A Preconditioned Fast Collocation Method for a Linear Nonlocal Diffusion Model in Convex Domains

XUHAO ZHANG1, AIJIE CHENG2, AND HONG WANG3

1School of Mathematics and Statistics, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250100, China
2School of Mathematics, Shandong University, Jinan 250100, China
3Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA

Corresponding author: Xuhao Zhang (zhanghaotimmy@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 91630207 and Grant 11971272.

ABSTRACT Recently, there are many papers dedicated to develop fast numerical methods for nonlocal diffusion and peridynamic models. However, these methods require the physical domain where we solve the governing equations is rectangular. To relax this restriction, in this article, we develop a fast collocation method for a static linear nonlocal diffusion model in convex domains via a volume-penalization approach. Based on the analysis of the structure of the coefficient matrix, we also present a fast solution technique of the linear system arising from the collocation discretization by accelerating the matrix-vector multiplications in the usual Krylov subspace iteration method. This technique helps to reduce the computational work in each Krylov subspace iteration from $O(N^2)$ to $O(N \log N)$ and the storage requirement of the coefficient matrix from $O(N^2)$ to $O(N)$ without any lossy compression, where $N$ is the number of unknowns. Moreover, an efficient preconditioner was also provided in this work to accelerate the convergence of the Krylov subspace iteration method. Numerical experiments show the applicability of this method.

INDEX TERMS Convex domains, nonlocal diffusion model, collocation method, fast method for the nonlocal model.

I. INTRODUCTION

In the last decades, nonlocal models have been extensively studied in many fields such as fluid dynamics [1], materials science [2], thermodynamics [3], fracture mechanics [4], biology [5] and image processing [6]. One of the nonlocal models is fractional partial differential equations (FPDEs). The other one that arouse researcher’s interests is nonlocal diffusion and peridynamic models. These nonlocal models all contain integral operators. For the new developments in usage of integral equations (especially the usage in FPDEs), we can refer to [7]–[11]. For nonlocal diffusion and peridynamic models, they have been applied and showed the effectiveness in the fields of mechanics of materials, including failure and damage in composite materials [12], [13], multiscale materials modeling [14], phase transformations in solids [15], crack nucleation [16] and so on. Nonlocal diffusion and peridynamic models can often provide a more natural description of physical problems especially near singularities and discontinuities than the PDE models [17], [18]. In contrast to the classical PDEs, the nonlocal diffusion and peridynamic models are free of spatial derivatives and feature interactions that occur between material points separated by a finite distance.

However the nonlocality increases the computational cost and the storage compared to the PDE models. It is mainly because the coefficient matrices arising from some numerical methods such as finite element method and the collocation method are often dense or full matrices, which usually require $O(N^2)$ storage and $O(N^2)$ operations in each Krylov subspace iteration. Nowadays, several numerical methods have been developed and analyzed for the nonlocal diffusion and peridynamic models to reduce the computational work and the storage requirement of the coefficient matrices. References [19] and [20] developed a fast Galerkin method and a discontinuous Galerkin method for a peridynamic model, respectively. References [21], [22] developed a fast collocation method for one-dimensional and two-dimensional peridynamic models, respectively. A fast finite difference method was developed for peridynamic model in [23]. A fast and accurate

Received July 24, 2020, accepted September 8, 2020, date of publication September 28, 2020, date of current version October 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027247
implementation of Fourier spectral approximation was developed for nonlocal diffusion model in [24]. In particular, [22] developed a fast collocation method for a two-dimensional linear nonlocal diffusion model. This method successfully got \( O(N) \) storage of the coefficient matrix and \( O(N \log N) \) operations per Krylov subspace iteration. However, this fast collocation method only holds for problems on rectangular domain.

To ease the similar restriction in developing the fast methods for fractional PDEs, [25] developed a fast finite volume method for conservative space-fractional diffusion equations in convex domains via a penalization approach. In addition, In [26], a fast finite difference method is developed for solving space-fractional diffusion equations with variable coefficient in convex domains using a volume penalization approach. The numerical experiments in [25] and [26] show the utility of this approach.

In this article, we follow the work started in [22] and develop a fast collocation method for a linear nonlocal diffusion model on convex domain via a volume-penalization technique. We extend the original nonlocal diffusion model on convex domain to a new model on standard rectangular domain by introducing a penalization term to the original model. The latter domain is required to contain the former physical convex domain. Furthermore, based on the analysis of the structure of the coefficient matrix, we present a fast solution technique of the linear system arising from the collocation discretization of the extended model by accelerating the matrix-vector multiplications in the usual Krylov subspace iteration method. This technique helps to reduce the computational work in each Krylov subspace iteration from \( O(N^2) \) to \( O(N \log N) \) and the storage requirement of the coefficient matrix from \( O(N^2) \) to \( O(N) \) without any lossy compression. In addition, to further accelerate the convergence of the Krylov subspace iteration method and reduce the CPU time, we introduce an efficient preconditioner which is easy to invert.

The rest of the paper is organized as follows. In Sect. 2, we extend the model problem on convex domain to the penalized problem via a volume penalization method and give the bilinear collocation method of the penalized problem. In Sect. 3, we analyze the structure of coefficient matrix obtained from Sect. 2. Based on this analysis, the operations in each Krylov subspace iteration and the storage of the coefficient matrix can be reduced. In Sect. 4, an efficient preconditioner is provided. In Sect. 5 and Sect. 6, we carry out some numerical experiments and draw some conclusion remarks, respectively.

### II. A VOLUME-PENALIZED COLLOCATION METHOD FOR THE LINEAR NONLOCAL DIFFUSION MODEL IN CONVEX DOMAIN

In this work, we consider the following two-dimensional steady-state linear nonlocal diffusion model:

\[
\int_{B_2(x,y)} \sigma(x - x', y - y')(u(x, y) - u(x', y'))dx'dy' = f(x, y), \quad (x, y) \in \Omega,
\]

\[
u(x, y) = g(x, y), \quad (x, y) \in \Gamma_c.
\]

Here \( \Omega = (a_1(y), b_1(y)) \times (a_2(x), b_2(x)) \) is a (not necessarily rectangular) convex domain; \( \sigma(x, y) \) is the kernel function; \( \delta \) is the horizon parameter of the material which determines the range of interactions; \( \Gamma_c \) denotes a boundary zone surrounding \( \Omega \) with width \( \delta \); \( u(x, y) \) and \( f(x, y) \) represent the diffusing material density and the prescribed source term, respectively. \( B_2(x, y) \) represents an open disk centered at \((x, y) \) with radius \( \delta \), i.e.,

\[
B_2(x, y) = \{(x', y') \in \Omega \cup \Omega_c : \sqrt{(x - x')^2 + (y - y')^2} < \delta \}.
\]

For clarity, we present the sketch of computational domain as follows:

We then follow the idea in [27] to adopt a volume penalization method to extend problem (1) to the following problem on an extended rectangular domain \( \Omega_r = (\hat{a}_1, \hat{b}_1) \times (\hat{a}_2, \hat{b}_2) \supset \Omega \)

\[
\int_{B_2(x,y)} \sigma(x - x', y - y')(u_\eta(x, y) - u_\eta(x', y'))dx'dy'
+ \frac{1 - \eta(x,y)}{\eta} (u_\eta - g(x, y)) = \hat{f}(x, y), \quad (x, y) \in \Omega_r,
\]

\[
u_\eta(x, y) = \hat{g}(x, y), \quad (x, y) \in \Gamma_{rc}.
\]

Here \( \eta \) represents the penalization parameter; \( \hat{f}(x, y) \) is the character function of \( \Omega \), which equals to 1 if \((x, y) \in \Omega \) or zero otherwise; \( \Gamma_{rc} \) is a boundary zone surrounding \( \Omega_r \) with width \( \delta \); \( \hat{f}(x, y) \) and \( \hat{g}(x, y) \) are extensions of \( f(x, y) \) and \( g(x, y) \) from \( \Omega \) to \( \Omega_r \), and from \( \Gamma_c \) to \( \Gamma_{rc} \), respectively. For example, zero extensions are used in the numerical experiments of this article. Heuristically, we anticipate to have

\[
\lim_{\eta \to 0^+} u_\eta(x, y) = g(x, y), \quad (x, y) \in \Omega_r \setminus \Omega.
\]

Therefore, the nonlocal diffusion model (2) reduces to (1) if \((x, y) \in \Omega \).

Let the positive integers \( N_x \) and \( N_y \) denote the number of mesh grids in the \( x \) and \( y \) directions, respectively. We define a spatial partition on \( \Omega_r \) by \( x_i := \hat{a}_1 + ih_x \) for \( i = 0, 1, \cdots, N_x \) and \( y_j := \hat{a}_2 + jh_y \) for \( j = 0, 1, \cdots, N_y \), where \( h_x := (b_1 - \hat{a}_1)/N_x \) and \( h_y := (b_2 - \hat{a}_2)/N_y \) be the mesh sizes in the \( x \) and \( y \) directions, respectively. To handle the nodes...
Here, 1 can be given by two dimensional pyramid function \( f \) and \( \psi \leq \frac{x-x_0}{h_x} \) or \( \frac{y-y_0}{h_y} \).

Because for the spatial nodes \((x_i, y_j)\) are known, we choose the spatial nodes \((x_i, y_j)\) for \( i = 1, 2, \ldots, N_x - 1 \) and \( j = 1, 2, \ldots, N_y - 1 \) as collocation points. This gives the following collocation formulation for problem (2)

\[
\int_{B_k(x_i, y_j)} \sigma(x - x', y - y') (u_{\eta}(x_i, y_j) - u_{\eta}(x', y')) dx' dy' + \frac{1 - 1_{\Omega(x_i, y_j)}}{\eta} (u_{\eta}(x_i, y_j) - g(x_i, y_j)) = \hat{f}(x_i, y_j),
\]

\[
1 \leq i \leq N_x - 1, 1 \leq j \leq N_y - 1.
\]

Substituting from (6) into (7) results in

\[
\int_{B_k(x_i, y_j)} \sigma(x - x', y - y') (u_{\eta}(x_i, y_j) - u_{\eta}(x', y')) dx' dy' + \frac{1 - 1_{\Omega(x_i, y_j)}}{\eta} (u_{\eta}(x_i, y_j) - g(x_i, y_j)) = \hat{f}(x_i, y_j),
\]

\[
1 \leq i \leq N_x - 1, 1 \leq j \leq N_y - 1.
\]

Here, \( 1_{\Omega(x_i, y_j)} := 1_{\Omega(x_i, y_j)} \); \( g_{ij} := g(x_i, y_j); \hat{f}_{ij} := \hat{f}(x_i, y_j) \).

Let \( N := (N_x - 1) \times (N_y - 1) \) be the number of collocation points at which the values of \( u_{\eta} \) need to be computed. Let \( u_{\eta} \) and \( f \) be \( N \)-dimensional vectors defined by

\[
u_{\eta} := [u_{\eta_{1,1}}, \ldots, u_{\eta_{N_x-1,1}}, u_{\eta_{1,2}}, \ldots, u_{\eta_{N_x-1,2}}, u_{\eta_{1,N_y-1}}, \ldots, u_{\eta_{N_x-1,N_y-1}}],
\]

and

\[
f := [f_{1,1}, \ldots, f_{N_x-1,1}, f_{1,2}, \ldots, f_{N_x-1,2}, f_{1,N_y-1}, \ldots, f_{N_x-1,N_y-1}].
\]

Then, the collocation scheme (8) can be expressed as a matrix form

\[
Bu_{\eta} := (A + P)u_{\eta} = f.
\]

Here, \( A \in \mathbb{R}^{N \times N} \) and \( P \in \mathbb{R}^{N \times N} \) are associated with the first term and the second term of the left-hand side of (8), respectively. We consider the matrix \( P \) firstly. The matrix \( P \) is referred to as the penalization matrix. From (8), we can obtain that the matrix \( P \) is a diagonal matrix, with the entries

\[
P_{m,m} := \begin{cases} \frac{1}{\eta}, & (x_i, y_j) \in \Omega \setminus \Omega \setminus \widehat{\Omega} \\ 0, & (x_i, y_j) \in \Omega \end{cases}
\]

For the matrix \( A \), each entry \( A_{m,n} \) for \( m = 1, \ldots, N \) and \( n = 1, \ldots, N \) in it is defined as

\[
A_{m,n} := \int_{B_k(x_i, y_j)} \sigma(x - x', y - y')(\delta_{m,n} - \phi(x', y')) dx' dy'.
\]

Here, the global indices \( m \) and \( n \) are related to the local indices \((i, j)\) and \((i', j')\) by

\[
m = (j - 1) \times (N_x - 1) + i,
\]

\[
1 \leq i \leq N_x - 1, 1 \leq j \leq N_y - 1,
\]

\[
n = (j' - 1) \times (N_x - 1) + i',
\]

\[
1 \leq i' \leq N_x - 1, 1 \leq j' \leq N_y - 1.
\]

\[
\delta_{m,n} := 1 \text{ if } m = n \text{ and zero otherwise.}
\]

The entries \( f_m \) for \( m = 1, \ldots, N \) in the right-hand side vector \( f \) are given by

\[
f_m = \hat{f}_{ij} + \frac{1 - 1_{\Omega(x_i, y_j)}}{\eta} g_{ij}
\]

\[
+ \sum_{i' = -K+1}^{L+1} \sum_{j' = -L+1}^{L+1} u_{\eta_{i',j'}} f(x', y') dx' dy',
\]

Here, the indices \( m \) and \( n \) as well as the indices \((i, j)\) and \((i', j')\) are related by (14) and the indices \((i', j')\) refer to the nodes on the boundary zone \( \Omega \).

Because of the nonlocality of the nonlocal diffusion model, the matrix \( A \) in (11) is usually dense since \( K \) and \( L \) tend to infinity as \( h_x \) and \( h_y \) tend to zero, for which the widely used direct solvers require \( O(N^2) \) memory and \( O(N^3) \) computational work to solve the linear system (11). In the next section, we will analyze the structure of the matrix \( B \). Accordingly, we can significantly reduce the memory requirement of the coefficient matrix \( B \) and the computational work for solving the linear system (11).

III. STRUCTURE OF THE COEFFICIENT MATRIX

Next we analyse the structure of the matrix \( A \).

**Theorem 1:** The matrix \( A \) can be expressed as a \( N_y - 1 \)-by-\( N_y - 1 \) banded-block Toeplitz matrix with a block
Here each entry $A_{i,j}^{i',j'}$ of $A^{i',j'}$ represents the interaction between the $i$-th node in the $j$-th row and the $i'$-th node in the $j'$-th row.

Combining (12) and (13), we have $A_{m,n} \neq 0$ if and only if

$\supp(\phi_{i,j}) \cap B_3(x_i, y_j) \neq \emptyset.$

We observe from (18) and (20) that all the matrix blocks $A^{i,j}$ with $|j-j'| > L$ vanish. Thus $A$ is a block-banded matrix with a block bandwidth $2L + 1$

$$A = \begin{pmatrix}
A_{1,1} & 0 & \cdots & 0 & \cdots & 0 \\
0 & A_{2,1} & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\
0 & \cdots & \cdots & A_{N_x-1,N_y-1} & \cdots & 0 \\
0 & \cdots & \cdots & \cdots & A_{N_x-1,N_y-1} & 0 \\
\end{pmatrix}.$$  

From (19) and (20), we also observe that all the entries $A_{i,j}^{i',j'}$ with $|i-i'| > K$ in $A^{i',j'}$ vanish. That is each matrix block $A^{i,j}$ with $|j-j'| \leq L$ is a banded matrix with bandwidth $2K + 1$

$$A^{i,j} = \begin{pmatrix}
A_{1,1}^{i,j} & 0 & \cdots & 0 & \cdots & 0 \\
0 & A_{2,1}^{i,j} & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\
0 & \cdots & \cdots & A_{N_x-1,N_y-1}^{i,j} & \cdots & 0 \\
0 & \cdots & \cdots & \cdots & A_{N_x-1,N_y-1}^{i,j} & 0 \\
\end{pmatrix}. $$

By the transformations

$$\xi_1 = x' - x_i, \quad \xi_2 = y' - y_j,$$

the pyramid function given in (5) can be written as

$$\phi_{i,j}(x',y') = \psi\left(\frac{x' - x_i}{h_x}\right)\psi\left(\frac{y' - y_j}{h_y}\right)$$

$$= \psi\left(\frac{\xi_1 - x_i - j}{h_x}\right)\psi\left(\frac{\xi_2 - y_j - i}{h_y}\right)$$

$$= \phi_{i'-j'-j}(\xi_1, \xi_2).$$

Then, equation (12) can be deduced to

$$A_{m,n} = \int_{B_3(0,0)} \sigma(-\xi_1, -\xi_2)(\delta_{m,n} - \phi_{i'-i'-j}(\xi_1, \xi_2))d\xi_1d\xi_2.$$  

(23)
Let $j_1' - j_1 = j_2' - j_2 = l, -L \leq l \leq L$. That is, the matrix blocks $A_{i_1,j_1}$ and $A_{i_2,j_2}$ are on the same diagonal. Let

$$m_1 = (j_1 - 1)(N_x - 1) + i,$$

$$1 \leq i \leq N_x - 1, 1 \leq j_1 \leq N_y - 1,$$

$$n_1 = (j_1' - 1)(N_x - 1) + i',$$

$$1 \leq i \leq N_x - 1, 1 \leq j_1' \leq N_y - 1,$$

$$m_2 = (j_2 - 1)(N_x - 1) + i,$$

$$1 \leq i \leq N_x - 1, 1 \leq j_2 \leq N_y - 1,$$

$$n_2 = (j_2' - 1)(N_x - 1) + i',$$

$$1 \leq i \leq N_x - 1, 1 \leq j_2' \leq N_y - 1.$$  \hspace{1cm} (24)

Then we observe that for $1 \leq i, i' \leq N_x - 1$

$$A_{i,i'}^{1,1} = A_{m_1,n_1}^{1,1},$$

$$= \int_{B_0(0,0)} \sigma(\xi_1, \xi_2)(\delta_{m_1,n_1} - \phi_y^{j_1' - j_1}(\xi_1, \xi_2))d\xi_1 d\xi_2$$

$$= \int_{B_0(0,0)} \sigma(\xi_1, \xi_2)(\delta_{m_2,n_2} - \phi_y^{j_1' - j_1}(\xi_1, \xi_2))d\xi_1 d\xi_2$$

$$= A_{m_2,m_2} = A_{i,i'}^{2,2}.$$ \hspace{1cm} (25)

According to (25), we have proved $A_{i_1,j_1}^{1,1} = A_{i_2,j_2}^{2,2}$.

Let $j_3' - j_3 = i_3' - i_4 = k, -K \leq k \leq K$. That is, the entries $A_{i_3,j_3}^{1,1}$ and $A_{i_4,j_4}^{1,1}$ in each matrix block $A_{i,j}$ are on the same diagonal. Let

$$m_3 = (j - 1)(N_x - 1) + i_3,$$

$$1 \leq i_3 \leq N_x - 1, 1 \leq j \leq N_y - 1,$$

$$n_3 = (j - 1)(N_x - 1) + i_3',$$

$$1 \leq i_3 \leq N_x - 1, 1 \leq j \leq N_y - 1,$$

$$m_4 = (j - 1)(N_x - 1) + i_4,$$

$$1 \leq i_4 \leq N_x - 1, 1 \leq j \leq N_y - 1,$$

$$n_4 = (j - 1)(N_x - 1) + i_4',$$

$$1 \leq i_4 \leq N_x - 1, 1 \leq j \leq N_y - 1.$$ \hspace{1cm} (26)

Then we observe that

$$A_{i_3,j_3}^{1,1} = A_{m_3,n_3}^{1,1},$$

$$= \int_{B_0(0,0)} \sigma(-\xi_1, -\xi_2)(\delta_{m_3,n_3} - \phi_y^{j_3' - j_3}(\xi_1, \xi_2))d\xi_1 d\xi_2$$

$$= \int_{B_0(0,0)} \sigma(-\xi_1, -\xi_2)(\delta_{m_4,n_4} - \phi_y^{j_3' - j_3}(\xi_1, \xi_2))d\xi_1 d\xi_2$$

$$= A_{m_4,n_4} = A_{i_4,j_4}^{1,1}.$$ \hspace{1cm} (27)

According to (27), we obtain that each block matrix $A_{i,j}$ is a banded Toeplitz matrix. Combining (25) and (27), we conclude that the matrix $A$ is a block-Toeplitz-Toeplitz-block (BTTB) matrix and can be expressed as the form showed in (16) and (17).

**Theorem 2:** The coefficient matrix $B$ can be stored in $O(N)$ memories.

**Proof:** From Theorem 3.1 we can find that the matrix $A$ can be stored only by storing the following $(2K + 1)$-by-$(2L + 1)$ entries

$$G = \begin{pmatrix}
I_{-K,-L} & \cdots & I_{-K,0} & \cdots & I_{-K,L} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
I_{0,-L} & \cdots & I_{0,0} & \cdots & I_{0,L} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
I_{K,-L} & \cdots & I_{K,0} & \cdots & I_{K,L}
\end{pmatrix},$$ \hspace{1cm} (28)

Hence, $A$ can be stored in $O((2K + 1) \times (2L + 1)) = O(N)$ memories. Together with the fact that the penalization matrix $P$ is diagonal, we can find that the stiffness matrix $B$ can be stored in $O(N)$ memories.

**Theorem 3:** For any vector $u \in \mathbb{R}^N$, the matrix-vector multiplication $Bu$ can be carried out in $O(N \log N)$ operations.

**Proof:** On the one hand, because the BTTB structure of the matrix $A$, the matrix-vector multiplication $Au$ for any $u \in \mathbb{R}^N$ can be computed in $O(N \log N)$ operations [28], [29]. On the other hand, the matrix-vector multiplication $Pu$ can be computed in $O(N)$ operations since the matrix $P$ is diagonal. In conclusion, the total matrix-vector multiplication $Bu = Au + Pu$ can be evaluated in $O(N \log N)$ operations.

From Theorem 4.2, we can reduce the memory requirement of the coefficient matrix $B$ from $O(N^2)$ to $O(N)$. Moreover, using Theorem 4.3, we can reduce the computational work of solving the linear system from $O(N^2)$ to $O(N \log N)$ per Krylov subspace iteration.

**IV. A PRECONDITIONED FAST KRYLOV SUBSPACE METHOD**

In the Krylov subspace iteration method, each iteration consists of matrix-vector multiplications and related vector operations. Because of the nonlocal property of the nonlocal diffusion model, the stiffness matrix $B$ is usually dense. Hence, the evaluation of the matrix-vector multiplication in each iteration requires $O(N^2)$ computational work. All other computations in each iteration still require $O(N)$ computational work. Thus, a fast Krylov subspace method can be developed due to the fast matrix-vector multiplication proved in the previous section. It can reduce the computational work from $O(N^2)$ to $O(N \log N)$ per Krylov subspace iteration. Furthermore, from Theorem 3.1, we can see that the memory requirement can also be reduced from $O(N^2)$ to $O(N)$. However, the number of iterations still be large due to the singularity of the kernel function $\sigma$. This would in turn increases the overall computational cost and might even affect the convergence behavior of a Krylov subspace iterative solver.

Accordingly, we need to introduce an effective preconditioner to reduce the number of iterations. We consider the following preconditioner.

To begin with, because of the BTB structure of the matrix $A$, we can apply the definition of the T. Chan’s
circuit matrix to $T_j$ for $-L \leq j \leq L$ to construct a block-Toeplitz-Circulant-block (BTCB) matrix [28],[29]. More precisely, we obtain the matrix as follows

$$C = \begin{pmatrix} C_0 & \cdots & C_L & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ C_{-L} & \cdots & C_0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & C_0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ 0 & \cdots & 0 & C_0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ 0 & \cdots & 0 & 0 & \cdots & C_{-L} & \cdots & C_0 \\ \end{pmatrix}$$

(29)

Here each $C_l$ with $-L \leq l \leq L$ is defined as

$$C_l = \begin{pmatrix} c_{0,l} & c_{1,l} & \cdots & c_{N_x-3,l} & c_{N_x-2,l} \\ c_{-1,l} & c_{0,l} & \cdots & c_{N_x-3,l} & c_{N_x-2,l} \\ \vdots & \ddots & \ddots & \vdots & \ddots \\ c_{3-N_x,l} & \cdots & c_{0,l} & c_{1,l} \\ c_{2-N_x,l} & c_{3-N_x,l} & \cdots & c_{1,l} & c_{0,l} \\ \end{pmatrix},$$

(30)

where the diagonals of $C_l$ are given by

$$c_{k,l} = \begin{cases} (N_x - 1 - k)y_{k,l} + kN_x - 1 - k, & 0 \leq k \leq N_x - 2, \\ c_{1-N_x,k}, & 2 - N_x \leq k < 0. \\ \end{cases}$$

(31)

Then using (29) and the definition of T.Chan’s circulant matrix, we obtain the following block-circulant-circular-block (BCCB) matrix

$$\tilde{C} = \begin{pmatrix} \tilde{C}_0 & \tilde{C}_1 & \cdots & \tilde{C}_{N_y-3} & \tilde{C}_{N_y-2} \\ \tilde{C}_1 & \tilde{C}_0 & \cdots & \tilde{C}_{N_y-3} & \tilde{C}_{N_y-2} \\ \vdots & \ddots & \ddots & \vdots & \ddots \\ \tilde{C}_{2-N_y} & \tilde{C}_{3-N_y} & \cdots & \tilde{C}_{0} & \tilde{C}_1 \\ \tilde{C}_{3-N_y} & \tilde{C}_{2-N_y} & \cdots & \tilde{C}_{1} & \tilde{C}_0 \\ \end{pmatrix},$$

(32)

where each circulant matrix block $\tilde{C}_{-k}$ with $2 - N_y \leq k \leq N_y - 2$ is defined as

$$\tilde{C}_{-k} = \begin{pmatrix} (N_y - 1 - k)C_{-k} + kC_{N_y-1-k} \\ N_y - 1 \\ C_{1-N_y-k}, & 2 - N_y \leq k < 0. \\ \end{pmatrix}$$

(33)

The matrix $\tilde{C}$ is $N_y - 1$ by $N_y - 1$ block-circular matrix with $N_x - 1$ by $N_y - 1$ circular blocks. It can be efficiently stored in $O(N)$ memory and inverted in $O(N \log N)$ operations.

A straightforward choice of preconditioner for the coefficient matrix $B$ is

$$B = \tilde{C} + P,$$

(34)

where $P$ is the penalty matrix whose entries equal to $0$ or $1/\eta$, depending on the geometric structure of the domains $\Omega$ and $\Omega_r$. However, it’s difficult to inverse the preconditioner $B$.

We then follow the idea in [30] to construct an efficient preconditioner.

Let $\Theta := \{1, 2, \cdots, N\}, \Theta_c := \{i | i \in \Theta, P_{l,i} = 0\}$ and $\Theta_r := \{i | i \in \Theta, P_{l,i} = 1/\eta\}$. Define

$$K_i = \tilde{C} + P_{l,i}I, \quad i = 1, 2, \cdots, N.$$

(35)

In fact, $(K_{ii})_{i=1}^N$ have only two values defined by

$$K_c = \tilde{C}$$

(36)

and

$$K_r = \tilde{C} + \frac{1}{\eta}I.$$ (37)

By (34) and (35), we can obtain

$$e_i^T B e_i = e_i^T K_i,$$

(38)

in which the vector $e_i$ represents the $i$-th column of the identity matrix. Thus we have

$$B^{-1} = \sum_{i=1}^N e_i e_i^T B^{-1} \approx \sum_{i=1}^N e_i e_i^T K_i^{-1} = \sum_{i \in \Theta_c} e_i e_i^T K_i^{-1} + \sum_{i \in \Theta_r} e_i e_i^T K_r^{-1} = \Phi_c K_c^{-1} + \Phi_r K_r^{-1},$$

(39)

where

$$\Phi_c = \sum_{i \in \Theta_c} e_i e_i^T = diag(\phi_c^1, \phi_c^2, \cdots, \phi_c^N),$$

$$\Phi_r = \sum_{i \in \Theta_r} e_i e_i^T = diag(\phi_r^1, \phi_r^2, \cdots, \phi_r^N),$$

$$\phi_i^j = \begin{cases} 1, & i \in \Theta_c, \\ 0, & i \in \Theta_r. \end{cases}$$

(40)

Then we obtain the practical preconditioner $B_1$ whose inverse can be expressed as

$$B_1^{-1} = \Phi_c K_c^{-1} + \Phi_r K_r^{-1}.$$ (41)

Note that $K_c$ and $K_r$ are BCCB matrices, each inverse of these two matrices can be computed in $O(N \log N)$ operations by fast Fourier transform method (FFT). Thus the inverse of the preconditioner $B_1$ can be computed in $O(N \log N)$ operations.
V. NUMERICAL EXPERIMENTS

In this section, we carry out several numerical experiments to investigate the performance of the fast collocation method for the nonlocal diffusion model in convex domains. In the numerical runs as follows, we consider the nonlocal diffusion model (1) with the kernel function

\[ \sigma(x, y) = \frac{1}{(x^2 + y^2)^{1+\epsilon}}. \]  

(42)

In the following numerical experiments, the convex domain \( \Omega \) in (1) is chosen to be a circle \( \Omega = \{(x, y) : (x - 0.5)^2 + (y - 0.5)^2 < 0.21^2\} \). The horizon parameter \( \delta = 1/8 \). The true solution \( u(x, y) = x(1-x)y(1-y) \). The right-hand term \( f(x, y) \) can be computed using polar coordinate transformation as follows:

\[ f(x, y) = \frac{\pi \delta^2 - 2\delta}{2 - 2\delta} (x + y - x^2 - y^2) - \frac{\pi \delta^4 - 2\delta}{16 - 8\delta}. \]  

(43)

The true solution is also used to define \( g(x, y) \) on the boundary zone \( \Gamma_c \). We use MATLAB to run all the numerical experiments on a 4G-memory laptop. The code is home-made and the pick speed of CPU is 1.8 GHz.

**Example 1:** In this numerical example, we investigate the convergence behavior of the volume-penalized fast collocation method by choosing different kernel functions. The value of \( s \) in (42) is chosen to be \( 3/8 \). We use MATLAB to run all the numerical experiments on a 4G-memory laptop. The code is home-made and the pick speed of CPU is 1.8 GHz.

As discussed in Section 2, in order to solve the problem (1), we turn to solve the penalized problem (2). Therefore, we extend the physical convex domain \( \Omega \) to a rectangular domain \( \Omega_r = (0, 1) \times (0, 1) \) and let \( \hat{f}(x, y) = 0 \) if \((x, y) \in \Omega \setminus \Omega \). Let \( \hat{g}(x, y) = 0 \), \((x, y) \in \Gamma_r \). We partition \( \Omega_r \) by uniform rectangular meshes, i.e., \( N_x = N_y \). We solve the resulting linear system (11) by Gauss elimination method(Gauss), the conjugate gradient squared method(CGS), the fast conjugate gradient squared method(FCGS), and the preconditioned fast conjugate gradient squared method(PFCGS), respectively. We present the \( L^2 \) errors of the numerical solutions in the original convex domain \( \Omega \), the corresponding convergence rates, the number of iterations in the corresponding conjugate gradient methods and the CPU time consumed by the above solvers for (11) in the Tables 1-3. Furthermore, we use a linear regression to fit the convergence rate \( \alpha \) and the associated constant \( C_\alpha \) in the error estimate

\[ \| u - u_h \|_{L^2} \leq C_\alpha h^\alpha. \]  

(44)

From Tables 1-3, we can obtain the following results: (i) The \( L^2 \) errors of the numerical solutions in the original convex domain \( \Omega \) become small as \( s \) decreases. This phenomenon is also observed in the work [22]. (ii) FCGS and PFCGS need less memory than the traditional solvers like Gauss and CGS. We can observe from Tables 1-3 that when \( N \) is equal to or bigger than \( 2^9 \), the memory is not enough. However, FCGS and PFCGS do not have this trouble because of less memory requirement. (iii) From Tables 1-3, we can observe that FCGS and PFCGS need less CPU time than Gauss and CGS especially for bigger \( N \). For example, when \( s = -1/2 \) and \( N = 2^9 \), the CPU time needed by Gauss

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\( N_x = N_y \) & \( \| u_h - u \|_{L^2} \) & \# of it. & CPUs \[ \hline
\hline
\text{Gauss} & \( 2^4 \) & 2.3963e-3 & 17 & 0.19s \\
& \( 2^5 \) & 1.4973e-3 & 29 & 0.33s \\
& \( 2^6 \) & 8.4352e-3 & 46 & 2.05s \\
& \( 2^7 \) & 4.1263e-4 & 69 & 5.86s \\
& \( 2^8 \) & 1.8965e-4 & 95 & 34.75s \\
& \( 2^9 \) & 8.8314e-5 & 133 & 335.19s \\
\hline
\text{CGS} & \( 2^4 \) & 2.3963e-3 & 17 & 0.19s \\
& \( 2^5 \) & 1.4973e-3 & 29 & 0.33s \\
& \( 2^6 \) & 8.4352e-3 & 46 & 2.05s \\
& \( 2^7 \) & 4.1263e-4 & 69 & 5.86s \\
& \( 2^8 \) & 1.8965e-4 & 95 & 34.75s \\
& \( 2^9 \) & 8.8314e-5 & 133 & 335.19s \\
\hline
\text{FCGS} & \( 2^4 \) & 2.3963e-3 & 17 & 0.19s \\
& \( 2^5 \) & 1.4973e-3 & 29 & 0.33s \\
& \( 2^6 \) & 8.4352e-3 & 46 & 2.05s \\
& \( 2^7 \) & 4.1263e-4 & 69 & 5.86s \\
& \( 2^8 \) & 1.8965e-4 & 95 & 34.75s \\
& \( 2^9 \) & 8.8314e-5 & 133 & 335.19s \\
\hline
\text{PFCGS} & \( 2^4 \) & 2.3963e-3 & 8 & 0.06s \\
& \( 2^5 \) & 1.4973e-3 & 10 & 0.08s \\
& \( 2^6 \) & 8.4352e-3 & 16 & 0.9s \\
& \( 2^7 \) & 4.1263e-4 & 18 & 2.25s \\
& \( 2^8 \) & 1.8965e-4 & 23 & 11.41s \\
& \( 2^9 \) & 8.8314e-5 & 28 & 112.81s \\
\hline
\end{tabular}
\caption{Performance of Gauss, CGS, FCGS, and PFCGS while \( s = 3/8 \).}
\end{table}
**TABLE 2.** Performance of Gauss, CGS, FCGS, and PFCGS while \( s = 0 \).

| \( N_x = N_y \) | \( \| u_h - u \|_{L^2} \) | # of iter. | CPUs |
|------------------|-----------------|----------|------|
| \( 2^4 \)        | 1.2517e-3       | 0.23s    |      |
| \( 2^5 \)        | 4.5916e-4       | 6.7s     |      |
| \( 2^6 \)        | 1.4481e-4       | 747.92s  |      |
| \( 2^7 \)        | out of memory    |          |      |
| \( 2^8 \)        | out of memory    |          |      |
| \( 2^9 \)        | \( C_\alpha = 0.10 \) |          |      |
| \( \alpha = 1.56 \) |                 |          |      |

**TABLE 3.** Performance of Gauss, CG, and FCG while \( s = -1/2 \).

| \( N_x = N_y \) | \( \| u_h - u \|_{L^2} \) | # of iter. | CPUs |
|------------------|-----------------|----------|------|
| \( 2^4 \)        | 7.5802e-4       |          |      |
| \( 2^5 \)        | 2.1117e-4       |          |      |
| \( 2^6 \)        | 5.3652e-5       |          |      |
| \( 2^7 \)        | out of memory    |          |      |
| \( 2^8 \)        | out of memory    |          |      |
| \( 2^9 \)        | \( C_\alpha = 0.15 \) |          |      |
| \( \alpha = 1.91 \) |                 |          |      |

**TABLE 4.** Performance of PFCGS with different penalization parameter.

| \( \eta \) | \( \| u_h - u \|_{L^2} \) |
|-----------|-----------------|
| \( 10^{-5} \) | 7.5803e-4       |
| \( 10^{-3} \) | 2.1117e-4       |
| \( 10^{-1} \) | 5.3652e-5       |
| \( 1 \)       | 1.3095e-5       |

and CGS exceed 10 hours. In the same case, FCGS and PFCGS need only dozens of seconds. (iv) The preconditioner introduced in the previous section is efficient to reduce the number of iterations and CPU time especially for bigger \( s \) and \( N \). For example, when \( s = 3/8 \) and \( N = 2^9 \), the run of FCGS requires 133 iterations and 335.19s while PFCGS requires only 28 iterations and 112.81s.

To further clearly illustrate the time reduction for the fast collocation method, we present the comparison of the CPU time consumed by Gauss, CGS, FCGS, and PFCGS in the Figure 2 (the top one). Moreover, to illustrate the effectiveness of the preconditioner developed in this article, we also show the the CPU time consumed by FCGS and PFCGS in the Figure 2 (the bottom one). We can observe from Figure 1 that the CPU time consumed by the fast solvers FCGS and PFCGS is significantly reduced, and the PFCGS solver is more effective than FCGS solver. To show the effectiveness of the preconditioner introduced in previous section, we present the distributions of eigenvalues before and after precondition in Figure 3. From Figure 3, we can observe the eigenvalues vary dramatically before precondition, but they fluctuate around 1 after introducing the preconditioner.

**Example 2:** In this numerical experiment, we run the example with \( s = -1/2 \) considered in Example 1 and we investigate the influence of the penalization parameter \( \eta \) introduced in (2). The value penalization parameter \( \eta \) is chosen to be \( 10^{-5}, 10^{-3}, 10^{-1}, \) and 1, respectively. We only use PFCGS to run this numerical example. The \( L^2 \) errors of the numerical results are shown in Table 4.

From Table 4, we can observe that the \( L^2 \) errors of the numerical results become small while \( \eta \) reduces. Moreover, the convergence rate of the errors improves when \( \eta \) reduces.

**Example 3:** In this numerical experiment, we consider \( \Omega \) is a right triangle computational domain. The three vertices of this domain are \((1/4, 1/4), (1/2, 1/4)\), and \((1/4, 3/4)\). We extend \( \Omega \) to a rectangular domain \( \Omega_r = (0, 1) \times (0, 1) \) and let \( f(x, y) = 0 \) if \((x, y) \in \Omega_r \setminus \Omega \). Let \( g(x, y) = 0 \) if \((x, y) \in \Gamma_r \). We run the example with \( s = -1/2 \) and \( \eta = 10^{-5} \). The true solution and the right hand side are similar as Example 1.
The numerical results of Example 3.

FIGURE 3. The distributions of eigenvalues before and after precondition.

| $N_x$ | $N_y$ | $\|u_h - u\|_{L^2}$ | # of iter. | CPUs |
|-------|-------|----------------------|------------|------|
| 2     | 2     | $3.3212e - 4$        | 17         | 0.03s|
| 2     | 3     | $3.7295e - 5$        | 17         | 0.09s|
| 2     | 4     | $1.6553e - 5$        | 17         | 1s   |
| 2     | 5     | $3.8147e - 6$        | 19         | 2.8s |
| 2     | 6     | $7.4542e - 7$        | 19         | 11.7s|
| 2     | 7     | $2.0854e - 7$        | 19         | 107.4s|

The $L^2$ errors from this example are shown in Table 5. From Table 5, we can observe that our fast collocation method can also be applied to right triangle domain (not necessarily symmetric).

VI. CONCLUSION AND FUTURE WORK

The fast methods for nonlocal models developed before require the physical domain where we numerically solve the governing equations is rectangular. To relax this restriction, in this article, we develop a fast collocation method for a static linear nonlocal diffusion model in convex domains via a volume-penalization approach. The main contributions of this article are listed as follows: (i) By introducing a penalization term to the original nonlocal diffusion model in convex domain, we extend the original model to a new problem on a rectangle which contains the physical convex domain. Then we develop a fast collocation method based on the extended problem by carefully analysing the structure of the coefficient matrix. (ii) We introduce an efficient preconditioner by carefully analysing the Toeplitz-like coefficient matrix. This method can be used when the physical domain is convex. Compared to the traditional collocation method, our method can significantly reduce the simulation time and computer storage when numerically solve the linear nonlocal diffusion model. However, because the coefficient matrix obtained from collocation discretization is no longer a BTTB matrix, this method does not apply to the nonlocal diffusion model with variable coefficients. We will investigate this problem in the future.

REFERENCES

[1] Q. Du, R. B. Lehoucq, and A. M. Tartakovsky, “Integral approximations to classical diffusion and smoothed particle hydrodynamics,” Comput. Methods Appl. Mech. Eng., vol. 286, pp. 216–229, Apr. 2015.
[2] P. W. Bates and A. Chmaj, “An integrodifferential model for phase transitions: Stationary solutions in higher space dimensions,” J. Stat. Phys., vol. 95, pp. 1119–1139, 1999.
[3] F. Bobaru and M. Duangpanya, “The peridynamic formulation for transient heat conduction,” Int. J. Heat Mass Transf., vol. 53, nos. 19–20, pp. 4047–4059, Sep. 2010.
[4] S. A. Silling, “Reformulation of elasticity theory for discontinuities and long-range forces,” J. Mech. Phys. Solids, vol. 48, no. 1, pp. 175–209, Jan. 2000.
[5] C. Y. Kao, Y. Lou, and W. Shen, “Random dispersal vs. non-local dispersal,” Discrete Contin. Dyn. Syst., vol. 26, pp. 551–596, 2010.
[6] G. Gilboa and S. Osher, “Nonlocal operators with applications to image processing,” Multiscale Model. Simul., vol. 7, no. 3, pp. 1005–1028, Jan. 2009.
[7] S. K. Panda, E. Karapinar, and A. Atangana, “A numerical schemes and comparisons for fixed point results with applications to the solutions of volterra integral equations in dislocatedextendeddiscrete-time metricspace,” Alexandria Eng. J., vol. 59, no. 2, pp. 815–827, Apr. 2020.
[8] S. K. Panda, T. Abdeljawad, and K. Kamara Swamy, “New numerical scheme for solving integral equations via fixed point method using distinct (o-F)-contractions,” Alexandria Eng. J., vol. 59, no. 4, pp. 2015–2026, Aug. 2020.
[9] T. Abdeljawad, R. P. Agarwal, E. Karapinar, and P. S. Kumari, “Solutions of the nonlinear integral equation and fractional differential equation using the technique of a fixed point with a numerical experiment in extended b-Metric space,” Symmetry, vol. 11, no. 5, p. 686, May 2019.
[10] E. Karapinar, P. Kumari, and D. Lateef, “A new approach to the solution of the fredholm integral equation via a fixed point on extended b-Metric spaces,” Symmetry, vol. 10, no. 10, p. 512, Oct. 2018.
[11] S. K. Panda, T. Abdeljawad, and C. Ravichandran, “A complex valued approach to the solutions of Riemann-Liouville integral and non-linear telegraph equation via fixed point method,” Chaos, Solitons Fractals, vol. 130, Jan. 2020, Art. no. 109439.
[12] M. Taylor and D. J. Steigmann, “A two-dimensional peridynamic model for thin plates,” Math. Mech. Solids, vol. 20, no. 8, pp. 998–1010, Sep. 2015.
[13] E. Oterkus and E. Madenci, “Peridynamic analysis of fiber-reinforced composite materials,” J. Mech. Mater. Struct., vol. 7, no. 1, pp. 45–84, Mar. 2012.
[14] E. Askari, F. Bobaru, R. B. Lehoucq, M. L. Parks, S. A. Silling, and O. Weckner, “Peridynamics for multiscale materials modeling,” J. Phys., Conf. Ser., vol. 125, Jul. 2008, Art. no. 012078.
[15] K. Volya and K. Bhatnacharya, “Kinetics of phase transformations in the peridynamic formulation of continuum mechanics,” J. Mech. Phys. Solids, vol. 54, no. 9, pp. 1811–1842, Sep. 2006.
[16] S. A. Silling, O. Weckner, E. Askari, and F. Bobaru, “Crack nucleation in a peridynamic solid,” Int. J. Fract., vol. 162, nos. 1–2, pp. 219–227, Mar. 2010.
[17] F. Bobaru, S. A. Silling, and H. Jiang, “Peridynamic fracture and damage modeling of membranes and nanofiber networks,” in Proc. 11th Int. Conf. Fract., vol. 2, 2005, pp. 976–981.
[18] S. A. Silling and E. Askari, “Peridynamic modeling of impact damage,” Amer. Soc. Mech. Engineers, Pressure Vessels Piping Division, vol. 489, pp. 197–205, Oct. 2004.
[19] H. Wang and H. Tian, “A fast Galerkin method with efficient matrix assembly and storage for a peridynamic model,” J. Comput. Phys., vol. 231, no. 23, pp. 7730–7738, Dec. 2012.
[20] H. Liu, A. Cheng, and H. Wang, “A fast discontinuous Galerkin method for a bond-based linear peridynamic model discretized on a locally refined composite mesh,” J. Scientific Comput., vol. 76, no. 2, pp. 913–942, Aug. 2018.
[21] H. Tian, H. Wang, and W. Wang, “An efficient collocation method for a non-local diffusion model,” Int. J. Numer. Anal. Mod., vol. 10, pp. 815–825, 2013.
[22] H. Wang and H. Tian, “A fast and faithful collocation method with efficient matrix assembly for a two-dimensional nonlocal diffusion model,” Comput. Methods Appl. Mech. Eng., vol. 273, pp. 19–36, May 2014.
[23] Z. Liu and X. Li, “A fast finite difference method for a continuous static linear bond-based peridynamics model of mechanics,” J. Sci. Comput., vol. 74, no. 2, pp. 728–742, Feb. 2018.
[24] Q. Du and J. Yang, “Fast and accurate implementation of Fourier spectral approximations of nonlocal diffusion operators and its applications,” J. Comput. Phys., vol. 332, pp. 118–134, Mar. 2017.
[25] J. Jia and H. Wang, “A fast finite volume method for conservative space-fractional diffusion equations in convex domains,” J. Comput. Phys., vol. 310, pp. 63–84, Apr. 2016.

[26] N. Du, H.-W. Sun, and H. Wang, “A preconditioned fast finite difference scheme for space-fractional diffusion equations in convex domains,” Comput. Appl. Math., vol. 38, no. 1, p. 24, Mar. 2019.

[27] P. Angot, C. H. Bruneau, and P. Fabrie, “A penalization method to take into account obstacles in incompressible viscous flows,” Numer. Math., vol. 81, pp. 491–520, May 1999.

[28] M. K. Ng, Iterative Methods for Toeplitz Systems. London, U.K.: Oxford Univ. Press, 2004.

[29] X. Q. Jin, Preconditioning Techniques for Toeplitz Systems, 1st ed. Beijing, China: Higher Education Press, 2010.

[30] N. Du and H. Wang, “A fast finite element method for space-fractional dispersion equations on bounded domains in $\mathbb{R}^2$,” SIAM J. Sci. Comput., vol. 37, no. 3, pp. 1614–1635, Jan. 2015.

XUHAO ZHANG received the Ph.D. degree in computational mathematics from Shandong University, China, in 2016. Since 2017, he has been working as a Lecturer with the School of Mathematics and Statistics, Qilu University of Technology (Shandong Academy of Sciences), China. His research interests include fast numerical methods for nonlocal models (fractional partial differential equations and peridynamic models) and software development of oil reservoir simulation.

AIJIE CHENG received the Ph.D. degree in computational mathematics from Shandong University, China. Since 2001, he has been working as a Professor of Mathematics with the School of Mathematics, Shandong University. His research interests include numerical solutions of partial differential equations (especially for equations governing porous media flow) and software development of oil reservoir simulation.

HONG WANG received the Ph.D. degree in mathematics from the University of Wyoming, USA. Since 2003, he has been working as a Professor of Mathematics with the Department of Mathematics, University of South Carolina, USA. His research interests include numerical approximation to differential/ integral equations and scientific computations.

* * *