introduce a fast algorithm for efficiently computing the right hand side of (2.9), we first rewrite it as

$$\begin{aligned}
&\left(\sum_{k=1}^{d} \mathrm{I}_1 \otimes \cdots \otimes \mathrm{I}_{k-1} \otimes \mathrm{A}_k \otimes \mathrm{I}_{k+1} \otimes \cdots \otimes \mathrm{I}_d\right)^{\alpha/2} \\
&= \sum_{m_1,\ldots,m_{d-1}=0}^{\infty} \binom{\alpha/2}{m_1, \ldots, m_{d-1}} \left(\mathrm{A}_1 \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)^{m_1} \cdots \\
&\quad \left(\mathrm{I}_1 \otimes \cdots \otimes \mathrm{A}_{d-1} \otimes \mathrm{I}_d\right)^{m_{d-1}} \left(\mathrm{I}_1 \otimes \cdots \otimes \mathrm{I}_{d-1} \otimes \mathrm{A}_d\right)^{\frac{\alpha}{2} - \sum_{i=1}^{d-1} m_i},
\end{aligned} \tag{2.10}$$

where $\binom{\alpha/2}{m_1,\ldots,m_{d-1}}$ represents the generalized multinomial coefficient for $\alpha > 0$. For $k = 1, 2, \ldots, d$, the matrix A_k can be decomposed as:

$$\mathrm{A}_k = \mathrm{P}_k \, \Lambda_k \, \mathrm{P}_k^{-1}, \tag{2.11}$$

where Λ_k is the diagonal matrix composed of the eigenvalues of A_k, respectively, and P_k is defined in the same manner as in (2.6), i.e.,

$$\Lambda_k = \mathrm{diag}\left[\lambda_1^{(k)}, \ \lambda_2^{(k)}, \ \ldots, \ \lambda_{N_k-1}^{(k)}\right], \qquad \lambda_i^{(k)} = \frac{4}{h_k^2} \sin^2\left(\frac{i\pi}{2N_k}\right), \quad 1 \le i \le N_k - 1,$$

and

$$(\mathrm{P}_k)_{i,j} = \sin\left(\frac{ij\pi}{N_k}\right), \quad i, j = 1, 2, \ldots, N_k - 1.$$

For $m_1 \in \mathbb{R}$, using the decomposition in (2.11), the mixed-product and inverse properties of Kronecker products, we obtain

$$\begin{aligned}
\left(\mathrm{A}_1 \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)^{m_1} &= \left[\left(\mathrm{P}_1 \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)\left(\Lambda_1 \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)\left(\mathrm{P}_1^{-1} \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)\right]^{m_1} \\
&= \left(\mathrm{P}_1 \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)\left(\Lambda_1 \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)^{m_1}\left(\mathrm{P}_1 \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d\right)^{-1} \\
&= \left(\mathrm{P}_1 \, \Lambda_1^{m_1} \, \mathrm{P}_1^{-1}\right) \otimes \mathrm{I}_2 \otimes \cdots \otimes \mathrm{I}_d.
\end{aligned}$$

Following the similar lines, we get for any $m_i \in \mathbb{R}$,

$$\left(\mathrm{I}_1 \otimes \cdots \otimes \mathrm{I}_{i-1} \otimes \mathrm{A}_i \otimes \mathrm{I}_{i+1} \cdots \otimes \mathrm{I}_d\right)^{m_i} = \mathrm{I}_1 \otimes \cdots \otimes \mathrm{I}_{i-1} \otimes \left(\mathrm{P}_i \, \Lambda_i^{m_i} \, \mathrm{P}_i^{-1}\right) \otimes \mathrm{I}_{i+1} \cdots \otimes \mathrm{I}_d. \tag{2.12}$$

Substituting (2.10) with (2.12) into (2.9), we obtain

$$(-\Delta)_{s,h}^{\alpha/2}u(\mathbf{x},t) = \Bigg[\sum_{m_1,\ldots,m_{d-1}=0}^{\infty}\binom{\alpha/2}{m_1,\ldots,m_{d-1}}\Big(\Big(\mathrm{P}_1\,\Lambda_1^{m_1}\,\mathrm{P}_1^{-1}\Big)\otimes \mathrm{I}_2\otimes\cdots\otimes \mathrm{I}_d\Big)\cdots$$
$$\Big(\mathrm{I}_1\otimes\cdots\otimes\Big(\mathrm{P}_{d-1}\Lambda_{d-1}^{m_{d-1}}\mathrm{P}_{d-1}^{-1}\Big)\otimes \mathrm{I}_d\Big)\Big(\mathrm{I}_1\otimes\cdots\otimes \mathrm{I}_{d-1}\otimes\Big(\mathrm{P}_d\Lambda_d^{\frac{\alpha}{2}-\sum_{i=1}^{d-1}m_i}\mathrm{P}_d^{-1}\Big)\Big)\Bigg]\mathbf{u}(t) \tag{2.13}$$

in terms of the solution vector $\mathbf{u}$.

We now focus on rewriting the scheme (2.13) in terms of the solution array $\mathrm{U}(t)$, such that the matrix–vector products in it can be efficiently computed by our fast algorithm. To this end, we first introduce two operators. For a matrix $\mathrm{A}\in\mathbb{R}^{(N_l-1)\times(N_l-1)}$ and an array $\mathrm{U}\in\mathbb{R}^{(N_1-1)\times\cdots\times(N_d-1)}$, we define the operator $\circledast_l$ as:

$$\big(\mathrm{A}\circledast_l\mathrm{U}\big)_{i_1,\ldots,i_d} = \sum_{j=1}^{N_l-1}\mathrm{A}_{i_l,j}\mathrm{U}_{i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_d}\in\mathbb{R}^{(N_1-1)\times\cdots\times(N_d-1)}.$$

It is easy to verify that for any $1\le j,l\le d$, the two operators $\circledast_j$ and $\circledast_l$ are commutative, i.e.,

$$\mathrm{A}\circledast_j\mathrm{B}\circledast_l\mathrm{U} = \mathrm{B}\circledast_l\mathrm{A}\circledast_j\mathrm{U}.$$

We also define the operator $\odot$ of two arrays $\mathrm{U},\mathrm{V}\in\mathbb{R}^{(N_1-1)\times\cdots\times(N_d-1)}$ as

$$\big(\mathrm{U}\odot\mathrm{V}\big)_{i_1,\ldots,i_d} = \mathrm{U}_{i_1,\ldots,i_d}\mathrm{V}_{i_1,\ldots,i_d},$$

i.e., element by element multiplication of arrays. By the definition of the Kronecker product and our new operators, we can rewrite (2.13) from the vector form of $\mathbf{u}(t)$ into the form of the d-dimensional array $\mathrm{U}(t)$ as:

$$\begin{aligned}(-\Delta)_{s,h}^{\alpha/2}u(\mathbf{x},t) &= \sum_{m_1,\ldots,m_{d-1}=0}^{\infty}\binom{\alpha/2}{m_1,\ldots,m_{d-1}}\Big(\mathrm{P}_1\,\Lambda_1^{m_1}\,\mathrm{P}_1^{-1}\Big)\circledast_1\cdots\\ &\quad\circledast_{d-2}\Big(\mathrm{P}_{d-1}\,\Lambda_{d-1}^{m_{d-1}}\mathrm{P}_{d-1}^{-1}\Big)\circledast_{d-1}\Big(\mathrm{P}_d\,\Lambda_d^{\frac{\alpha}{2}-\sum_{i=1}^{d-1}m_i}\mathrm{P}_d^{-1}\Big)\circledast_d\mathrm{U}(t)\\ &= \sum_{m_1,\ldots,m_{d-1}=0}^{\infty}\binom{\alpha/2}{m_1,\ldots,m_{d-1}}\mathrm{P}_1\circledast_1\mathrm{P}_2\circledast_2\cdots\circledast_{d-1}\mathrm{P}_d\\ &\quad\circledast_d\Big[\mathrm{Q}_{m_1,\ldots,m_d}\odot\Big(\mathrm{P}_1^{-1}\circledast_1\mathrm{P}_2^{-1}\circledast_2\cdots\circledast_{d-1}\mathrm{P}_d^{-1}\circledast_d\mathrm{U}(t)\Big)\Big],\end{aligned} \tag{2.14}$$

where the d-dimensional array $\mathrm{Q}_{m_1,\ldots,m_d}$ is composed of

$$\big(\mathrm{Q}_{m_1,\ldots,m_d}\big)_{i_1,\ldots,i_d} = \Big(\lambda_{i_1}^{(1)}\Big)^{m_1}\cdots\Big(\lambda_{i_{d-1}}^{(d-1)}\Big)^{m_{d-1}}\Big(\lambda_{i_d}^{(d)}\Big)^{\frac{\alpha}{2}-\sum_{i=1}^{d-1}m_i},$$

for $(i_1,i_2,\ldots,i_d)\in\mathcal{T}_d$. The scheme in (2.14) can be further written as:

$$\begin{aligned}(-\Delta)_{s,h}^{\alpha/2}u(\mathbf{x},t) &= \mathrm{P}_1\circledast_1\mathrm{P}_2\circledast_2\cdots\circledast_{d-1}\mathrm{P}_d\circledast_d\Bigg[\Bigg(\sum_{m_1,\ldots,m_{d-1}=0}^{\infty}\binom{\alpha/2}{m_1,\ldots,m_{d-1}}\mathrm{Q}_{m_1,\ldots,m_d}\Bigg)\\ &\quad\odot\Big(\mathrm{P}_1^{-1}\circledast_1\mathrm{P}_2^{-1}\circledast_2\cdots\circledast_{d-1}\mathrm{P}_d^{-1}\circledast_d\mathrm{U}(t)\Big)\Bigg]\\ &= \mathrm{P}_1\circledast_1\cdots\circledast_{d-1}\mathrm{P}_d\circledast_d\Big[\mathrm{H}_d\odot\Big(\mathrm{P}_1^{-1}\circledast_1\cdots\circledast_{d-1}\mathrm{P}_d^{-1}\circledast_d\mathrm{U}(t)\Big)\Big],\end{aligned} \tag{2.15}$$

where the d-dimensional array H_d is defined as

$$(\mathrm{H}_d)_{i_1,\ldots,i_d} = \sum_{m_1,\ldots,m_{d-1}=0}^{\infty}\binom{\alpha/2}{m_1,\ldots,m_{d-1}}\big(\mathrm{Q}_{m_1,\ldots,m_d}\big)_{i_1,\ldots,i_d} = \Bigg(\sum_{k=1}^{d}\lambda_{i_k}^{(k)}\Bigg)^{\alpha/2}, \tag{2.16}$$

for $(i_1,i_2,\ldots,i_d)\in\mathcal{T}_d$.

Note that the scheme in (2.15) is the d-dimensional array form of (2.13). Noticing the definition of the matrix P_k, we can apply the d-dimensional DST or its inverse transform to compute the special products in (2.15) [25,26], i.e.,

$$\mathrm{P}_1\circledast_1\cdots\circledast_{d-1}\mathrm{P}_d\circledast_d\mathrm{U}(t) = \mathcal{S}_d[\mathrm{U}(t)],$$
$$\mathrm{P}_1^{-1}\circledast_1\cdots\circledast_{d-1}\mathrm{P}_d^{-1}\circledast_d\mathrm{U}(t) = \mathcal{S}_d^{-1}[\mathrm{U}(t)].$$

Thus, the spatial semi-discretization scheme of the d-dimensional fractional diffusion equations (1.1) can be written as

$$\partial_t^\gamma \mathrm{U}(t) = -\kappa_\alpha \, \mathcal{S}_d \left[\mathrm{H}_d \odot \mathcal{S}_d^{-1}[\mathrm{U}(t)]\right] + \mathrm{F}(t), \tag{2.17}$$

where the d-dimensional array $\mathrm{F}(t) = \left\{f(x_{i_1}^{(1)}, x_{i_2}^{(2)}, \ldots, x_{i_d}^{(d)}, t)\right\}_{(N_1-1)\times\cdots\times(N_d-1)}$. As a result, the computational cost for computing the spatial approximations is only $\mathcal{O}(N^d \log N)$ where $N = \max_{1\le k\le d} N_k$, instead of $\mathcal{O}(N^{d+1})$ by directly using the matrix–vector multiplications.

Remark 2.1. For the time-independent fractional diffusion equation

$$\kappa_\alpha(-\Delta)_s^{\alpha/2} u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2.18}$$

$$u(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega, \tag{2.19}$$

its solution (steady state) can be computed by

$$\mathrm{U} = \frac{1}{\kappa_\alpha} \mathcal{S}_d\Big[\mathrm{L}_d \odot \mathcal{S}_d^{-1}[\mathrm{F}]\Big] \tag{2.20}$$

with the d-dimensional array L_d defined as

$$\big(\mathrm{L}_d\big)_{i_1,\ldots,i_d} = 1/\big(\mathrm{H}_d\big)_{i_1,\ldots,i_d}, \quad (i_1, i_2, \ldots, i_d) \in \mathcal{T}_d.$$

3. Fully discrete scheme

In this section, we present time integration of the semi-discretized scheme in (2.17) to obtain a fully discrete system for numerical solution of the fractional diffusion equations (1.1)–(1.3). Since the system (2.17) is linear in time, we use the Laplace transform for time integration [9]. For other numerical methods of the Caputo fractional derivative, we refer readers to [27] and references therein.

Let $\widetilde{\mathrm{U}}(s) = \mathcal{L}\{\mathrm{U}\}(s)$ denote the Laplace transform of $\mathrm{U}(t)$. Taking the Laplace transform and then the inverse DST at both sides of (2.17) yields

$$s^\gamma \mathcal{S}_d^{-1}\big[\widetilde{\mathrm{U}}(s)\big] - s^{\gamma-1}\mathcal{S}_d^{-1}\big[\mathrm{U}(0)\big] = -\kappa_\alpha \left(\mathrm{H}_d \odot \mathcal{S}_d^{-1}\big[\widetilde{\mathrm{U}}(s)\big]\right) + \mathcal{S}_d^{-1}\big[\widetilde{\mathrm{F}}(s)\big].$$

Solving the above equation, we obtain

$$\widetilde{\mathrm{U}}(s) = \mathcal{S}_d\left[\left\{\frac{1}{s + \kappa_\alpha\, s^{1-\gamma}\,(\mathrm{H}_d)_{\mathbf{k}}}\right\}_{\mathbf{k}\in\mathcal{T}_d} \odot \left(\mathcal{S}_d^{-1}\big[\mathrm{U}(0)\big] + s^{1-\gamma}\mathcal{S}_d^{-1}\big[\widetilde{\mathrm{F}}(s)\big]\right)\right], \tag{3.1}$$

where $\mathcal{T}_d$ is the d-dimensional index set. We then take the inverse Laplace transform of (3.1) and obtain the numerical solution of (1.1)–(1.3) at time t as

$$\mathrm{U}(t) = \mathcal{S}_d\Big[\mathrm{M}_d^{\gamma,1}(t) \odot \mathcal{S}_d^{-1}\big[\mathrm{U}(0)\big] + \big(t^{\gamma-1}\mathrm{M}_d^{\gamma,\gamma}(t)\big) \circledast \mathcal{S}_d^{-1}\big[\mathrm{F}(t)\big]\Big], \quad t \ge 0. \tag{3.2}$$

For any $a, b > 0$, $\mathrm{M}_d^{a,b}(t)$ defines a d-dimensional array function with

$$\big(\mathrm{M}_d^{a,b}(t)\big)_{i_1,\ldots,i_d} = E_{a,b}\big(-\kappa_\alpha(\mathrm{H}_d)_{i_1,\ldots,i_d} t^\gamma\big), \quad \text{for } (i_1,\ldots,i_d) \in \mathcal{T}_d, \tag{3.3}$$

and $E_{a,b}(z)$ represents the Mittag-Leffler function defined as

$$E_{a,b}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(ak+b)}.$$

In practice, the Mittag-Leffler function $E_{a,b}(z)$ can be efficiently computed by using the algorithm proposed in [28]. The operator $\circledast$ is defined as

$$(\mathrm{U} \circledast \mathrm{V})_{i_1,\ldots,i_d} = \int_0^t \mathrm{U}_{i_1,\ldots,i_d}(t-\tau)\mathrm{V}_{i_1,\ldots,i_d}(\tau)\, d\tau,$$

i.e., element by element convolution of two arrays $\mathrm{U}(t)$ and $\mathrm{V}(t)$.

If the function f is time-dependent, one usually has to evaluate the convolutions in (3.2) numerically. Here, we introduce a weighted trapezoidal method for its computation. To simplify notations, let us denote

$$\xi_{\mathbf{k}} = -\kappa_\alpha\big(\mathrm{H}_d\big)_{\mathbf{k}}, \qquad \mathrm{G}_{\mathbf{k}}(t) = \big(\mathcal{S}_d^{-1}[\mathrm{F}(t)]\big)_{\mathbf{k}}, \quad \text{for } \mathbf{k} \in \mathcal{T}_d.$$

Then, each convolution in (3.2) can be written as

$$\int_0^t \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}} \mathrm{G}_{\mathbf{k}}(t-s)ds = \int_0^{\varepsilon} \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}} \mathrm{G}_{\mathbf{k}}(t-s)ds + \int_{\varepsilon}^t \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}} \mathrm{G}_{\mathbf{k}}(t-s)ds, \tag{3.4}$$

for $\mathbf{k} \in \mathcal{T}_d$. Here, $0 < \varepsilon \ll 1$ is a very small number.

Assuming that $\mathrm{G}_{\mathbf{k}}$ is uniformly bounded on $(0, t)$, then it holds

$$\lim_{\varepsilon \to 0} \int_0^{\varepsilon} \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}} \mathrm{G}_{\mathbf{k}}(t-s)ds = 0.$$

Define time sequence $t_n = \varepsilon + n\tau$ (for $n = 0, 1, \ldots$) with τ a uniform time step. Then at time $t = t_n$, we introduce a weighted trapezoidal method to approximate the second integral in (3.4), i.e.,

$$\begin{aligned}\int_{\varepsilon}^{t_n} \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}} \mathrm{G}_{\mathbf{k}}(t_n-s)ds &= \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}} \mathrm{G}_{\mathbf{k}}(t_n-s)\, ds \\ &\approx \sum_{i=0}^{n-1} \frac{\mathrm{G}_{\mathbf{k}}(t_n-t_i) + \mathrm{G}_{\mathbf{k}}(t_n-t_{i+1})}{2} \int_{t_i}^{t_{i+1}} \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}}\, ds.\end{aligned}$$

Simplifying it, we obtain the approximation

$$\int_{\varepsilon}^{t_n} \frac{E_{\gamma,\gamma}\left(\xi_{\mathbf{k}} s^{\gamma}\right)}{s^{1-\gamma}} \mathrm{G}_{\mathbf{k}}(t_n-s)ds \approx \sum_{i=0}^{n-1} \frac{\mathrm{G}_{\mathbf{k}}(t_{n-i}) + \mathrm{G}_{\mathbf{k}}(t_{n-i-1})}{2} \left[t_{i+1}^{\gamma} E_{\gamma,\gamma+1}\left(\xi_{\mathbf{k}} t_{i+1}^{\gamma}\right) - t_i^{\gamma} E_{\gamma,\gamma+1}\left(\xi_{\mathbf{k}} t_i^{\gamma}\right)\right].$$

Combining the above approximation with (3.2), we obtain the numerical solution at $t = t_n$ as:

$$\begin{aligned}\mathrm{U}^n = \mathcal{S}_d\Big[&\mathrm{M}_d^{\gamma,1}(t_n) \odot \mathcal{S}_d^{-1}\left[\mathrm{U}^0\right] + \frac{1}{2}\Big(\left(t_1^{\gamma}\mathrm{M}_d^{\gamma,\gamma+1}(t_1) - t_0^{\gamma}\mathrm{M}_d^{\gamma,\gamma+1}(t_0)\right) \odot \mathcal{S}_d^{-1}\left[\mathrm{F}(t_n)\right] \\ &+ \sum_{i=1}^{n-1}\left(t_{i+1}^{\gamma}\mathrm{M}_d^{\gamma,\gamma+1}(t_{i+1}) - t_{i-1}^{\gamma}\mathrm{M}_d^{\gamma,\gamma+1}(t_{i-1})\right) \odot \mathcal{S}_d^{-1}\left[\mathrm{F}(t_{n-i})\right] \\ &+ \left(t_n^{\gamma}\mathrm{M}_d^{\gamma,\gamma+1}(t_n) - t_{n-1}^{\gamma}\mathrm{M}_d^{\gamma,\gamma+1}(t_{n-1})\right) \odot \mathcal{S}_d^{-1}\left[\mathrm{F}(t_0)\right]\Big)\Big], \quad n = 0, 1, \ldots,\end{aligned} \tag{3.5}$$

where U^n denotes the numerical approximation of $\mathrm{U}(t_n)$. It shows that the solution at time $t = t_n$ depends on all of those at previous steps, since the Caputo fractional derivative is a nonlocal operator.

Remark 3.1. As $\gamma \to 1$, there is

$$\lim_{\gamma \to 1} E_{\gamma,\gamma}(z) = E_{1,1}(z) = e^z, \quad \text{for } z \in \mathbb{R}.$$

Thus, the solution in (3.2) converges to

$$\mathrm{U}(t) = \mathcal{S}_d\Big[\mathrm{M}_d(t) \odot \mathcal{S}_d^{-1}\left[\mathrm{U}(0)\right] + \mathrm{M}_d(t) \circledast \mathcal{S}_d^{-1}\left[\mathrm{F}(t)\right]\Big], \quad t \geq 0,$$

with the array $\mathrm{M}_d(t)$ defined as

$$\left(\mathrm{M}_d(t)\right)_{i_1,\ldots,i_d} = \exp\left(-\kappa_{\alpha}(\mathrm{H}_d)_{i_1,\ldots,i_d} t\right), \quad \text{for } (i_1, \ldots, i_d) \in \mathcal{T}_d$$

i.e., the solution of the standard space-fractional diffusion equation.

Remark 3.2. If f is time-independent, i.e., $f(\mathbf{x}, t) \equiv f(\mathbf{x})$ for any $t > 0$, the convolutions in (3.2) can be computed explicitly. Hence, the solution in (3.2) reduces to

$$\mathrm{U}(t) = \mathcal{S}_d\Big[\mathrm{M}_d^{\gamma,1}(t) \odot \mathcal{S}_d^{-1}\left[\mathrm{U}(0)\right] + \left(t^{\gamma}\mathrm{M}_d^{\gamma,\gamma+1}(t)\right) \odot \mathcal{S}_d^{-1}\left[\mathrm{F}\right]\Big], \quad t \geq 0.$$

Remark 3.2 implies that if the function f is time-independent, our scheme is exact in time, and thus the only numerical errors are from the spatial discretization.

4. Numerical experiments

In this section, we will investigate the performance of the proposed fast algorithm for solving the diffusion problem (1.1)–(1.3) and its steady state through various examples.

Table 1
Spatial discretization errors $\|u_h - u_{\text{exact}}\|_2$ and convergence rates in solving (4.1)–(4.2) with the source term (4.5) in Example 1.

α	h						
	1/16	1/32	1/64	1/128	1/256	1/512	1/1024
0.4	5.378E−4	1.342E−4	3.353E−5	8.381E−6	2.095E−6	5.238E−7	1.310E−7
	C.R.	2.0028	2.0007	2.0002	2.0000	2.0000	2.0000
1.0	3.632E−4	9.049E−5	2.260E−5	5.650E−6	1.412E−6	3.531E−7	8.827E−8
	C.R.	2.0049	2.0012	2.0003	2.0001	2.0000	2.0000
1.4	2.125E−4	5.289E−5	1.321E−5	3.301E−6	8.253E−7	2.063E−7	5.158E−8
	C.R.	2.0063	2.0016	2.0004	2.0001	2.0000	2.0000
2.0	8.201E−5	2.038E−5	5.089E−6	1.272E−6	3.179E−7	7.947E−8	1.987E−8
	C.R.	2.0084	2.0021	2.0005	2.0001	2.0000	2.0000
2.9	1.670E−5	4.141E−6	1.033E−6	2.582E−7	6.454E−8	1.614E−8	4.034E−9
	C.R.	2.0115	2.0029	2.0007	2.0002	2.0000	2.0000
4.0	2.091E−6	5.172E−7	1.290E−7	3.222-8	8.053E−9	2.013E−9	5.03E−10
	C.R.	2.0154	2.0038	2.0010	2.0002	2.0002	2.0007

4.1. Steady states

We first study steady states of the two-dimensional diffusion problem and test the spatial accuracy of our method. Let the domain $\Omega = (0, 1)^2$. We solve the problem of the following form:

$$(-\Delta)_s^{\alpha/2} u(x, y) = f(x, y), \quad (x, y) \in \Omega, \tag{4.1}$$

$$u(x, y) = 0, \quad (x, y) \in \partial\Omega. \tag{4.2}$$

Its exact solution can be easily found from the definition of $(-\Delta)_s^{\alpha/2}$ as:

$$u_{\text{exact}}(x, y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \lambda_{mn}^{-\alpha/2} \widehat{f}_{mn}\, \varphi_{mn}(x, y), \quad (x, y) \in \overline{\Omega}, \tag{4.3}$$

where λ_{mn} and φ_{mn} are eigenvalues and eigenfunctions of the two-dimensional Dirichlet Laplace operator $-\Delta$, i.e.,

$$\lambda_{mn} = (m^2 + n^2)\pi^2, \qquad \varphi_{mn}(x, y) = 2\sin(m\pi x)\sin(n\pi y), \quad m, n \in \mathbb{N},$$

and the coefficients

$$\widehat{f}_{mn} = \int_0^1 \int_0^1 f(x, y)\varphi_{mn}(x, y)\, dx\, dy, \quad m, n \in \mathbb{N}. \tag{4.4}$$

In our simulations, we choose the mesh size as $h := h_x = h_y$ and denote u_h as the numerical solution. The spatial accuracy of our method will be studied for various functions f.

Example 1. We choose the function

$$f(x, y) = \sin(2\pi x)\sin(2\pi y). \tag{4.5}$$

Then, the exact solution in (4.3) reduces to

$$u_{\text{exact}}(x, y) = \frac{1}{8^{\alpha/2}\,\pi^{\alpha}} \sin(2\pi x)\sin(2\pi y).$$

This example has been used as a benchmark to test the accuracy of the numerical methods in [7,1] for solving (4.1)–(4.2), where a PDE approach based on the Dirichlet-to-Neumann mapping is used.

Table 1 presents the spatial discretization errors $\|u_h - u_{\text{exact}}\|_2$ and the convergence rate ("C.R.") of our method, for various $\alpha > 0$. It shows that for fixed mesh size h, numerical errors decrease when the power α increases. Moreover, our method has the second order rate of convergence, independent of the power α.

Example 2. We consider a constant function

$$f(x, y) \equiv 1. \tag{4.6}$$

In contrast to (4.5), it is smooth but incompatible as it does not vanish at the boundary. Then the exact solution $u_{\text{exact}}(x, y)$ is given by (4.3) with

$$\widehat{f}_{mn} = \frac{2(1 - (-1)^m)(1 - (-1)^n)}{mn\pi^2}, \quad \text{for } m, n \in \mathbb{N}.$$

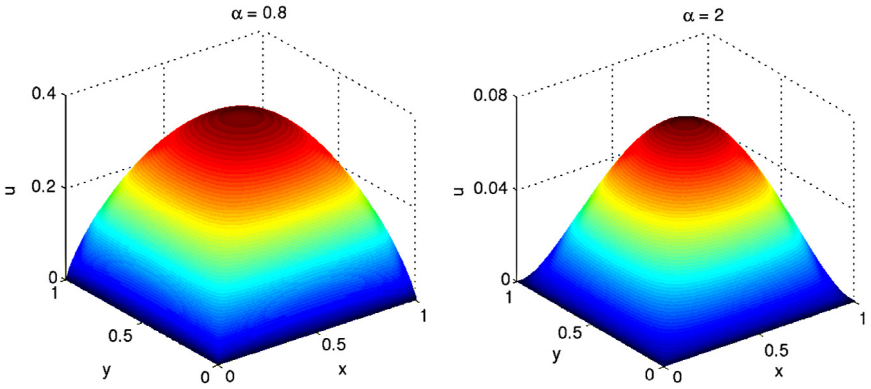


Fig. 1. Plots of the solution $u(x, y)$ of (4.1)–(4.2) with the source term (4.6) for different α in Example 2.

Table 2
Spatial discretization errors $\|u_h - u_{\text{exact}}\|_2$ and convergence rates in solving (4.1)–(4.2) with the source term (4.6) in Example 2.

α	h						
	1/16	1/32	1/64	1/128	1/256	1/512	1/1024
0.4	3.950E−3	2.023E−3	1.055E−3	5.563E−4	2.944E−4	1.550E−4	7.993E−5
	C.R.	0.9651	0.9392	0.9235	0.9182	0.9252	0.9556
1.0	1.210E−3	4.046E−4	1.359E−4	4.621E−5	1.590E−5	5.523E−6	1.928E−6
	C.R.	1.5807	1.5737	1.5565	1.5391	1.5257	1.5181
1.4	4.844E−4	1.328E−4	3.579E−5	9.567E−6	2.548E−6	6.776E−7	1.801E−7
	C.R.	1.8667	1.8918	1.9036	1.9088	1.9108	1.9113
2.0	1.345E−4	3.384E−5	8.473E−6	2.119E−6	5.299E−7	1.325E−7	3.312E−8
	C.R.	1.9913	1.9975	1.9993	1.9998	1.9999	2.0000
2.9	1.942E−5	4.853E−6	1.213E−6	3.033E−7	7.583E−8	1.896E−8	4.739E−9
	C.R.	2.0008	2.0000	2.0000	2.0000	2.0000	2.0001
4.0	8.389E−7	2.123E−7	5.321E−8	1.331E−8	3.329E−9	8.32E−10	2.08E−10
	C.R.	1.9826	1.9960	1.9991	1.9996	2.0004	2.0000

Fig. 1 illustrates the solution $u(x, y)$ for $\alpha = 0.8$ and 2. It shows that the steady state is symmetric with respect to the center of the domain, and its magnitude reduces when α increases. Furthermore, boundary layers may appear in the solution u if α is small; the smaller the power α, the sharper the boundary layers.

Table 2 presents the numerical errors $\|u_h - u_{\text{exact}}\|_2$ and convergence rates for various α. In practice, the exact solution in (4.3) is approximated by truncating it into finite terms, i.e., $1 \le m, n \le 2^{14}$. Table 2 shows that for a fixed mesh size h, the discretization errors reduce when α increases. We find that the convergence rate of our method in this case depends on the power α. From Table 2 and our extensive simulations, we conjecture that in this case our method has a convergence rate of $\mathcal{O}(h^p)$, where

$$p = \begin{cases} \alpha + \dfrac{1}{2}, & \text{if } \alpha < \dfrac{3}{2}, \\ 2, & \text{otherwise}, \end{cases}$$

that is, the power $\alpha = 3/2$ is a critical point of the convergence rate.

Example 3. We choose f as a step function, i.e.,

$$f(x, y) = \begin{cases} 1, & \text{if } (x - 1/2)(y - 1/2) > 0, \\ -1, & \text{otherwise} \end{cases} \tag{4.7}$$

which is incompatible and discontinuous. In this case, the exact solution of (4.1)–(4.2) is given by (4.3) with

$$\widehat{f}_{mn} = \frac{2}{mn\pi^2}\Big[\cos(n\pi) - 2\cos\Big(\frac{n\pi}{2}\Big) + 1\Big]\Big[\cos(m\pi) - 2\cos\Big(\frac{m\pi}{2}\Big) + 1\Big], \quad \text{for } m, n \in \mathbb{N}.$$

Fig. 2 displays the solution $u(x, y)$ for $\alpha = 0.8$ and 2. It shows that although the function f is discontinuous along the lines of $x = 1/2$ and $y = 1/2$, the solution u is continuous everywhere. Moreover, it is antisymmetric with respect to these two lines. The smaller the power α, the larger the magnitude of the solution u.

Table 3 illustrates the numerical errors $\|u_h - u_{\text{exact}}\|_2$ and convergence rates of our methods for various α, where again the exact solution u_{exact} is obtained by truncating (4.3) into terms for $1 \le m, n \le 2^{14}$. Similar to previous examples, for a

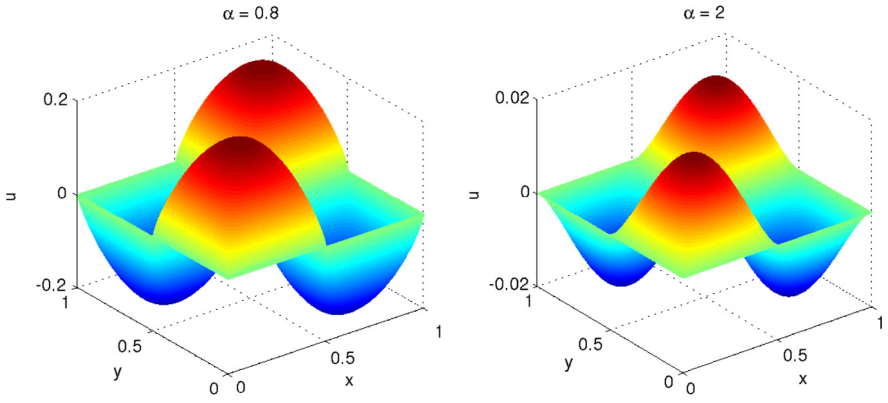


Fig. 2. Plots of the solution $u(x, y)$ of (4.1)–(4.2) with the source term (4.7) for different α in Example 3.

Table 3
Spatial discretization errors $\|u_h - u_{\text{exact}}\|_2$ and convergence rates in solving (4.1)–(4.2) with the source term (4.7) in Example 3.

α	h						
	1/16	1/32	1/64	1/128	1/256	1/512	1/1024
0.1	2.555E−1	1.736E−1	1.162E−1	7.718E−2	5.109E−2	3.377E−2	2.229E−2
	C.R.	0.5573	0.5795	0.5900	0.5951	0.5976	0.5989
0.4	1.206E−1	6.974E−2	3.932E−2	2.185E−2	1.204E−2	6.593E−3	3.596E−3
	C.R.	0.7904	0.8267	0.8474	0.8602	0.8687	0.8747
1.0	3.715E−2	1.916E−2	9.717E−3	4.892E−3	2.454E−3	1.229E−3	6.151E−4
	C.R.	0.9555	0.9794	0.9901	0.9952	0.9976	0.9988
1.4	1.937E−2	9.931E−3	5.026E−3	2.529E−3	1.268E−3	6.351E−4	3.178E−4
	C.R.	0.9638	0.9824	0.9912	0.9955	0.9977	0.9989
2.0	7.731E−3	3.964E−3	2.007E−3	1.010E−3	5.064E−4	2.536E−4	1.269E−4
	C.R.	0.9637	0.9820	0.9910	0.9955	0.9978	0.9989
4.0	3.897E−4	1.994E−4	1.009E−4	5.076E−5	2.546E−5	1.275E−5	6.379E−6
	C.R.	0.9665	0.9828	0.9913	0.9956	0.9978	0.9989

fixed mesh size h, the larger the power α, the smaller the numerical errors. Moreover, the convergence rate depends on the power α. Our extensive simulations show that the convergence rate of our method is $\mathcal{O}(h^p)$ with

$$p = \begin{cases} \alpha + \dfrac{1}{2}, & \text{if } \alpha < \dfrac{1}{2}, \\ 1, & \text{otherwise.} \end{cases}$$

Compared to that in Example 2, the convergence rate in this case is lower due to the discontinuity of the function f.

4.2. Evolution dynamics

Due to the nonlocality of the operator $(-\Delta)_s^{\alpha/2}$, the computational costs and memory of high dimensional problems are considerably large. So far, no report on three-dimensional results can be found in the literature yet. In the following, we will apply our method to study the dynamics of the fractional diffusion equations (1.1)–(1.3) in both two-dimensional and three-dimensional cases.

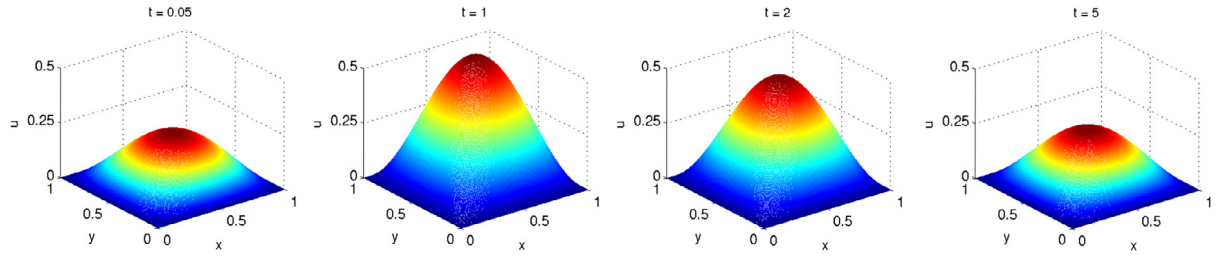
Example 4. We solve the two-dimensional fractional diffusion equations (1.1) on the domain $\Omega = (0, 1)^2$, where the diffusion coefficient $\kappa_\alpha = 0.1$, and the source function

$$f(x, y) = \sqrt{2}e^{-t} \sin(\pi x) \sin(\pi y). \tag{4.8}$$

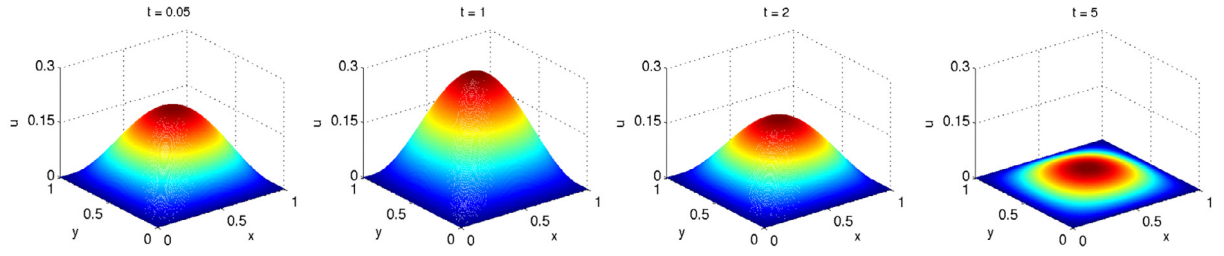
The initial condition in (1.3) is taken as

$$u_0(x, y) = xy(1 - x)(1 - y), \quad (x, y) \in [0, 1]^2. \tag{4.9}$$

Fig. 3 presents the simulated time evolution of $u(x, y, t)$ for $\alpha = 0.4$ or 1.6, where $\gamma = 0.6$ is fixed. When t is small, the dynamics are dominant by the source term f, which makes the solution u increase over time. But, the diffusion becomes significant after certain time, and thus the solution quickly decays. We find that the decay rate depends on α—the larger the power α is, the faster the solution diffuses.



(a) $\alpha = 0.4$ and $\gamma = 0.6$.



(b) $\alpha = 1.6$ and $\gamma = 0.6$.

Fig. 3. Time evolution of the solution of (1.1)–(1.3) with $\kappa_\alpha = 0.1$, the source term (4.8), and the initial condition (4.9) in Example 4.

Table 4
Temporal discretization errors $\|u_{h,\tau}(t) - u_{\text{exact}}(t)\|_2$ and convergence rates in solving (1.1)–(1.3) for time $t = 0.5$ with $\kappa_\alpha = 0.1$, the source term (4.8), and the initial condition (4.9) in Example 4.

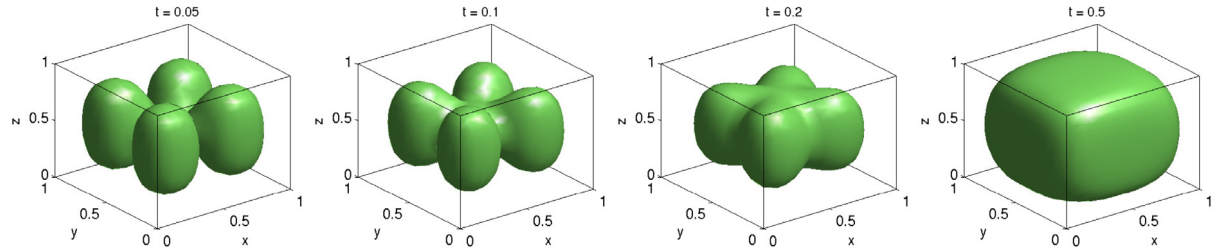
(α, γ)	τ				
	1/32	1/64	1/128	1/256	1/512
(0.4, 0.6)	1.814E−4	5.257E−5	1.466E−5	3.900E−6	9.642E−7
	C.R.	1.7868	1.8425	1.9118	2.0145
(1.0, 0.6)	1.810E−4	5.250E−5	1.465E−5	3.894E−6	9.586E−7
	C.R.	1.7853	1.8417	1.9114	2.0143
(1.4, 0.6)	1.791E−4	5.208E−5	1.455E−5	3.871E−6	9.586E−7
	C.R.	1.7819	1.8396	1.9103	2.0138
(2.0, 0.6)	1.694E−4	4.980E−5	1.401E−5	3.741E−6	9.279E−7
	C.R.	1.7659	1.8299	1.9049	2.0112
(1.4, 0.2)	3.818E−4	1.301E−4	4.142E−5	1.211E−5	3.174E−6
	C.R.	1.5528	1.6518	1.7741	1.9319
(1.4, 0.5)	2.569E−4	7.791E−5	2.249E−5	6.120E−6	1.535E−6
	C.R.	1.7216	1.7924	1.8777	1.9953
(1.4, 0.8)	7.389E−5	1.988E−5	5.231E−6	1.338E−6	3.241E−7
	C.R.	1.8944	1.9258	1.9677	2.0451
(1.4, 1.0)	2.327E−5	5.812E−6	1.449E−6	3.579E−7	8.522E−8
	C.R.	2.0011	2.0042	2.0171	2.0704

Table 4 lists the temporal discretization errors $\|u_{h,\tau}(t) - u_{\text{exact}}(t)\|_2$ and convergence rates at time $t = 0.5$, where the "exact" solution $u_{\text{exact}}(t)$ is numerically obtained by solving the diffusion problem (1.1)–(1.3) with fine mesh size and time step, i.e., $h = \tau = 1/2048$. In all simulations, we fix the spatial mesh size $h = 1/2048$ and choose the small parameter $\varepsilon = 0.001$. We find that the temporal errors reduce as the power α and/or γ increases, and our method has a convergence rate of $\mathcal{O}(\tau^2)$ in time. As discussed previously, the only temporal errors in our proposed method come from numerically evaluating the convolutions, which can be improved if a higher order quadrature method is used.

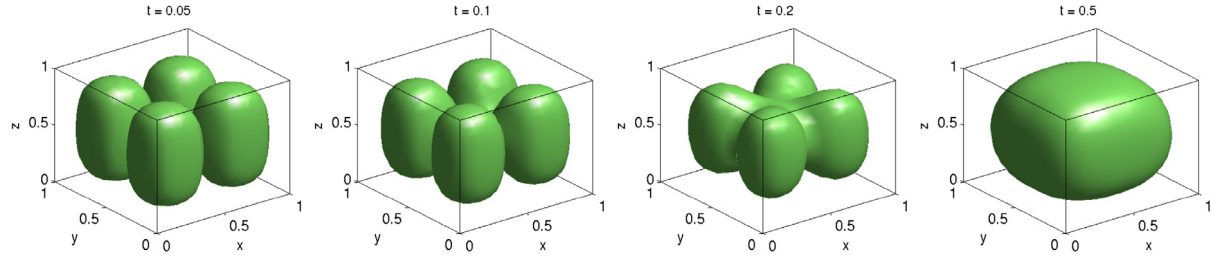
To further demonstrate its effectiveness, in the following example we will apply our method to study the dynamics of the three-dimensional fractional diffusion equations (1.1)–(1.3).

Example 5. We study the diffusion problem (1.1)–(1.3) on the domain $\Omega = (0, 1)^3$, where the diffusion coefficient $\kappa_\alpha = 0.1$ and a time-periodic source function is applied:

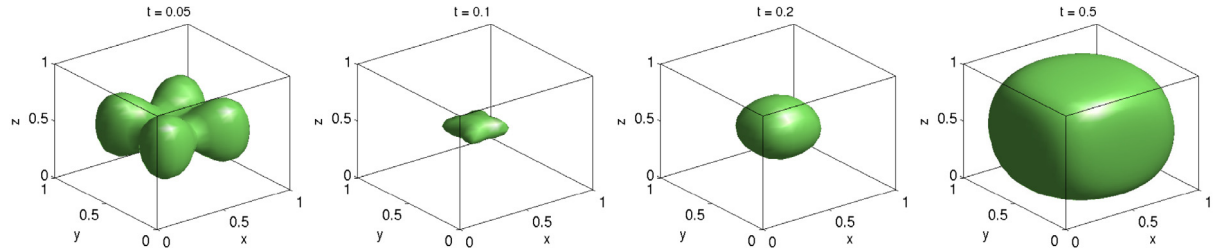
$$f(x, y, z, t) = 0.1 \exp\left(-20\left(\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2 + \left(z - \frac{1}{2}\right)^2\right)\right) \sin(4\pi t).$$



(a) $\alpha = 0.5$ and $\gamma = 0.4$.



(b) $\alpha = 0.5$ and $\gamma = 0.8$.



(c) $\alpha = 1.3$ and $\gamma = 0.8$.

Fig. 4. Time evolution of the diffusion equations (1.1)–(1.3) with different parameter sets in Example 5, where the isosurface plots of $u(x, y, z, t) = 0.05$ are presented.

The initial condition in (1.3) is chosen as

$$u_0(x, y, z) = |\sin(2\pi x)\sin(2\pi y)|(z - z^2), \quad (x, y, z) \in [0, 1]^3. \tag{4.10}$$

Fig. 4 presents the simulated isosurface plots of $u(x, y, z, t) = 0.05$ for various powers α and γ. Initially the solution is not smooth, but it is quickly smoothed along the dynamics. This phenomenon is similar to that observed in the classical diffusion equation (i.e., $\alpha = 2$ and $\gamma \to 1$). The nonlocality of $(-\Delta)_s^{\alpha/2}$ usually demands extremely large computational costs and memory, which hampers the existing numerical methods from solving (1.1)–(1.3) in three dimensions. However, our method with fast algorithm significantly reduces the computational costs, and thus it is effective in simulating high dimensional problems.

5. Conclusions

We proposed a fast algorithm for efficient and accurate solutions of the fractional diffusional equation with the spectral fractional Laplacian in space and the Caputo fractional derivative in time. Usually, numerical discretization of the spectral fractional Laplacian results in a large dense matrix. In practice, it is challenging not only to store the matrix but also to compute the matrix–vector products, especially in high dimensional cases. In our method, the central finite difference scheme and matrix transfer technique are used for spatial discretization. We utilized the compact structure of the discrete system and introduced a fast algorithm based on the discrete sine transform. Our method avoids to store the large coefficient matrix and significantly reduces the computational costs in the study of fractional diffusion problems. For time integration, we proposed the Laplace transform method and introduced a weighted trapezoidal method to approximate the convolutions in the resulting scheme. Our method is exact in time if the source term is time-independent; otherwise, it has the second order of accuracy in time. Our fast algorithm is easy to implement and effective for simulating the fractional diffusion equation. We conducted systematic numerical experiments to test the accuracy of the method. Rigorous numerical analysis on the accuracy will be carried out in our future work.

References

[1] L. Chen, R.H. Nochetto, E. Otárola, A.J. Salgado, A PDE approach to fractional diffusion: a posteriori error analysis, J. Comput. Phys. 293 (2015) 339–358.
[2] Z.-Q. Chen, M.M. Meerschaert, E. Nane, Space–time fractional diffusion on bounded domains, J. Math. Anal. Appl. 393 (2012) 479–488.
[3] F. Huang, F. Liu, The space–time fractional diffusion equation with Caputo derivatives, J. Appl. Math. Comput. 19 (2005) 179–190.
[4] M. Ilic, F. Liu, I. Turner, V. Anh, Numerical approximation of a fractional-in-space diffusion equation. I, Fract. Calc. Appl. Anal. 8 (2005) 323–341.
[5] M.M. Meerschaert, D.A. Benson, H.-P. Scheffler, B. Baeumer, Stochastic solution of space–time fractional diffusion equations, Phys. Rev. E 65 (2002) 041103. 4.
[6] F. Mainardi, Y. Luchko, G. Pagnini, The fundamental solution of the space–time fractional diffusion equation, Fract. Calc. Appl. Anal. 4 (2001) 153–192.
[7] R.H. Nochetto, E. Otárola, A.J. Salgado, A PDE approach to fractional diffusion in general domains: a priori error analysis, Found. Comput. Math. 15 (2015) 733–791.
[8] S. Vong, P. Lyu, X. Chen, S.-L. Lei, High order finite difference method for time-space fractional differential equations with Caputo and Riemann–Liouville derivatives, Numer. Algorithms 72 (2016) 195–210.
[9] Q. Yang, I. Turner, F. Liu, M. Ilić, Novel numerical methods for solving the time-space fractional diffusion equation in two dimensions, SIAM J. Sci. Comput. 33 (2011) 1159–1180.
[10] J.W. Hanneken, B.N. Narahari Achar, D.M. Vaught, K.L. Harrington, A random walk simulation of fractional diffusion, J. Mol. Liq. 114 (2004) 153–157.
[11] J. Klafter, M.F. Shlesinger, G. Zumofen, Beyond Brownian motion, Phys. Today 49 (1996) 33–39.
[12] F.J. Molz III, G.J. Fix III, S. Lu, A physical interpretation for the fractional derivative in Lévy diffusion, Appl. Math. Lett. 15 (2002) 907–911.
[13] A. Bueno-Orovio, D. Kay, K. Burrage, Fourier spectral methods for fractional-in-space reaction–diffusion equations, BIT 54 (2014) 937–954.
[14] A. Bonito, J.E. Pasciak, Numerical approximation of fractional powers of elliptic operators, Math. Comp. 84 (2015) 2083–2110.
[15] R. Servadei, E. Valdinoci, On the spectrum of two different fractional operators, Proc. Roy. Soc. Edinburgh Sect. A 144 (2014) 831–855.
[16] X. Cabré, J. Tan, Positive solutions of nonlinear problems involving the square root of the Laplacian, Adv. Math. 224 (2010) 2052–2093.
[17] A. Capella, J. Dávila, L. Dupaigne, Y. Sire, Regularity of radial extremal solutions for some non-local semilinear equations, Comm. Partial Differential Equations 36 (2011) 1353–1384.
[18] N.S. Landkof, Foundations of Modern Potential Theory, Springer-Verlag, New York-Heidelberg, 1972.
[19] S.G. Samko, A.A. Kilbas, O.I. Marichev, Fractional Integrals and Derivatives, Gordon and Breach Science Publishers, Yverdon, 1993.
[20] E.M. Stein, Singular Integrals and Differentiability Properties of Functions, in: Princeton Mathematical Series, No. 30, Princeton University Press, Princeton, N.J, 1970.
[21] S. Duo, H. Wang, Y. Zhang, A review and comparison of nonlocal diffusion operators related to the fractional Laplacian, preprint.
[22] L. Caffarelli, L. Silvestre, An extension problem related to the fractional Laplacian, Comm. Partial Differential Equations 32 (2007) 1245–1260.
[23] K. Burrage, N. Hale, D. Kay, An efficient implicit FEM scheme for fractional-in-space reaction–diffusion equations, SIAM J. Sci. Comput. 34 (2012) A2145–A2172.
[24] I.P. Gavrilyuk, W. Hackbusch, B.N. Khoromskij, Data-sparse approximation to the operator-valued functions of elliptic operator, Math. Comp. 73 (2004) 1297–1324.
[25] L. Ju, J. Zhang, L. Zhu, Q. Du, Fast explicit integration factor methods for semilinear parabolic equations, J. Sci. Comput. 62 (2015) 431–455.
[26] L. Zhu, L. Ju, W.-D. Zhao, Fast high-order compact exponential time differencing Runge–Kutta methods for second-order semilinear parabolic equations, J. Sci. Comput. 67 (2016) 1043–1065.
[27] Y. Lin, C. Xu, Finite difference/spectral approximations for the time-fractional diffusion equation, J. Comput. Phys. 225 (2007) 1533–1552.
[28] R. Garrappa, Numerical evaluation of two and three parameter Mittag-Leffler functions, SIAM J. Numer. Anal. 53 (2015) 1350–1369.