A fast method for a generalized nonlocal elastic model

Ning Du ^a   Hong Wang ^b   Che Wang ^b

^a*School of Mathematics, Shandong University, Jinan, Shandong 250100, China*

^b*Department of Mathematics, University of South Carolina, Columbia, South Carolina 29208, USA (E-mail address: hwang@math.sc.edu)*

Abstract

We develop a numerical method for a generalized nonlocal elastic model, which is expressed as a composition of a Riesz potential operator with a fractional differential operator, by composing a collocation method with a finite difference discretization.

By carefully exploring the structure of the coefficient matrix of the numerical method, we develop a preconditioned fast Krylov subspace method, which reduces the computations to $(N \log N)$ per iteration and the memory to $O(N)$. The use of the preconditioner significantly reduces the number of iterations, and the preconditioner can be inverted in $O(N \log N)$ computations. Numerical results show the utility of the method.

Key words: fractional differential equation, circulant preconditioners, fast Fourier transform, Riesz potential operator, Toeplitz systems

---

1. Introduction

In recent years many nonlocal models are emerging as a powerful tool for modeling challenging phenomena in a variety of disciplines, including anomalous transport, and long range time memory or spatial interactions. In the context of diffusion processes, traditional second-order diffusion equations model Fickian diffusion processes, in which a particle's movement obeys a Brownian motion. However, in the past few decades, many diffusion processes were found to exhibit anomalous diffusion behavior, in which the probability density functions of the underlying particle motions are characterized by an algebraically decaying tail and so cannot be modeled properly by second-order diffusion equations. Fractional diffusion equations provide an adequate

and accurate description of transport processes that exhibit anomalous diffusion, as the probability density functions of anomalous diffusion processes satisfy fractional diffusion equations [2, 21].

In the context of solid mechanics, classical theory assumes that all internal forces in a body act through a zero distance, which leads to mathematical models described by partial differential equations. These models do not provide a proper description of problems with spontaneous formation of discontinuities. In the last decade or so, the peridynamic theory was proposed as a reformulation of solid mechanics [8, 25], and provides an appropriate description of the deformation of a continuous body involving discontinuities or other singularities, which cannot be described properly by the classical partial differential equation models. Mathematically, the peridynamic models are expressed as an integral equation in terms of a Riesz potential operator, which does not explicitly involve the notion of deformation gradients.

Continuum elastic models have been widely used to study the dynamics of real physical systems in such as applications as flexible polymers, growing interfaces, and membranes [7, 18, 37]. Recently, a generalized elastic model was developed in [27] to study a generalization and relations of these elastic models. This model is expressed as a composition of a Riesz-like potential operator with a fractional differential operator [1, 24]. In a sense, the generalized elastic model can be viewed as a composition of a peridynamic model with a fractional differential equation model.

Due to the nonlocal nature and complexity of these nonlocal models, the corresponding numerical methods typically generate dense or full coefficient matrices. Traditionally, direct solvers were used to solve the numerical schemes, which have $O(N^3)$ computational complexity and require $O(N^2)$ storage for a numerical discretization of N unknowns [8, 17, 20]. Extensive research has been conducted on the development of fast and accurate numerical methods for nonlocal models. In the context of fractional differential equations, fast operator-splitting, or alternating-direction, or Krylov subspace-based numerical methods were developed with significantly reduced computational complexity and storage requirement [5, 15, 19, 20, 28, 30, 33, 34, 36]. In the context of peridynamic or nonlocal diffusion models, because the corresponding integral operator is fully coupled in all the directions, alternating-direction approach does not seem to be very efficient. A fast Krylov subspace-based method was developed that has an optimal order storage and an almost linear computational complexity [31, 32]. Numerical experiments show the utility of these fast methods.

However, due to the complexity of the generalized elastic model, its numerical discretization requires extra care. The finite volume methods [16, 29, 35] and aforementioned finite difference methods are well suited for the internal fractional differential operators in the model. However, they do not conveniently discretize the external Riesz potential operator. On the other hand, a collocation method is well fit for the external Riesz potential operator [11], but is inconvenient to discretize the internal fractional differential operator. A Galerkin method can be applied to discretize both the external Riesz potential operator and the internal fractional differential operator [9, 12, 13], but the computational cost is increased significantly as a double integral has to be used to discretize the external Riesz potential operator.

The objective of this paper is two folds: (i) we design a feasible numerical method for the model by composing a collocation method, which is well fit for the external Riesz potential operator, with a finite difference method, which is well suited for the discretization of the internal fractional differential operator. (ii) By carefully exploring the structure of the coefficient matrix of the numerical method, we develop a preconditioned fast Krylov subspace method, which reduces the computations to $(N \log N)$ per iteration and the storage to $O(N)$. The use of the preconditioner significantly reduces the number of iterations, and the preconditioner can be inverted in $O(N \log N)$ computations.

2. Model problem

We consider the following nonconventional Dirichlet boundary-value problem of the generalized fractional elastic model

$$\int_0^1 \Lambda(x-y) \left(d^+(y) \frac{\partial^\beta u(y)}{\partial_+ y^\beta} + d^-(y) \frac{\partial^\beta u(y)}{\partial_- y^\beta} \right) dy = f(x), \quad x \in (0, 1), \quad (1)$$

$$u(x) = 0, \quad x \notin (0, 1).$$

Here $d^+(x)$ and $d^-(x)$ are the left-sided and right-sided diffusivity coefficients, $\frac{\partial^\beta u}{\partial_+ x^\beta}$ and $\frac{\partial^\beta u}{\partial_- x^\beta}$ are the left-sided and right-sided (Grünwald-Letnikov) frac-

tional derivatives of order β ($1 < \beta < 2$) [22, 24]

$$\begin{aligned}\frac{\partial^\beta u}{\partial_+ x^\beta} &= \lim_{h \rightarrow 0} \frac{1}{h^\beta} \sum_{l=0}^{\lfloor x/h \rfloor} g_l^{(\beta)} u(x - lh), \\ \frac{\partial^\beta u}{\partial_- x^\beta} &= \lim_{h \rightarrow 0} \frac{1}{h^\beta} \sum_{l=0}^{\lfloor (1-x)/h \rfloor} g_l^{(\beta)} u(x + lh)\end{aligned}\tag{2}$$

where $\lfloor x \rfloor$ represents the floor of x and $g_k^{(\beta)} = (-1)^k \binom{\beta}{k}$ with $\binom{\beta}{k}$ being the fractional binomial coefficients. It is known that the Grunwald-Letnikov fractional derivatives coincide with the Riemann–Liouville fractional derivatives defined by [22, 24]

$$\begin{aligned}{}_0^R D_x^\beta u &:= \frac{1}{\Gamma(2-\beta)} \frac{d^2}{dx^2} \int_0^x \frac{u(s)}{(x-s)^{\beta-1}} ds, \\ {}_x^R D_1^\beta u &:= \frac{1}{\Gamma(2-\beta)} \frac{d^2}{dx^2} \int_x^1 \frac{u(s)}{(s-x)^{\beta-1}} ds.\end{aligned}\tag{3}$$

An arithmetic average of the left and right Riemann-Liouville fractional derivatives with a proper normalization constant gives a Riesz fractional derivative [24].

In addition, $\Lambda(x)$ represents the long-range hydrodynamic interactions, which is of the form

$$\Lambda(x) = \frac{1}{|x|^\alpha}, \quad 0 < \alpha < 1\tag{4}$$

or

$$\Lambda(x) = \frac{1}{|x| + a}, \quad 0 < a < 1.\tag{5}$$

To develop an efficient and accurate numerical approximation to the model problem, we decompose the problem as the following system

$$\begin{aligned}d^+(x) \frac{\partial^\beta u(x)}{\partial_+ x^\beta} + d^-(x) \frac{\partial^\beta u(x)}{\partial_- x^\beta} &= v(x), \quad x \in (0, 1), \\ u &= 0, \quad x \notin (0, 1),\end{aligned}\tag{6}$$

and

$$\int_0^1 \Lambda(x-y) v(y) dy = f(x), \quad x \in (0, 1),\tag{7}$$

Below we will apply the Meerschaert-Tadjeran finite difference method for the discretization of the fractional part (6) and the collocation method for the integral part (7). The two schemes are then combined to give a numerical scheme for the elastic model.

3. A numerical scheme for (1)

Let $h := 1/(N+1)$ be the mesh size for some positive integer N . We define an equidistant spatial partition $x_i := ih$ for $i = 0, 1, \dots, N+1$. Let u_i and v_i be the approximations to $u(x_i)$ and $v(x_i)$, respectively, and $d_i^+ = d^+(x_i)$, $d_i^- = d^-(x_i)$, $f_i = f(x_i)$ for $i = 0, 1, \dots, N+1$. The system (6)–(7), at least in principle, allows the use of numerical methods already developed for fractional diffusion equations and related nonlocal problems. As we shall see, the nonlocal nature of the problem makes a direct composition impossible, and so we still have to develop a numerical method for the problem.

3.1. A finite difference approximation to (6)

Suppose that v is known in (6), then problem (6) reduces to the homogeneous Dirichlet boundary-value problem of a space-fractional diffusion equation. Meerschaert and Tadjeran [20] showed that a fully implicit finite difference scheme with a direct truncation of the series in (2) turns out to be unconditionally unstable. Instead, they modified the scheme by a shifted Grünwald approximation and proved that the resulting Meerschaert-Tadjeran scheme

$$\frac{d_i^+}{h^\beta} \sum_{s=0}^i g_s^{(\beta)} u_{i-s+1} + \frac{d_i^-}{h^\beta} \sum_{s=0}^{N-i+1} g_s^{(\beta)} u_{i+s-1} = v_i, \quad 1 \leq i \leq N, \quad (8)$$

$$u_0 = u_{N+1} = 0$$

is unconditionally stable and convergent. However, due to the non-local property of fractional differential operators, the numerical scheme (8) generates a full stiffness matrix, which require $O(N^2)$ memory and $O(N^3)$ computations to solve. In [33] we proved that the stiffness matrix of the Meerschaert-Tadjeran scheme (8) has a Toeplitz-like structure and developed fast solution methods accordingly, which significantly reduced the storage to $O(N)$ and computational cost to $O(N \log N)$ (per iteration in a Krylov subspace iterative method). An optimal preconditioner was developed in [29].

However, again due to the non-local nature of fractional differential operators, the nonlocal homogeneous Dirichlet boundary condition for u in (1) does not ensure the left (or right) sided fractional derivative to vanish for $x \geq 1$ (or $x \leq 0$), or in other words, the homogeneous Dirichlet boundary condition for v . Therefore, one cannot solve the system (7)–(6) sequentially. One still has to solve the system in a coupled fashion. To do so, one has to express v_0 and v_{N+1} in terms of u_i for $i = 1, 2, \dots, N$. Here we approximate them by a direct truncation of (2):

$$v_0 = \frac{d_0^-}{h^\beta} \sum_{s=1}^N g_s^{(\beta)} u_s, \quad v_{N+1} = \frac{d_{N+1}^+}{h^\beta} \sum_{s=1}^N g_s^{(\beta)} u_{N+1-s}. \quad (9)$$

Let u and v be the N and $N + 2$ dimensional column vectors

$$u = (u_1, u_2, \dots, u_N)^T, \quad v = (v_0, v_1, \dots, v_{N+1})^T. \quad (10)$$

Then we rewrite (8) and (9) into a matrix form

$$Bu = v \quad (11)$$

where B is an $(N + 2)$ -by- N stiffness matrix.

Remark 3.1. *The finite difference scheme (8), under the assumption that all the v_i s are known, has been very well analyzed [14, 15, 20]. However, in the current context, all the v_i s are intermediate variables. In fact, together with the discretization (9), the scheme (8) actually leads a rectangular stiffness matrix of size $(N + 2)$ -by- N . Hence, the scheme (8) and (9) is overdetermined. In general, the linear system (11) does not admit a solution.*

3.2. A collocation approximation to (7)

Let $\{\phi_i\}_{i=0}^{N+1}$ be the set of hat functions satisfying $\phi_i(x_j) = \delta_{i,j}$, where $\delta_{i,j}$ is the discrete δ function. Let

$$v(x) = \sum_{j=0}^{N+1} v_j \phi_j(x). \quad (12)$$

We insert (12) into (7) and choose $\{x_i\}_{i=1}^N$ to be the set of collocation points to obtain the following collocation scheme

$$\sum_{j=0}^{N+1} v_j \int_0^1 \Lambda(x_i - y) \phi_j(y) dy = f(x_i), \quad i = 1, 2, \dots, N. \quad (13)$$

(13) can be written in a matrix form

$$Av = f. \quad (14)$$

Here the entries of the N -dimensional vector $f = (f_1, f_2, \dots, f_N)^T$ and the N -by- $(N+2)$ matrix $A = [a_{i,j}]$ are given by

$$\begin{aligned} f_i &= f(x_i), \quad 1 \leq i \leq N, \\ a_{i,j} &= \int_0^1 \Lambda(x_i - y) \phi_j(y) dy, \quad 1 \leq i \leq N, \quad 0 \leq j \leq N+1. \end{aligned}$$

Remark 3.2. *The scheme (13) is actually underdetermined, as the stiffness matrix A is N -by- $(N+2)$. Hence, the problem is expected to have infinitely many solutions, in general.*

Nevertheless, after combining the matrix equations (11) and (14) we obtain the following matrix system

$$ABu = f. \quad (15)$$

Remark 3.3. *Although the linear system (11) is overdetermined and the linear system (12) is underdetermined, the linear system (15), as a composition of both, has a square stiffness matrix. And the numerical experiments show the stability and convergence of the numerical scheme. However, as the stiffness matrix AB of the linear system (15) is expressed as a product of an N -by- $(N+2)$ matrix with an $(N+2)$ -by- N matrix, the stability and convergence analysis of the numerical scheme requires further investigation in the future.*

4. A fast Krylov subspace method for (15) with efficient storage

The N -by- N square matrix AB in the linear system (15) is full and has a complicated structure. Nevertheless, the stiffness matrix can be decomposed as a product of two rectangular matrices A and B . Hence, we need to investigate the structure of A and B .

4.1. The structure and efficient storage of A and B

In this subsection we prove the following theorem on the structure and efficient storage of A and B .

Theorem 4.1. *Suppose that the kernel Λ is defined by (4) or (5). Then the N -by- $(N+2)$ matrix A can be expressed in the form*

$$A = (a, A_N, \tilde{a}) \quad (16)$$

where the N -dimensional column vectors a and $\tilde{a}$, and the N -by- N symmetric Toeplitz matrix A_N are of the form

$$a := (a_{10}, a_{20}, \dots, a_{N0})^T, \quad \tilde{a} := (a_{N0}, \dots, a_{20}, a_{10})^T,$$

and

$$A_N := \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{N-1} \\ a_1 & a_0 & a_1 & \ddots & \vdots \\ a_2 & a_1 & \ddots & \ddots & a_2 \\ \vdots & \ddots & \ddots & a_0 & a_1 \\ a_{N-1} & \cdots & a_2 & a_1 & a_0 \end{pmatrix}.$$

In other words, A can be stored using $2N$ memory.

Proof. A direct calculation shows that A has the form

$$A = \begin{pmatrix} a_{10} & a_0 & a_1 & a_2 & \cdots & a_{N-1} & a_{N0} \\ a_{20} & a_1 & a_0 & a_1 & \ddots & \vdots & \vdots \\ a_{30} & a_2 & a_1 & \ddots & \ddots & a_2 & a_{30} \\ \vdots & \vdots & \ddots & \ddots & a_0 & a_1 & a_{20} \\ a_{N0} & a_{N-1} & \cdots & a_2 & a_1 & a_0 & a_{10} \end{pmatrix}. \quad (17)$$

When Λ is defined by (4), the entries in (17) are given by

$$\begin{aligned} a_0 &= \frac{2h^{1-\alpha}}{(1-\alpha)(2-\alpha)}, \\ a_1 &= \frac{h^{1-\alpha}(2^{2-\alpha} - 2)}{(1-\alpha)(2-\alpha)}, \\ a_2 &= \frac{h^{1-\alpha}(1 - 2 \cdot 2^{2-\alpha} + 3^{2-\alpha})}{(1-\alpha)(2-\alpha)}, \\ &\vdots \\ a_{N-1} &= \frac{h^{1-\alpha}[(N-2)^{2-\alpha} - 2 \cdot (N-1)^{2-\alpha} + N^{2-\alpha}]}{(1-\alpha)(2-\alpha)}, \end{aligned}$$

and

$$\begin{aligned}
a_{10} &= \frac{h^{1-\alpha}}{2-\alpha}, \\
a_{20} &= h^{1-\alpha} \left[\frac{1-2^{2-\alpha}}{(1-\alpha)(2-\alpha)} + \frac{2^{1-\alpha}}{1-\alpha} \right], \\
a_{30} &= h^{1-\alpha} \left[\frac{2^{2-\alpha}-3^{2-\alpha}}{(1-\alpha)(2-\alpha)} + \frac{3^{1-\alpha}}{1-\alpha} \right], \\
&\vdots \\
a_{N0} &= h^{1-\alpha} \left[\frac{(N-1)^{2-\alpha}-N^{2-\alpha}}{(1-\alpha)(2-\alpha)} + \frac{N^{1-\alpha}}{1-\alpha} \right].
\end{aligned}$$

When Λ is defined by (5), the entries in (17) are given by

$$\begin{aligned}
a_0 &= -2 + 2 \left(1 + \frac{a}{h} \right) \ln \frac{a+h}{a}, \\
a_1 &= -\frac{a}{h} \ln \frac{a+h}{a} + \left(2 + \frac{a}{h} \right) \ln \frac{a+2h}{a+h}, \\
a_2 &= -\left(1 + \frac{a}{h} \right) \ln \frac{a+2h}{a+h} + \left(3 + \frac{a}{h} \right) \ln \frac{a+3h}{a+2h}, \\
&\vdots \\
a_{N-1} &= -\left(N-2 + \frac{a}{h} \right) \ln \frac{a+(N-1)h}{a+(N-2)h} \\
&\quad + \left(N + \frac{a}{h} \right) \ln \frac{a+Nh}{a+(N-1)h},
\end{aligned}$$

and

$$\begin{aligned}
a_{10} &= 1 - \frac{a}{h} \ln \frac{a+h}{a}, \\
a_{20} &= 1 - \left(1 + \frac{a}{h} \right) \ln \frac{a+2h}{a+h}, \\
a_{30} &= 1 - \left(2 + \frac{a}{h} \right) \ln \frac{a+3h}{a+2h}, \\
&\vdots \\
a_{N0} &= 1 - \left(N-1 + \frac{a}{h} \right) \ln \frac{a+Nh}{a+(N-1)h}.
\end{aligned}$$

We thus finish the proof of the theorem. As a matter of fact, we have also given the explicit expression of the entries. $\square$

Theorem 4.2. *The $(N+2)$ -by- N matrix B has a Toeplitz-like structure and can be stored in $O(N)$ memory.*

Proof. We note that the matrix B has a similar but different structure from that in [?], as B is not a square matrix. As a matter of fact, we conclude from (8) and (9) that B can be decomposed as follows

$$B = D^+ B_L + D^- B_R, \quad (18)$$

with

$$D^+ = \frac{1}{h^\beta} \text{diag}(d_0^+, d_1^+, \dots, d_{N+1}^+), \quad D^- = \frac{1}{h^\beta} \text{diag}(d_0^-, d_1^-, \dots, d_{N+1}^-), \quad (19)$$

$$B_L = \begin{pmatrix} 0 & 0 & \cdots & \cdots & 0 \\ g_1^{(\beta)} & g_0^{(\beta)} & 0 & \cdots & 0 \\ g_2^{(\beta)} & g_1^{(\beta)} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ g_{N-1}^{(\beta)} & \ddots & \ddots & g_1^{(\beta)} & g_0^{(\beta)} \\ g_N^{(\beta)} & g_{N-1}^{(\beta)} & \cdots & g_2^{(\beta)} & g_1^{(\beta)} \\ g_N^{(\beta)} & g_{N-1}^{(\beta)} & \cdots & g_2^{(\beta)} & g_1^{(\beta)} \end{pmatrix}, \quad (20)$$

$$B_R = \begin{pmatrix} g_1^{(\beta)} & g_2^{(\beta)} & \cdots & g_{N-1}^{(\beta)} & g_N^{(\beta)} \\ g_1^{(\beta)} & g_2^{(\beta)} & \cdots & g_{N-1}^{(\beta)} & g_N^{(\beta)} \\ g_0^{(\beta)} & g_1^{(\beta)} & \ddots & \ddots & g_{N-1}^{(\beta)} \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & g_1^{(\beta)} & g_2^{(\beta)} \\ 0 & \cdots & 0 & g_0^{(\beta)} & g_1^{(\beta)} \\ 0 & \cdots & \cdots & 0 & 0 \end{pmatrix}. \quad (21)$$

Introduce the N -dimensional row vectors

$$0 = (0, 0, \dots, 0), \quad r = (g_1^{(\beta)}, g_2^{(\beta)}, \dots, g_N^{(\beta)}), \quad \tilde{r} = (g_N^{(\beta)}, \dots, g_2^{(\beta)}, g_1^{(\beta)})$$

and an N -by- N Toeplitz matrix

$$G_N = \begin{pmatrix} g_1^{(\beta)} & g_0^{(\beta)} & 0 & \cdots & 0 \\ g_2^{(\beta)} & g_1^{(\beta)} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ g_{N-1}^{(\beta)} & \ddots & \ddots & g_1^{(\beta)} & g_0^{(\beta)} \\ g_N^{(\beta)} & g_{N-1}^{(\beta)} & \cdots & g_2^{(\beta)} & g_1^{(\beta)} \end{pmatrix}.$$

Then the matrices B_L and B_R in (20) and (21) can be expressed in the block form as

$$B_L = \begin{pmatrix} 0 \\ G_N \\ \tilde{r} \end{pmatrix}, \quad B_R = \begin{pmatrix} r \\ G_N^T \\ 0 \end{pmatrix}. \quad (22)$$

Therefore, we need only $3N + 5$ memory to store B . $\square$

4.2. A fast implementation of matrix-vector multiplication ABw

In this subsection we prove the following theorem.

Theorem 4.3. *For any vector $w \in \mathbb{R}^N$, the matrix-vector multiplication ABw can be carried out in $O(N \log N)$ operations.*

Proof. We begin by proving that $v = Bw$ can be performed in $O(N \log N)$ operations for any $w \in \mathbb{R}^N$. It is clear from (18) that it suffices to show that $B_L w$ can be carried out in $O(N \log N)$ operations. Since the dot product $\tilde{r} \cdot w$ can be evaluated in $O(N)$ operations, we conclude from (22) that we need only to show that $G_N w$ can be carried out in $O(N \log N)$ operations.

It is known that the Toeplitz matrix G_N can be embedded into a $(2N) \times (2N)$ circulant matrix C_{2N} as follows [6, 10]

$$C_{2N} = \begin{pmatrix} G_N & \tilde{G}_N \\ \tilde{G}_N & G_N \end{pmatrix}, \quad (23)$$

with

$$\tilde{G}_N = \begin{pmatrix} 0 & g_N^{(\beta)} & \dots & g_3^{(\beta)} & g_2^{(\beta)} \\ 0 & 0 & g_N^{(\beta)} & \ddots & g_3^{(\beta)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & g_N^{(\beta)} \\ g_0^{(\beta)} & 0 & \dots & 0 & 0 \end{pmatrix}.$$

The circulant matrix C_{2N} can be diagonalized by Fourier transform matrix [6, 10]

$$C_{2N} = F_{2N}^{-1} \text{diag}(F_{2N} c) F_{2N}, \quad (24)$$

where c is the first column vector of C_{2N} and F_{2N} is the $(2N) \times (2N)$ discrete Fourier transform matrix. It is well known that the matrix-vector multiplication $F_{2N} w_{2N}$ for $w_{2N} \in \mathbb{R}^{2N}$ can be carried out in $O((2N) \log(2N)) =$

$O(N \log N)$ operations via the fast Fourier transform (FFT). (24) shows that $C_{2N} w_{2N}$ can be evaluated in $O(N \log N)$ operations, so that (23) implies that $G_N w$ can be evaluated in $O(N \log N)$ operations for any N -dimensional vector w . Thus, we have proved that the computational cost of $B_L w$ is $O(N \log N)$. The same estimate holds for $B_R w$, and then from (18) we conclude that $v = Bw$ can be evaluated in $O(N \log N)$ operations.

Next, we denote any $(N+2)$ -dimensional vector v of the form in (10) into a block form

$$v = (v_0, v_N^T, v_{N+1})^T.$$

We obtain from (16) that

$$Av = av_0 + A_N v_N + \tilde{a} v_{N+1}.$$

Since A_N is Toeplitz, we can apply the preceding technique to compute $A_N v_N$, and so Av in $O(N \log N)$ operations. $\square$

5. A preconditioned fast Krylov subspace method

The application of the fast matrix-vector multiplication algorithm to any Krylov subspace iterative solver naturally leads to a fast version of the solver, which reduces computational cost from $O(N^2)$ to $O(N \log N)$ per iteration. This significant reduction of computations also reduces round-off errors and improves the convergence behavior of the underlying Krylov subspace method, as shown in [29] in the context of a fractional diffusion equation.

5.1. Construction of a preconditioner and its fast inversion

Because the coefficient matrix AB is ill conditioned, the number of iterations can grow rapidly as the mesh size is refined. Moreover, the accumulated round-off error may affect the accuracy of the numerical approximations as shown in [29] in the context of fractional differential equations.

In this subsection we propose a preconditioner for the system (15) based on the well developed circulant preconditioners for Toeplitz systems [3, 4, 26]. We take the Strang circulant preconditioner as an example to expose the idea. We note that the matrices A and B are not Toeplitz, in fact they are not even square matrices. To make the best use of the Strang preconditioner, we construct Toeplitz approximations to A and B to which we can apply the Strang preconditioner.

Note that A_N in (16) is a symmetric Toeplitz matrix, we use its Strang circulant preconditioner $S(A_N)$ as a preconditioner for the approximation of A . The construction of a circulant preconditioner for B is more involved than that for A , since B contains space-dependent diffusion coefficients $d^+(x)$ and $d^-(x)$ that form the diagonal matrices D^+ and D^- . Consequently, we observe from (18), (19), and (22) that

$$B_N := D_N^+ G_N + D_N^- G_N^T, \quad (25)$$

where

$$D_N^+ = \frac{1}{h^\beta} \text{diag}(d_1^+, \dots, d_N^+), \quad D_N^- = \frac{1}{h^\beta} \text{diag}(d_1^-, \dots, d_N^-),$$

is not a Toeplitz matrix, so Strang preconditioner cannot be applied. To apply the same idea as in the construction of $S(A_N)$, we approximate the Toeplitz-like matrix in (25) by the following Toeplitz matrix

$$\bar{B}_N := \bar{D}^+ G_N + \bar{D}^- G_N^T$$

where

$$\bar{D}^+ = \frac{1}{Nh^\beta} \sum_{i=1}^N d_i^+, \quad \bar{D}^- = \frac{1}{Nh^\beta} \sum_{i=1}^N d_i^-$$

and use $S(\bar{B}_N)$ as a preconditioner for the approximation of B .

In summary, we use $M = S(A_N)S(\bar{B}_N)$ as a preconditioner for the stiffness matrix AB in the numerical scheme (15). We deduce from the construction of M that it only requires $O(N)$ memory. Finally, we show that the preconditioner M can be inverted in $O(N \log N)$ operations. In fact, we note that the linear system

$$Mx = y$$

can be decomposed as

$$S(A_N)z = y, \quad S(\bar{B}_N)x = z. \quad (26)$$

We conclude from (24) that the two linear circulant systems in (26) can be inverted in $O(N \log N)$ operations.

5.2. A preconditioned fast conjugate gradient squared method

The fast matrix-vector multiplication algorithm and the preconditioner described in the preceding sections can be applied to formulate preconditioned fast Krylov subspace methods. As an example, a preconditioned fast conjugate gradient squared method is formulated as follows [23]:

For a given initial guess $u^{(0)}$, compute

$$r^{(0)} = f - ABu^{(0)}$$

Choose $\tilde{r}$ (for example, $\tilde{r} = r^{(0)}$)

for $i = 1, 2, \dots$

$$\rho_{i-1} = \tilde{r}^T r^{(i-1)}$$

if $\rho_{i-1} = 0$ choose $\tilde{r} := r^{(i-1)}$

if $i = 1$

$$w^{(1)} = r^{(0)}$$

$$e^{(1)} = w^{(1)}$$

else

$$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$$

$$w^{(i)} = r^{(i-1)} + \beta_{i-1} q^{(i-1)}$$

$$e^{(i)} = w^{(i)} + \beta_{i-1} (q^{(i-1)} + \beta_{i-1} e^{(i-1)})$$

end if

Solve $M\hat{e} = e^{(i)}$, where M is the preconditioner

$$\hat{v} = AB\hat{e}$$

$$\alpha_i = \rho_{i-1} / \tilde{r}^T \hat{v}$$

$$q^{(i)} = w^{(i)} - \alpha_i \hat{v}$$

Solve $M\hat{w} = w^{(i)} + q^{(i)}$

$$u^{(i)} = u^{(i-1)} + \alpha_i \hat{w}$$

$$\hat{q} = AB\hat{w}$$

$$r^{(i)} = r^{(i-1)} - \alpha_i \hat{q}$$

$$\delta = \|r^{(i)}\|$$

Check for convergence; continue if necessary

end

$$u = u^{(i)}$$

In the above algorithm, all steps require $O(N)$ operations except for the three matrix-vector multiplications $ABu^{(0)}$, $ABe^{(i)}$, $AB\hat{w}$, and the solutions of the two systems $M\hat{e} = e^{(i)}$, $M\hat{w} = w^{(i)} + q^{(i)}$. The matrix-vector multiplications by AB generally require $O(N^2)$ operations while the solution of the

systems with the coefficient matrix M generally require $O(N^3)$ operations. Nevertheless, with the fast matrix-vector multiplications by AB and the fast inversion of the systems with the coefficient matrix M developed in preceding sections, we can reduce the computations to $O(N \log N)$. The use of the preconditioner M is expected to greatly reduce the number of iterations in the fast CGS algorithm, as observed in the numerical examples in the next section.

Table 1: Errors and convergence rates of the methods solved by Gauss, FCGS, and PFCGS solvers in Example 1 in section 6.1

	Gauss	FCGS	PFCGS	
N	$\ u - u_h\ $	$\ u - u_h\ $	$\ u - u_h\ $	Conv. Rate
2^5	2.2752×10^{-3}	2.2752×10^{-3}	2.2752×10^{-3}	
2^6	1.1315×10^{-3}	1.1315×10^{-3}	1.1315×10^{-3}	1.0078
2^7	5.6072×10^{-4}	5.6072×10^{-4}	5.6072×10^{-4}	1.0129
2^8	2.7800×10^{-4}	2.7800×10^{-4}	2.7800×10^{-4}	1.0122
2^9	1.3806×10^{-4}	1.3806×10^{-4}	1.3806×10^{-4}	1.0098
2^{10}	6.8684×10^{-5}	N/A	6.8684×10^{-5}	1.0073
2^{11}	3.4220×10^{-5}	N/A	3.4220×10^{-5}	1.0051
2^{12}	1.7068×10^{-5}	N/A	1.7068×10^{-5}	1.0035
2^{13}	N/A	N/A	8.5196×10^{-6}	1.0024
2^{14}	N/A	N/A	4.2550×10^{-6}	1.0016
2^{15}	N/A	N/A	2.1259×10^{-6}	1.0011
2^{16}	N/A	N/A	1.0624×10^{-6}	1.0007
2^{17}	N/A	N/A	5.3103×10^{-7}	1.0005

6. Numerical Experiments

In this section we carry out numerical experiments to investigate the performance of the numerical scheme (15), which is solved by the fast CGS (FCGS) algorithm, the preconditioned fast CGS (PFCGS) algorithm method, and Gaussian elimination (Gauss). These methods were implemented using Compaq Visual Fortran 6.6 on a ThinkPad T410 Laptop.

In the numerical example runs, we measure the discrete maximum norm of the global truncation error

$$\|u_h - u\| := \max_{1 \leq i \leq N} |u_i - u(x_i)|.$$

6.1. Numerical simulation of (1) and (4)

We simulate the model problem (1) and (4) with two sets of parameters.

Example 1 We simulate problem (1) and (4) with $\alpha = 0.5$, $\beta = 1.5$, $d^+ = d^- = \Gamma(1.5)$, and the source term given by

$$\begin{aligned} f(x) = & (4x^3 - 6x^2 + 2x) \ln \frac{1 + \sqrt{1-x}}{\sqrt{x}} \\ & + (4(1-x)^3 - 6(1-x)^2 + 2(1-x)) \ln \frac{1 + \sqrt{x}}{\sqrt{1-x}} \\ & + \left(4x^2 - \frac{10}{3}x + \frac{2}{15}\right) \sqrt{1-x} + \left(4(1-x)^2 - \frac{10}{3}(1-x) + \frac{2}{15}\right) \sqrt{x} \\ & + \pi(2x^3 + 2(1-x)^3 - 3x^2 - 3(1-x)^2 + 1). \end{aligned}$$

The true solution to this problem is given by

$$u(x) = x^2(1-x)^2.$$

Table 2: The consumed CPU time and the numbers of iterations of Gauss, FCGS, and PFCGS solvers in Example 1 in section 6.1

	Gauss	FCGS		PFCGS	
N	CPU	CPU	Iter. Num.	CPU	Iter. Num.
2^5	0.000s	0.000s	13	0.000s	6
2^6	0.000s	0.000s	19	0.000s	6
2^7	0.000s	0.016s	28	0.000s	6
2^8	0.094s	0.031s	39	0.000s	6
2^9	0.797s	0.156s	99	0.016s	6
2^{10}	21.125s	N/A	N/A	0.047s	6
2^{11}	4m 56s	N/A	N/A	0.125s	6
2^{12}	44m 11s	N/A	N/A	0.375s	6
2^{13}	N/A	N/A	N/A	1.344s	8
2^{14}	N/A	N/A	N/A	4.703s	8
2^{15}	N/A	N/A	N/A	17.39s	8
2^{16}	N/A	N/A	N/A	1m 5s	8
2^{17}	N/A	N/A	N/A	4m 12s	8

We present the errors and convergence rates of the methods in Table 1 and the CPU cost and the number of iterations (for FCGS and PFCGS) in

Table 2. We note that all the three methods generate numerical solutions with the same accuracy. In the numerical experiments we use the solution by Gauss elimination as the underlying benchmark. We noticed that the CPU time consumed by Gaussian elimination increases about 10 times each time we refine the mesh size. In fact, each time the mesh size is reduced by half, the number of grid points is doubled. Hence, the overall computational work would increase 8 times (asymptotically for very fine mesh sizes). Therefore, we stopped the Gauss elimination for $N = 2^{12}$, when the simulation took about 44 minutes of CPU to finish. The FCGS solver consumed less CPU time than Gaussian elimination. However, the FCGS (and so the CGS) solver diverged for $N \geq 2^{10}$. The PFCGS solver is the most efficient among the three solvers. It used only 6 iterations and 0.375s to finish the simulation at $N = 2^{12}$. Even for a very fine mesh with $N = 2^{17} = 131,072$ interior nodes, the PFCGS solver used only 8 iterations and took only takes 4 minutes 12 seconds to finish the simulation. This shows the strong potential of the PFCGS solver of the numerical scheme (15), which not only significantly reduced the number of iterations and the CPU time of the FCGS solver, but also improved its convergence behavior, as observed in [29] in the context of fractional differential equations.

Example 2 We simulate problem (1) and (4) with $\alpha = 0.5$, $\beta = 1.8$, $d^+ = \Gamma(1.2)x^{0.8}$, $d^- = \Gamma(1.2)(1-x)^{0.8}$, and the source term given by

$$f(x) = \frac{20}{33}[4.8(x^{2.5} + (1-x)^{2.5}) - 8(2x-1)(x^{1.5} + (1-x)^{1.5}) + (24x^2 - 24x + 3.6)(x^{0.5} + (1-x)^{0.5})].$$

The true solution to this problem is the same as the one in Example 1.

We present the errors and convergence rates of the methods in Table 3 and the CPU cost and the number of iterations (for FCGS and PFCGS) in Table 4. We observe roughly the same convergence behavior as those in Example 1, except for the following facts: (i) The number of iterations in the FCGS solver increases much faster than that in Example 1 and diverged for an even coarser mesh size of $N = 2^8$ due to the variable diffusivity coefficients. (ii) Accordingly, we also observed that the PFCGS solver used more iterations, but still managed to converge in 4m 26s for the finest mesh size of $N = 2^{17}$. (iii) At finer mesh sizes of $N = 2^{11}$ and $N = 2^{12}$, the PFCGS solver generates slightly more accurate numerical solutions than Gaussian elimination due to significantly reduced computational work, as observed in [29] in the context of fractional differential equations.

Table 3: Errors and convergence rates of the methods solved by Gauss, FCGS, and PFCGS solvers in Example 2 in section 6.1

	Gauss	FCGS	PFCGS	
N	$\ u - u_h\ $	$\ u - u_h\ $	$\ u - u_h\ $	Conv. Rate
2^5	3.7340×10^{-3}	3.7340×10^{-4}	3.7340×10^{-4}	
2^6	2.1209×10^{-3}	2.1209×10^{-4}	2.1209×10^{-4}	0.8161
2^7	1.0976×10^{-4}	1.0976×10^{-4}	1.0976×10^{-4}	0.9503
2^8	5.5145×10^{-4}	N/A	5.5145×10^{-5}	0.9931
2^9	2.7491×10^{-4}	N/A	2.7491×10^{-5}	1.0043
2^{10}	1.3693×10^{-5}	N/A	1.3693×10^{-5}	1.0055
2^{11}	6.8267×10^{-5}	N/A	6.8263×10^{-6}	1.0042
2^{12}	3.4070×10^{-5}	N/A	3.4062×10^{-6}	1.0030
2^{13}	N/A	N/A	1.7000×10^{-6}	1.0026
2^{14}	N/A	N/A	8.4733×10^{-7}	1.0046
2^{15}	N/A	N/A	4.1992×10^{-7}	1.0128
2^{16}	N/A	N/A	2.0427×10^{-7}	1.0397
2^{17}	N/A	N/A	9.5362×10^{-8}	1.0990

Table 4: The consumed CPU time and the numbers of iterations of Gauss, FCGS, and PFCGS solvers in Example 2 in section 6.1

	Gauss	FCGS		PFCGS	
N	CPU	CPU	Iter. Num.	CPU	Iter. Num.
2^5	0.000s	0.000s	17	0.000s	7
2^6	0.000s	0.000s	30	0.000s	9
2^7	0.000s	0.031s	85	0.000s	9
2^8	0.094s	N/A	N/A	0.016s	9
2^9	0.813s	N/A	N/A	0.031s	9
2^{10}	21.000s	N/A	N/A	0.063s	10
2^{11}	4m 56s	N/A	N/A	0.172s	10
2^{12}	44m 20s	N/A	N/A	0.453s	10
2^{13}	N/A	N/A	N/A	1.438s	10
2^{14}	N/A	N/A	N/A	4.984s	10
2^{15}	N/A	N/A	N/A	17.70s	10
2^{16}	N/A	N/A	N/A	1m 7s	11
2^{17}	N/A	N/A	N/A	4m 26s	16

Table 5: Errors and convergence rates of the methods solved by Gauss, FCGS, and PFCGS solvers in section 6.2

	Gauss	FCGS	PFCGS	
N	$\ u - u_h\ $	$\ u - u_h\ $	$\ u - u_h\ $	Conv. Rate
2^5	4.9069×10^{-4}	4.9069×10^{-4}	4.9069×10^{-4}	
2^6	2.3418×10^{-4}	2.3418×10^{-4}	2.3418×10^{-4}	1.0672
2^7	1.1381×10^{-4}	N/A	1.1381×10^{-4}	1.0409
2^8	5.5879×10^{-5}	N/A	5.5879×10^{-5}	1.0263
2^9	2.7626×10^{-5}	N/A	2.7626×10^{-5}	1.0163
2^{10}	1.3719×10^{-5}	N/A	1.3719×10^{-5}	1.0098
2^{11}	6.8324×10^{-6}	N/A	6.8323×10^{-6}	1.0058
2^{12}	3.4083×10^{-6}	N/A	3.4081×10^{-6}	1.0034
2^{13}	N/A	N/A	1.7016×10^{-6}	1.0021
2^{14}	N/A	N/A	8.4972×10^{-7}	1.0018
2^{15}	N/A	N/A	4.2406×10^{-7}	1.0027
2^{16}	N/A	N/A	2.1140×10^{-7}	1.0043
2^{17}	N/A	N/A	1.0372×10^{-7}	1.0273

6.2. Numerical simulation of (1) and (5)

We simulate the model problem (1) and (5) with $a = 0.00001$, $\beta = 1.8$, the diffusivity coefficients $d^+ = \Gamma(1.2)x^{0.8}$, $d^- = \Gamma(1.2)(1-x)^{0.8}$, and the source term given by

$$\begin{aligned}
 f(x) = & \left[\frac{80}{11}x^2 - \frac{80}{11}(2a+1)x + \left(\frac{80}{11}a^2 + \frac{80}{11}a + \frac{12}{11} \right) \right] \ln \frac{1+a-x}{a} \\
 & + \left[\frac{80}{11}x^2 + \frac{80}{11}(2a-1)x + \left(\frac{80}{11}a^2 - \frac{80}{11}a + \frac{12}{11} \right) \right] \ln \frac{a+x}{a} \\
 & + \left(-\frac{240}{11}x^2 + \frac{240}{11}x - \frac{40}{11}(2a+1) \right).
 \end{aligned}$$

The true solution to this problem is the same as the one in Example 1.

We present the errors and convergence rates of the methods in Table 5 and the CPU cost and the number of iterations (for FCGS and PFCGS) in Table 6. We have roughly the same observations as those in Example 2 in section 6.1. We notice that the FCGS (and so the CGS) solver diverged at an even coarser mesh size of $N = 2^7$. Nevertheless, the PFCGS solver still performed efficiently and reliably as expected.

Table 6: The consumed CPU time and the numbers of iterations of Gauss, FCGS, and PFCGS solvers in section 6.2

	Gauss	FCGS		PFCGS	
N	CPU	CPU	Iter. Num.	CPU	Iter. Num.
2^5	0.000s	0.000s	16	0.000s	7
2^6	0.000s	0.016s	33	0.000s	9
2^7	0.000s	N/A	N/A	0.000s	9
2^8	0.094s	N/A	N/A	0.000s	9
2^9	0.797s	N/A	N/A	0.016s	9
2^{10}	20.906s	N/A	N/A	0.063s	9
2^{11}	4m 55s	N/A	N/A	0.156s	10
2^{12}	44m 7s	N/A	N/A	0.453s	10
2^{13}	N/A	N/A	N/A	1.422s	10
2^{14}	N/A	N/A	N/A	4.938s	10
2^{15}	N/A	N/A	N/A	17.64s	10
2^{16}	N/A	N/A	N/A	1m 6s	11
2^{17}	N/A	N/A	N/A	4m 18s	12

Acknowledgements

This work was supported by the National Natural Science Foundation of China under grants 11371229, 91130010, and 11471194, by the National Science Foundation under Grants EAR-0934747 and DMS-1216923, and by the China Scholarship Council (File No. 201308370102). The authors would like to express their sincere thanks to the referees for their very helpful comments and suggestions, which greatly improved the quality of this paper. The authors would like to express their sincere thanks to Dr. Rina Schumer in Desert Research Institute at Reno, Nevada for the very helpful discussions and suggestions.

References

- [1] R.A. Adams and J.J.F. Fournier, *Sobolev Spaces*, Elsevier, San Diego, 2003.
- [2] D. Benson, S.W. Wheatcraft, and M.M. Meerschaert, The fractional-order governing equation of Lévy motion, *Water Resour. Res.* 36 (2000), 1413–1423.

- [3] R.H. Chan and M.K. Ng, Conjugate gradient methods for Toeplitz systems, *SIAM Review*, 38 (1996), 427–482.
- [4] T. Chan, An optimal circulant preconditioner for Toeplitz systems, *SIAM J. Sci. Stat. Comput.*, 9 (1988), 766–771.
- [5] S. Chen, F. Liu, X. Jiang, I. Turner and V. Anh, A fast semi-implicit difference method for a nonlinear two-sided space-fractional diffusion equation with variable diffusivity coefficients, *Appl. Math. Comp.*, 257 (2015), 591–601.
- [6] P.J. Davis, *Circulant Matrices*, Wiley-Intersciences, New York, 1979.
- [7] M. Doi and S.F. Edwards, *The Theory of Polymer Dynamics*, Clarendon, Oxford, 1986.
- [8] Q. Du, M. Gunzburger, R. Lehoucq, and K. Zhou, Analysis and approximation of nonlocal diffusion problems with volume constraints, *SIAM Review*, 54 (2012), 667–696.
- [9] V.J. Ervin and J.P. Roop, Variational formulation for the stationary fractional advection dispersion equation, *Numer. Methods. Partial Differential Eq* 22 (2005), 558–576.
- [10] R.M. Gray, *Toeplitz and Circulant Matrices: A Review*, Foundations and Trends in Communications and Information Theory, 2(3) (2006), 155–239.
- [11] G.C. Hsiao, P. Kopp, and W. Wendland, A Galerkin-collocation method for some integral equations of the first kind, *Computing*, 25 (1980), 89–130.
- [12] G.C. Hsiao and W. Wendland, A finite element method for an integral equation of the first kind, *J. Math. Anal. Appl.*, 58 (1977), 449–481.
- [13] F. Liu, V. Anh and I. Turner, Numerical solution of the space fractional Fokker-Planck Equation, *Journal of Computational and Applied Mathematics*, 166, 2004, 209–219.
- [14] F. Liu, P. Zhuang, V. Anh, I. Turner and K. Burrage, Stability and Convergence of the difference Methods for the space-time fractional advection-diffusion equation, *Appl. Math. Comp.*, 191 (2007), 12–20.

- [15] F. Liu, P. Zhuang and K. Burrage, Numerical methods and analysis for a class of fractional advection-dispersion models, *Computer & Math. Appl.*, 64 (2012), 2990–3007.
- [16] F. Liu, P. Zhuang, I. Turner, K. Burrage and V. Anh, A new fractional finite volume method for solving the fractional diffusion equation, *Appl. Math. Modelling*, 38 (2014), 3871–3878.
- [17] V.E. Lynch, B.A. Carreras, D. del-Castillo-Negrete, K.M. Ferreira-Mejias, and H.R. Hicks, Numerical methods for the solution of partial differential equations of fractional order, *J. Comput. Phys.* 192 (2003), 406-421.
- [18] S.N. Majumdar and A.J. Bray, Spatial persistence of fluctuating interfaces, *Phys. Rev. Lett.*, 86 (2001), 3700.
- [19] M.M. Meerschaert, H.P. Scheffler, and C. Tadjeran, Finite difference methods for two-dimensional fractional dispersion equation, *J. Comput. Phys.*, 211 (2006) 249–261.
- [20] M.M. Meerschaert and C. Tadjeran, Finite difference approximations for two-sided space-fractional partial differential equations, *Appl. Numer. Math.*, 56 (2006), 80–90.
- [21] Metzler R. and J. Klafter, The restaurant at the end of the random walk: recent developments in the description of anomalous transport by fractional dynamics, *J. Phys. A Math. Gen.*, 37 (2004) R161-R208.
- [22] I. Podlubny, *Fractional Differential Equations*, Academic Press, New York, 1999.
- [23] Y. Saad, *Iterative methods for sparse linear systems*, 2nd ed., SIAM, Rhode Island, 2003.
- [24] Samko, S., A. Kilbas, and O. Marichev, *Fractional Integrals and Derivatives: Theory and Applications*, Gordon and Breach, London, 1993.
- [25] S. Silling, Reformulation of elasticity theory for discontinuities and long-range forces, *J. Mech. Phys. Solids*, 48 (2000), 175–209.
- [26] G. Strang, A proposal for Toeplitz matrix calculations, *Stud. Appl. Math.*, 74 (1986), 171–176.

- [27] A. Taloni, A. Chechkin, and J. Klafter, Generalized elastic model yields a fractional Langevin equation description, *Phys., Rev. Lett.*, 104 (2010), 160602.
- [28] H. Wang and T.S. Basu, A fast finite difference method for two-dimensional space-fractional diffusion equations, *SIAM J. Sci. Comput.*, 34 (2012), A2444–A2458.
- [29] H. Wang and N. Du, A superfast-preconditioned iterative method for steady-state space-fractional diffusion equations, *J. Comput. Phys.*, 240 (2013), 49–57.
- [30] H. Wang and N. Du, Fast alternating-direction finite difference methods for three-dimensional space-fractional diffusion equations, *J. Comput. Phys.*, 258 (2013), 305–318.
- [31] H. Wang and H. Tian, A fast Galerkin method with efficient matrix assembly and storage for a peridynamic model, *J. Comput. Phys.*, 231 (2012), 7730–7738.
- [32] H. Wang and H. Tian, A fast and faithful collocation method with efficient matrix assembly for a two dimensional peridynamic model, *Comput. Methods Appl. Mech. Engrg.*, 273C (2014), 19–36.
- [33] H. Wang, K. Wang, and T. Sircar, A direct $O(N \log^2 N)$ finite difference method for fractional diffusion equations, *J. Comput. Phys.*, 229 (2010), 8095–8104.
- [34] Q. Yang, I. Turner, F. Liu and M. Ilis, Novel numerical methods for solving the time-space fractional diffusion equation in 2D, *SIAM J. Sci. Comput.*, 33 (2011) 1159–1180.
- [35] Q. Yang, I. Turner, T. Moroney and F. Liu, A finite volume scheme with preconditioned Lanczos method for two-dimensional space-fractional reaction-diffusion equations, *Appl. Math. Modelling*, 38 (2014), 3755–3762.
- [36] M. Zheng, F. Liu, I. Turner and V. Anh, A novel high order space-time spectral method for the time-fractional Fokker-Planck equation, *SIAM J. Sci. Comput.*, 37, (2015), A701–A724.

- [37] A.G. Zilman and R. Granek, Membrane dynamics and structure factor,
Chem. Phys., 284 (2002), 195–204.