Commun. Comput. Phys.
doi: 10.4208/cicp.120314.230115a

Vol. **18**, No. 2, pp. 469-488
August 2015

Preconditioned Iterative Methods for Two-Dimensional Space-Fractional Diffusion Equations

Xiao-Qing Jin[1], Fu-Rong Lin[2] and Zhi Zhao[1,*]

[1] *Department of Mathematics, University of Macau, Macao 999078, China.*
[2] *Department of Mathematics, Shantou University, Shantou 515063, China.*

Received 12 March 2014; Accepted (in revised version) 23 January 2015

Abstract. In this paper, preconditioned iterative methods for solving two-dimensional space-fractional diffusion equations are considered. The fractional diffusion equation is discretized by a second-order finite difference scheme, namely, the Crank-Nicolson weighted and shifted Grünwald difference (CN-WSGD) scheme proposed in [W. Tian, H. Zhou and W. Deng, *A class of second order difference approximation for solving space fractional diffusion equations*, Math. Comp., 84 (2015) 1703-1727]. For the discretized linear systems, we first propose preconditioned iterative methods to solve them. Then we apply the D'Yakonov ADI scheme to split the linear systems and solve the obtained splitting systems by iterative methods. Two preconditioned iterative methods, the preconditioned generalized minimal residual (preconditioned GMRES) method and the preconditioned conjugate gradient normal residual (preconditioned CGNR) method, are proposed to solve relevant linear systems. By fully exploiting the structure of the coefficient matrix, we design two special kinds of preconditioners, which are easily constructed and are able to accelerate convergence of iterative solvers. Numerical results show the efficiency of our preconditioners.

AMS subject classifications: 65F10, 65L06, 65U05

Key words: Fractional diffusion equation, CN-WSGD scheme, preconditioned GMRES method, preconditioned CGNR method, Toeplitz matrix, fast Fourier transform.

1 Introduction

In this paper, we consider the following initial boundary value problem of two-dimensional space-fractional diffusion equation [25, 43]:

*Corresponding author. *Email addresses:* xqjin@umac.mo (X.-Q. Jin), frlin@stu.edu.cn (F.-R. Lin), zhaozhi231@163.com (Z. Zhao)

 ©2015 Global-Science Press

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

$$\begin{cases} \dfrac{\partial u(x,y,t)}{\partial t} = d_+(x,y,t)\,{}_aD_x^\alpha u(x,y,t) + d_-(x,y,t)\,{}_xD_b^\alpha u(x,y,t) \\ \qquad\qquad + e_+(x,y,t)\,{}_cD_y^\beta u(x,y,t) + e_-(x,y,t)\,{}_yD_d^\beta u(x,y,t) + f(x,y,t), \\ \qquad\qquad\qquad\qquad (x,y,t)\in\Omega\times(0,T], \\ u(x,y,0) = u_0(x,y), \qquad (x,y)\in\overline{\Omega}, \\ u(x,y,t) = 0, \qquad (x,y,t)\in\partial\Omega\times[0,T], \end{cases} \tag{1.1}$$

where $1<\alpha,\beta<2$, $d_\pm(x,y,t), e_\pm(x,y,t)\geq 0$, and $\Omega=(a,b)\times(c,d)$. In addition, ${}_aD_x^\alpha$, ${}_xD_b^\alpha$, ${}_cD_y^\beta$, and ${}_yD_d^\beta$ are the left-sided and right-sided Riemann-Liouville fractional derivatives defined by

$$\begin{cases} {}_aD_x^\alpha u(x,y,t) = \dfrac{1}{\Gamma(2-\alpha)}\dfrac{\mathrm{d}^2}{\mathrm{d}x^2}\displaystyle\int_a^x \frac{u(\xi,y,t)}{(x-\xi)^{\alpha-1}}\mathrm{d}\xi, \\ {}_xD_b^\alpha u(x,y,t) = \dfrac{1}{\Gamma(2-\alpha)}\dfrac{\mathrm{d}^2}{\mathrm{d}x^2}\displaystyle\int_x^b \frac{u(\xi,y,t)}{(\xi-x)^{\alpha-1}}\mathrm{d}\xi, \\ {}_cD_y^\beta u(x,y,t) = \dfrac{1}{\Gamma(2-\beta)}\dfrac{\mathrm{d}^2}{\mathrm{d}y^2}\displaystyle\int_c^y \frac{u(x,\eta,t)}{(y-\eta)^{\beta-1}}\mathrm{d}\eta, \\ {}_yD_d^\beta u(x,y,t) = \dfrac{1}{\Gamma(2-\beta)}\dfrac{\mathrm{d}^2}{\mathrm{d}y^2}\displaystyle\int_y^d \frac{u(x,\eta,t)}{(\eta-y)^{\beta-1}}\mathrm{d}\eta. \end{cases}$$

Fractional diffusion equations (FDEs) have many applications in various fields, such as fluid mechanics, finance, biology, turbulent flow, electrochemistry physics, and image processing [2,8,18,22,28,31,33,35,36]. Since closed-form solutions are only available for very few FDEs, a variety of numerical methods for FDEs have been developed in the last decade [12–16,21,24–27,32,37,38,41,44].

In 2004 and 2006, Meerschaert and Tadjeran proposed a shifted Grünwald discretization scheme to approximate the FDE with a left-sided fractional derivative and the FDE with two-sided fractional derivatives, respectively [24,25]. In 2006, Tadjeran, Meerschaert, and Scheffler proposed a second-order accurate numerical scheme based on the classical Crank-Nicolson method and the Richardson extrapolation scheme [38]. Their methods were proved to be unconditionally stable. In 2012, Tian, Zhou, and Deng proposed a weighted and shifted Grünwald difference (WSGD) scheme to approximate left-sided and right-sided Riemann-Liouville fractional derivatives, and obtained a class of second-order accurate CN-WSGD schemes [39].

Although the discretization of second-order partial differential equations leads to linear systems with sparse coefficient matrices, the matrices corresponding to FDEs are full. Fortunately, the coefficient matrices of the above mentioned methods possess a special Toeplitz-like structure: it can be written as a sum of diagonal-multiply-Toeplitz matrices [39, 41]. Thus the storage requirement is significantly reduced from $\mathcal{O}(N^2)$ to

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

$\mathcal{O}(N)$, and the complexity of the matrix-vector multiplication is reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log N)$ by using fast algorithms based on the fast Fourier transform (FFT) [10,11,17], where "N" is the number of unknowns at each time step.

Recently, several fast iterative solvers have been developed for solving space-fractional diffusion equations [4, 19, 20, 42, 43]. Wang and Wang proposed the conjugate gradient normal residual (CGNR) method to solve the discretized systems [42]. Numerical results show that the CGNR method converges fast when the diffusion coefficients are very small. However, if the diffusion coefficients are not sufficiently small, the discretized systems may become very ill-conditioned when the grid size is small, and the CGNR method converges slowly; see for instance [19,20]. Pang and Sun proposed an efficient multigrid method to solve the discretized systems [29]. Wang and Basu proposed a fast conjugate gradient squared (CGS) method for two-dimensional space-fractional diffusion equations [43]. Wang and Wang proposed an alternating-direction finite difference method for two-dimensional space-fractional diffusion equations by fully decoupling the linear systems in x-direction and y-direction [40].

Preconditioning techniques have been introduced to accelerate the convergence of iterative methods [19, 20]. Lei and Sun proposed a circulant preconditioner in [19], and proved that the preconditioned CG type methods converge superlinearly when the diffusion coefficients are constant. Lin, Yang, and Jin developed banded preconditioners for FDEs by exploiting the off-diagonal decay property of the coefficient matrix [20]. The above-mentioned preconditioned iterative methods require only $\mathcal{O}(N\log N)$ operations at each time step by using the Toeplitz-like structure of the coefficient matrix. To our knowledge, there have been no preconditioned iterative methods proposed for solving two-dimensional space-fractional diffusion equations up to now.

In this paper, the CN-WSGD scheme is used to discretize two-dimensional space-fractional diffusion equations. Two preconditioned iterative methods are proposed to solve the discretized problem by extending a banded preconditioner proposed in [20]. Firstly, a preconditioner defined by the LU factorization of a block-banded-banded-block approximation of the coefficient matrix is introduced, and the linear systems are solved by a preconditioned GMRES method and a preconditioned CGNR method. Then we consider the D'Yakonov ADI scheme with a banded preconditioner for the case where the diffusion coefficients have special structures. The cost at each time step requires $\mathcal{O}(N^2\log N)$ operations only, where "$N=N_1=N_2$" is the product of the numbers of spatial partitions in x-direction and y-direction (see Section 2).

The rest of this paper is organized as follows. In Section 2, we discretize the FDE (1.1) by the CN-WSGD scheme, and give the matrix form of this finite difference method. Then we apply the D'Yakonov ADI scheme to split the discretized linear systems, and obtain the corresponding matrix form of the scheme. In Section 3, the preconditioned GMRES method and the preconditioned CGNR method are proposed to solve the discretized linear systems of (1.1). Finally, in Section 4, we present numerical results to show the efficiency of our methods.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

2 The CN-WSGD scheme

In this section, we use the CN-WSGD scheme to discrete the FDE (1.1). The time interval $[0,T]$ is partitioned into M subintervals of length $\tau:=T/M$, and the spatial domain Ω is partitioned uniformly with grid sizes $h_x:=(b-a)/(N_1+1)$ and $h_y:=(d-c)/(N_2+1)$, where M, N_1, and N_2 are positive integers. Let $x_i:=a+ih_x$ for $i=0,1,\cdots,N_1+1$, $y_j:=c+jh_y$ for $j=0,1,\cdots,N_2+1$, and $t^{(m)}:=m\tau$ for $m=0,1,\cdots,M$. Let $t^{(m+1/2)}:=(t^{(m)}+t^{(m+1)})/2$ for $m=1,2,\cdots,M-1$.

In the following, we consider numerical methods for the solution of the discretized linear systems, which are obtained by using the second-order accurate CN-WSGD scheme with shifting parameters $(p,q)=(1,0)$ given by [39]:

$$\begin{cases} {}_L\mathcal{D}^{\alpha}_{h_x}u(x_i,y_j,t^{(m)})=\dfrac{1}{h_x^{\alpha}}\displaystyle\sum_{k=0}^{i}w_k^{(\alpha)}u(x_{i-k+1},y_j,t^{(m)}),\\ {}_R\mathcal{D}^{\alpha}_{h_x}u(x_i,y_j,t^{(m)})=\dfrac{1}{h_x^{\alpha}}\displaystyle\sum_{k=0}^{N_1-i+1}w_k^{(\alpha)}u(x_{i+k-1},y_j,t^{(m)}),\\ {}_L\mathcal{D}^{\beta}_{h_y}u(x_i,y_j,t^{(m)})=\dfrac{1}{h_y^{\beta}}\displaystyle\sum_{k=0}^{j}w_k^{(\beta)}u(x_i,y_{j-k+1},t^{(m)}),\\ {}_R\mathcal{D}^{\beta}_{h_y}u(x_i,y_j,t^{(m)})=\dfrac{1}{h_y^{\beta}}\displaystyle\sum_{k=0}^{N_2-j+1}w_k^{(\beta)}u(x_i,y_{j+k-1},t^{(m)}), \end{cases}$$

with $u_0=u_{N+1}=0$. Here the coefficients $w_k^{(\gamma)}$ are given by

$$w_0^{(\gamma)}=\frac{\gamma}{2}g_0^{(\gamma)},\quad w_k^{(\gamma)}=\frac{\gamma}{2}g_k^{(\gamma)}+\frac{2-\gamma}{2}g_{k-1}^{(\gamma)},\quad k\geq 1,$$

where $g_k^{(\gamma)}=(-1)^k\binom{\gamma}{k}$ with $\binom{\gamma}{k}$ being the fractional binomial coefficients. For $\gamma=\alpha,\beta$, the fractional binomial coefficients $g_j^{(\gamma)}$ have the following properties [24,25,29]:

$$\begin{cases} g_0^{(\gamma)}=1,\quad g_1^{(\gamma)}=-\gamma,\quad g_2^{(\gamma)}>g_3^{(\gamma)}>\cdots>0,\\ \displaystyle\sum_{j=0}^{\infty}g_j^{(\gamma)}=0,\quad \sum_{j=0}^{n}g_j^{(\gamma)}<0,\quad n\geq 1,\\ g_j^{(\gamma)}=\mathcal{O}\big(j^{-(\gamma+1)}\big). \end{cases}\tag{2.1}$$

Similarly, the coefficients $w_k^{(\gamma)}$ have the following basic properties.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

Proposition 2.1 ([39]). The coefficients $w_k^{(\gamma)}$ satisfy the following properties for $1<\gamma<2$,

$$\begin{cases} w_0^{(\gamma)}=\dfrac{\gamma}{2}, \quad w_1^{(\gamma)}=\dfrac{2-\gamma-\gamma^2}{2}<0, \quad w_2^{(\gamma)}=\dfrac{\gamma(\gamma^2+\gamma-4)}{4}, \\ 1\geq w_0^{(\gamma)}\geq w_3^{(\gamma)}\geq w_4^{(\gamma)}\geq\cdots\geq 0, \\ \displaystyle\sum_{j=0}^{\infty} w_j^{(\gamma)}=0, \quad \sum_{j=0}^{n} w_j^{(\gamma)}<0, \quad n\geq 2. \end{cases}$$

Moreover, we have

$$w_j^{(\gamma)}=\mathcal{O}\big(j^{-(\gamma+1)}\big).$$

Let

$$\begin{cases} \delta_x^{\alpha,(m)}=\dfrac{d_{i,j}^{+,(m)}\tau}{2}{}_L\mathcal{D}_{h_x}^{\alpha}+\dfrac{d_{i,j}^{-,(m)}\tau}{2}{}_R\mathcal{D}_{h_x}^{\alpha}, \\ \delta_y^{\beta,(m)}=\dfrac{e_{i,j}^{+,(m)}\tau}{2}{}_L\mathcal{D}_{h_y}^{\beta}+\dfrac{e_{i,j}^{-,(m)}\tau}{2}{}_R\mathcal{D}_{h_y}^{\beta}, \end{cases}$$

where

$$\begin{aligned} d_{i,j}^{+,(m)}&=d_+(x_i,y_j,t^{(m)}), \qquad d_{i,j}^{-,(m)}=d_-(x_i,y_j,t^{(m)}), \\ e_{i,j}^{+,(m)}&=e_+(x_i,y_j,t^{(m)}), \qquad e_{i,j}^{-,(m)}=e_-(x_i,y_j,t^{(m)}). \end{aligned}$$

The two-dimensional CN-WSGD finite difference method can be formulated as follows:

$$\big(1-\delta_x^{\alpha,(m+1)}-\delta_y^{\beta,(m+1)}\big)u_{i,j}^{(m+1)}=\big(1+\delta_x^{\alpha,(m)}+\delta_y^{\beta,(m)}\big)u_{i,j}^{(m)}+\tau f_{i,j}^{(m+1/2)}, \tag{2.2}$$

where $u_{i,j}^{(m)}=u(x_i,y_j,t^{(m)})$ and $f_{i,j}^{(m+1/2)}=f(x_i,y_j,t^{(m+1/2)})$.

We now introduce the matrix form of the above equation. First we need to define the following notations.

1. Let $N=N_1=N_2$. Define N^2-dimensional vectors $\mathbf{u}^{(m)}$, $\mathbf{d}_+^{(m)}$, $\mathbf{d}_-^{(m)}$, $\mathbf{e}_+^{(m)}$, $\mathbf{e}_-^{(m)}$, and $\mathbf{f}^{(m+1/2)}$ as follows:

$$\begin{cases} \mathbf{u}^{(m)}=\big[u_{1,1}^{(m)},\cdots,u_{N_1,1}^{(m)},u_{1,2}^{(m)},\cdots,u_{N_1,2}^{(m)},\cdots,u_{1,N_2}^{(m)},\cdots,u_{N_1,N_2}^{(m)}\big]^T, \\ \mathbf{d}_+^{(m)}=\big[d_{1,1}^{+,(m)},\cdots,d_{N_1,1}^{+,(m)},d_{1,2}^{+,(m)},\cdots,d_{N_1,2}^{+,(m)},\cdots,d_{1,N_2}^{+,(m)},\cdots,d_{N_1,N_2}^{+,(m)}\big]^T, \\ \mathbf{d}_-^{(m)}=\big[d_{1,1}^{-,(m)},\cdots,d_{N_1,1}^{-,(m)},d_{1,2}^{-,(m)},\cdots,d_{N_1,2}^{-,(m)},\cdots,d_{1,N_2}^{-,(m)},\cdots,d_{N_1,N_2}^{-,(m)}\big]^T, \\ \mathbf{e}_+^{(m)}=\big[e_{1,1}^{+,(m)},\cdots,e_{N_1,1}^{+,(m)},e_{1,2}^{+,(m)},\cdots,e_{N_1,2}^{+,(m)},\cdots,e_{1,N_2}^{+,(m)},\cdots,e_{N_1,N_2}^{+,(m)}\big]^T, \\ \mathbf{e}_-^{(m)}=\big[e_{1,1}^{-,(m)},\cdots,e_{N_1,1}^{-,(m)},e_{1,2}^{-,(m)},\cdots,e_{N_1,2}^{-,(m)},\cdots,e_{1,N_2}^{-,(m)},\cdots,e_{N_1,N_2}^{-,(m)}\big]^T, \\ \mathbf{f}^{(m+1/2)}=\big[f_{1,1}^{(m+1/2)},\cdots,f_{N_1,1}^{(m+1/2)},f_{1,2}^{(m+1/2)},\cdots,f_{N_1,2}^{(m+1/2)},\cdots,f_{1,N_2}^{(m+1/2)},\cdots,f_{N_1,N_2}^{(m+1/2)}\big]^T. \end{cases}$$

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

2. Define the following $N^2\times N^2$ diagonal matrices:

$$D_+^{(m+1)}=\mathrm{diag}\big(r\mathbf{d}_+^{(m+1)}\big),\qquad D_-^{(m+1)}=\mathrm{diag}\big(r\mathbf{d}_-^{(m+1)}\big),$$
$$E_+^{(m+1)}=\mathrm{diag}\big(s\mathbf{e}_+^{(m+1)}\big),\qquad E_-^{(m+1)}=\mathrm{diag}\big(s\mathbf{e}_-^{(m+1)}\big),$$

where $r=\frac{\tau}{2h_x^\alpha}$ and $s=\frac{\tau}{2h_y^\beta}$.

3. Define the Toeplitz matrices $A_Q^{\gamma,L}$ and $A_Q^{\gamma,R}$ by

$$A_Q^{\gamma,L}=\begin{bmatrix} w_1^{(\gamma)} & w_0^{(\gamma)} & 0 & \cdots & 0 & 0\\ w_2^{(\gamma)} & w_1^{(\gamma)} & w_0^{(\gamma)} & \ddots & \ddots & 0\\ \vdots & w_2^{(\gamma)} & w_1^{(\gamma)} & \ddots & \ddots & \vdots\\ \vdots & \ddots & \ddots & \ddots & \ddots & 0\\ w_{Q-1}^{(\gamma)} & \ddots & \ddots & \ddots & w_1^{(\gamma)} & w_0^{(\gamma)}\\ w_Q^{(\gamma)} & w_{Q-1}^{(\gamma)} & \cdots & \cdots & w_2^{(\gamma)} & w_1^{(\gamma)} \end{bmatrix},\qquad A_Q^{\gamma,R}=\big(A_Q^{\gamma,L}\big)^T.$$

4. Define the $N^2\times N^2$ matrices $A_x^{(m+1)}$ and $A_y^{(m+1)}$ by

$$\begin{cases} A_x^{(m+1)}=D_+^{(m+1)}\big(I_{N_2}\otimes A_{N_1}^{\alpha,L}\big)+D_-^{(m+1)}\big(I_{N_2}\otimes A_{N_1}^{\alpha,R}\big),\\ A_y^{(m+1)}=E_+^{(m+1)}\big(A_{N_2}^{\beta,L}\otimes I_{N_1}\big)+E_-^{(m+1)}\big(A_{N_2}^{\beta,R}\otimes I_{N_1}\big). \end{cases}$$

5. Define $A^{(m+1)}=A_x^{(m+1)}+A_y^{(m+1)}$.

Then the CN-WSGD finite difference method (2.2) can be written in the following matrix form

$$\big(I_{N^2}-A^{(m+1)}\big)\mathbf{u}^{(m+1)}=\big(I_{N^2}+A^{(m)}\big)\mathbf{u}^{(m)}+\tau\mathbf{f}^{(m+1/2)},$$

where I_Q denotes the $Q\times Q$ identity matrix. Let

$$\underline{A}^{(m+1)}=I_{N^2}-A^{(m+1)},\qquad \mathbf{b}^{(m+1)}=\big(I_{N^2}+A^{(m)}\big)\mathbf{u}^{(m)}+\tau\mathbf{f}^{(m+1/2)}.$$

The above linear system can be written as

$$\underline{A}^{(m+1)}\mathbf{u}^{(m+1)}=\mathbf{b}^{(m+1)}. \tag{2.3}$$

Let $\gamma_0=(-1+\sqrt{17})/2\approx 1.562$. It can be seen from Proposition 2.1 that if $\gamma\in[\gamma_0,2)$, then $w_2^{(\gamma)}\geq 0$, and then the coefficient matrix $\underline{A}^{(m+1)}$ is a strictly diagonally dominant M-matrix. In addition, the spectral radius of the matrix $\big(I_{N^2}-A^{(m+1)}\big)^{-1}\big(I_{N^2}+A^{(m+1)}\big)$ is less than 1. Thus, when the diffusion coefficients are time independent, the CN-WSGD

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

scheme is unconditionally stable and the truncation error is $\mathcal{O}(h_x^2+h_y^2+\tau^2)$; see [39]. Compared with the method given in [38], the CN-WSGD scheme is second-order accurate without the Richardson extrapolation scheme.

To solve the two-dimensional space-fractional diffusion equations (1.1) numerically, we can use iterative methods to solve the discretized linear systems (2.2) directly. Actually, we can also apply the D'Yakonov ADI scheme to split the discretized linear systems (2.2) first, and then solve the obtained splitting linear systems by iterative methods. The linear systems splitted by the D'Yakonov ADI scheme are given by [39, 40]:

$$\begin{cases} \big(1-\delta_x^{\alpha,(m+1)}\big)v_{i,j}^{(m+1)} = \big(1+\delta_x^{\alpha,(m)}\big)\big(1+\delta_y^{\beta,(m)}\big)u_{i,j}^{(m)} + \tau f_{i,j}^{(m+1/2)}, \\ \big(1-\delta_y^{\beta,(m+1)}\big)u_{i,j}^{(m+1)} = v_{i,j}^{(m+1)}, \end{cases} \tag{2.4}$$

where $u_{i,j}^{(m)} = u(x_i,y_j,t^{(m)})$ and $f_{i,j}^{(m+1/2)} = f(x_i,y_j,t^{(m+1/2)})$.

We now introduce the matrix form of the above equation. First we need to define the following notations.

1. Define the N_1-dimensional vectors $\mathbf{u}_{\cdot,j}^{(m)}$, $\mathbf{v}_{\cdot,j}^{(m)}$, $\mathbf{d}_{+,j}^{(m)}$, $\mathbf{d}_{-,j}^{(m)}$, $\mathbf{f}_{\cdot,j}^{(m+1/2)}$ for $j=1,\cdots,N_2$, and the N_2-dimensional vectors $\mathbf{u}_{i,\cdot}^{(m)}$, $\mathbf{v}_{i,\cdot}^{(m)}$, $\mathbf{e}_{i,+}^{(m)}$, $\mathbf{e}_{i,-}^{(m)}$, $\mathbf{f}_{i,\cdot}^{(m+1/2)}$ for $i=1,\cdots,N_1$ as follows:

$$\begin{cases} \mathbf{u}_{i,\cdot}^{(m)} = \big[u_{i,1}^{(m)},\cdots,u_{i,N_2}^{(m)}\big]^T, & \mathbf{u}_{\cdot,j}^{(m)} = \big[u_{1,j}^{(m)},\cdots,u_{N_1,j}^{(m)}\big]^T, \\ \mathbf{v}_{i,\cdot}^{(m)} = \big[v_{i,1}^{(m)},\cdots,v_{i,N_2}^{(m)}\big]^T, & \mathbf{v}_{\cdot,j}^{(m)} = \big[v_{1,j}^{(m)},\cdots,v_{N_1,j}^{(m)}\big]^T, \\ \mathbf{d}_{+,j}^{(m)} = \big[d_{1,j}^{+,(m)},\cdots,d_{N_1,j}^{+,(m)}\big]^T, & \mathbf{d}_{-,j}^{(m)} = \big[d_{1,j}^{-,(m)},\cdots,d_{N_1,j}^{-,(m)}\big]^T, \\ \mathbf{e}_{i,+}^{(m)} = \big[e_{i,1}^{+,(m)},\cdots,e_{i,N_2}^{+,(m)}\big]^T, & \mathbf{e}_{i,-}^{(m)} = \big[e_{i,1}^{-,(m)},\cdots,e_{i,N_2}^{-,(m)}\big]^T, \\ \mathbf{f}_{i,\cdot}^{(m+1/2)} = \big[f_{i,1}^{(m+1/2)},\cdots,f_{i,N_2}^{(m+1/2)}\big]^T, & \mathbf{f}_{\cdot,j}^{(m+1/2)} = \big[f_{1,j}^{(m+1/2)},\cdots,f_{N_1,j}^{(m+1/2)}\big]^T. \end{cases}$$

2. Define the following $N_1\times N_1$ diagonal matrices $D_{+,j}^{(m+1)}$, $D_{-,j}^{(m+1)}$ for $j=1,\cdots,N_2$, and the $N_2\times N_2$ diagonal matrices $E_{i,+}^{(m+1)}$, and $E_{i,-}^{(m+1)}$ for $i=1,\cdots,N_1$:

$$\begin{aligned} D_{+,j}^{(m+1)} &= \mathrm{diag}\big(r\mathbf{d}_{+,j}^{(m+1)}\big), & D_{-,j}^{(m+1)} &= \mathrm{diag}\big(r\mathbf{d}_{-,j}^{(m+1)}\big), \\ E_{i,+}^{(m+1)} &= \mathrm{diag}\big(s\mathbf{e}_{i,+}^{(m+1)}\big), & E_{i,-}^{(m+1)} &= \mathrm{diag}\big(s\mathbf{e}_{i,-}^{(m+1)}\big). \end{aligned}$$

where $r=\frac{\tau}{2h_x^\alpha}$ and $s=\frac{\tau}{2h_y^\beta}$.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

3. Define the $N_1\times N_1$ matrices $A_{x,j}^{(m+1)}$ for $j=1,\cdots,N_2$, and the $N_2\times N_2$ matrices $A_{i,y}^{(m+1)}$ for $i=1,\cdots,N_1$ by

$$\begin{cases} A_{x,j}^{(m+1)}=D_{+,j}^{(m+1)}A_L^{\alpha,N_1}+D_{-,j}^{(m+1)}A_R^{\alpha,N_1}, \\ A_{i,y}^{(m+1)}=E_{i,+}^{(m+1)}A_L^{\beta,N_2}+E_{i,-}^{(m+1)}A_R^{\beta,N_2}. \end{cases}$$

4. Define the $N_1\times N_2$ matrix $G^{(m)}$ as follow

$$G^{(m)}=\left[\mathbf{g}_1^{(m)},\mathbf{g}_2^{(m)},\cdots,\mathbf{g}_{N_1}^{(m)}\right]^T,$$

where $\mathbf{g}_i^{(m)}=(I_{N_2}+A_{i,y}^{(m)})\mathbf{u}_{i,\cdot}^{(m)}$ for $i=1,\cdots,N_1$. Then denote the $N_2\times N_1$ matrix $H^{(m)}$ as follow

$$H^{(m)}=\left[\mathbf{h}_1^{(m)},\mathbf{h}_2^{(m)},\cdots,\mathbf{h}_{N_2}^{(m)}\right],$$

where $\mathbf{h}_j^{(m)}=(I_{N_1}+A_{x,j}^{(m)})G^{(m)}(:,j)$ for $j=1,\cdots,N_2$.

The D'Yakonov ADI scheme(2.4) can be written in the following matrix form

$$\begin{cases} (I_{N_1}-A_{x,j}^{(m+1)})\mathbf{v}_{\cdot,j}^{(m+1)}=\mathbf{h}_j^{(m)}+\tau\mathbf{f}_{\cdot,j}^{(m+1/2)}, \quad j=1,\cdots,N_2, \\ (I_{N_2}-A_{i,y}^{(m+1)})\mathbf{u}_{i,\cdot}^{(m+1)}=\mathbf{v}_{i,\cdot}^{(m+1)}, \quad i=1,\cdots,N_1. \end{cases}$$

Let

$$\underline{A}_{x,j}^{(m+1)}=I_{N_1}-A_{x,j}^{(m+1)}, \qquad \widetilde{\mathbf{b}}_j^{(m+1)}=\mathbf{h}_j^{(m)}+\tau\mathbf{f}_{\cdot,j}^{(m+1/2)}, \quad j=1,\cdots,N_2,$$
$$\underline{A}_{i,y}^{(m+1)}=I_{N_2}-A_{i,y}^{(m+1)}, \qquad \widehat{\mathbf{b}}_i^{(m+1)}=\mathbf{v}_{i,\cdot}^{(m+1)}, \quad i=1,\cdots,N_1.$$

Thus the above linear systems can be written as

$$\begin{cases} \underline{A}_{x,j}^{(m+1)}\mathbf{v}_{\cdot,j}^{(m+1)}=\widetilde{\mathbf{b}}_j^{(m+1)}, \qquad j=1,\cdots,N_2, \\ \underline{A}_{i,y}^{(m+1)}\mathbf{u}_{i,\cdot}^{(m+1)}=\widehat{\mathbf{b}}_i^{(m+1)}, \qquad i=1,\cdots,N_1. \end{cases} \tag{2.5}$$

It has been proved in [39] that if $\alpha,\beta\in(1,2)$ and all diffusion coefficients are constants, then the D'Yakonov ADI scheme is unconditionally stable and convergent, and the truncation error is $\mathcal{O}(h_x^2+h_y^2+\tau^2)$.

In the following, we discuss a special case where the diffusion coefficients are not constant: assume that $\alpha,\beta\in[\gamma_0,2)$ and the diffusion coefficients $d_\pm$ and $e_\pm$ depend only on x and y, respectively. That is,

$$d_+(x,y,t)=d_+(x), \qquad d_-(x,y,t)=d_-(x),$$
$$e_+(x,y,t)=e_+(y), \qquad e_-(x,y,t)=e_-(y).$$

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

Let

$$\begin{cases} \widehat{A}_x = D_+ A_{N_1}^{\alpha,L} + D_- A_{N_1}^{\alpha,R}, & j=1,2,\cdots,N_2, \\ \widehat{A}_y = E_+ A_{N_2}^{\beta,L} + E_- A_{N_2}^{\beta,R}, & i=1,2,\cdots,N_1. \end{cases}$$

The diagonal matrices $D_\pm$ and $E_\pm$ are defined by

$$D_+ = \mathrm{diag}\big(r[d^+(x_1),\cdots,d^+(x_{N_1})]\big), \qquad D_- = \mathrm{diag}\big(r[d^-(x_1),\cdots,d^-(x_{N_1})]\big),$$

$$E_+ = \mathrm{diag}\big(s[e^+(y_1),\cdots,e^+(y_{N_2})]\big), \qquad E_- = \mathrm{diag}\big(s[e^-(y_1),\cdots,e^-(y_{N_2})]\big),$$

where $r=\frac{\tau}{2h_x^\alpha}$ and $s=\frac{\tau}{2h_y^\beta}$. Then the D'Yakonov ADI scheme (2.4) can be rewritten as

$$\begin{cases} \big(I_{N_2}\otimes(I_{N_1}-\widehat{A}_x)\big)\mathbf{v}^{(m+1)} = \widetilde{\mathbf{b}}^{(m+1)}, \\ \big((I_{N_2}-\widehat{A}_y)\otimes I_{N_1}\big)\mathbf{u}^{(m+1)} = \mathbf{v}^{(m+1)}, \end{cases}$$

where

$$\widetilde{\mathbf{b}}^{(m+1)} = \big((I_{N_2}+\widehat{A}_y)\otimes(I_{N_1}+\widehat{A}_x)\big)\mathbf{u}^{(m)} + \tau\mathbf{f}^{(m+1/2)}.$$

Let

$$\underline{A}_x = I_{N_2}\otimes(I_{N_1}-\widehat{A}_x), \qquad \underline{A}_y = (I_{N_2}-\widehat{A}_y)\otimes I_{N_1}.$$

The above linear systems can be written as

$$\begin{cases} \underline{A}_x\mathbf{v}^{(m+1)} = \widetilde{\mathbf{b}}^{(m+1)}, \\ \underline{A}_y\mathbf{u}^{(m+1)} = \mathbf{v}^{(m+1)}. \end{cases} \tag{2.6}$$

3 Preconditioned iterative methods

In this section, we propose the preconditioned GMRES method and the preconditioned CGNR method to solve the discretized linear systems (2.3) and (2.6). Notice that the Toeplitz matrices $A_{N_1}^{\alpha,L}$, $A_{N_1}^{\alpha,R}$, $A_{N_2}^{\beta,L}$, and $A_{N_2}^{\beta,R}$ have a special diagonal decay property: the (i,j)th entry of each of the above-mentioned matrices is of order $\mathcal{O}(|i-j|^{-\gamma})$, where $\gamma=\alpha,\beta$; refer to (2.1). Thus, we consider constructing preconditioners based on banded approximation of the coefficient matrix. As diagonal compensation is a useful technique in designing preconditioners [1,20], we also consider using such technique.

First we consider banded preconditioners for the discretized systems (2.3). Let

$$A_{Q,k}^{\gamma,L} = \begin{bmatrix} w_1^{(\gamma)} & w_0^{(\gamma)} & & & \\ \vdots & w_1^{(\gamma)} & w_0^{(\gamma)} & & \\ w_k^{(\gamma)} & & \ddots & \ddots & \\ & \ddots & & \ddots & w_0^{(\alpha)} \\ & & w_k^{(\gamma)} & \cdots & w_1^{(\gamma)} \end{bmatrix}_{Q\times Q}, \qquad A_{Q,k}^{\gamma,R} = \big(A_{Q,k}^{\gamma,L}\big)^T.$$

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

Let $C_Q^{(\gamma)}$ be the diagonal matrices defined by

$$C_Q^{\gamma}=\begin{bmatrix} 0 & & & & & \\ & \ddots & & & & \\ & & 0 & & & \\ & & & w_{k+1}^{(\gamma)} & & \\ & & & & \ddots & \\ & & & & & \sum_{j=k+1}^{Q} w_j^{(\gamma)} \end{bmatrix}_{Q\times Q}.$$

The matrices $A_{x,k}^{(m+1)}$ and $A_{y,k}^{(m+1)}$ are defined by

$$\begin{cases} A_{x,k}^{(m+1)}=D_+^{(m+1)}\Big(I_{N_2}\otimes\big(A_{N_1,k}^{\alpha,L}+C_{N_1}^{\alpha}\big)\Big)+D_-^{(m+1)}\Big(I_{N_2}\otimes\big(A_{N_1,k}^{\alpha,R}+C_{N_1}^{\alpha}\big)\Big), \\ A_{y,k}^{(m+1)}=E_+^{(m+1)}\Big(\big(A_{N_2,k}^{\beta,L}+C_{N_2}^{\beta}\big)\otimes I_{N_1}\Big)+E_-^{(m+1)}\Big(\big(A_{N_2,k}^{\beta,R}+C_{N_2}^{\beta}\big)\otimes I_{N_1}\Big). \end{cases}$$

Then the matrix $A^{(m+1)}$ can be split into two parts

$$A^{(m+1)}=A_k^{(m+1)}+B_k^{(m+1)},$$

where

$$A_k^{(m+1)}=A_{x,k}^{(m+1)}+A_{y,k}^{(m+1)},\qquad B_k^{(m+1)}=A^{(m+1)}-A_k^{(m+1)}.$$

The matrix $A_k^{(m+1)}$ is a sparse banded matrix containing $4k-3$ nonzero diagonals of $A^{(m+1)}$ only, while the bandwidth of $I_{N^2}-A_k^{(m+1)}$ is $2N_1(k-1)+1$. Since $w_k^{(\gamma)}=\mathcal{O}\big(k^{-(\gamma+1)}\big)$ for $\gamma=\alpha,\beta$, it follows that

$$\frac{\|(I_{N^2}-A^{(m+1)})-(I_{N^2}-A_k^{(m+1)})\|_\infty}{\|I_{N^2}-A^{(m+1)}\|_\infty}=\frac{\|B_k^{(m+1)}\|_\infty}{\|I_{N^2}-A^{(m+1)}\|_\infty}=\mathcal{O}\big(k^{-\varsigma}\big),$$

where $\varsigma=\min\{\alpha,\beta\}$. The relative distance between $I_{N^2}-A_k^{(m+1)}$ and $I_{N^2}-A^{(m+1)}$ is very small if k is sufficiently large. Thus the sparse banded matrix $I_{N^2}-A_k^{(m+1)}$ is a good approximation of the coefficient matrix $\underline{A}^{(m+1)}$. Therefore, we can consider constructing preconditioners for $\underline{A}^{(m+1)}$ by using the incomplete LU factorization of the sparse banded matrix $I_{N^2}-A_k^{(m+1)}$.

3.1 Preconditioned GMRES method

Since the banded matrix $I_{N^2}-A_k^{(m+1)}$ has $4k-3$ nonzero diagonals only, it is a very sparse matrix. However, the bandwidth of $I_{N^2}-A_k^{(m+1)}$ is $2N_1(k-1)+1$. It is very expensive to

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

calculate its inverse or LU factorization since the sparsity of the matrix is destroyed. To overcome this difficulty, we calculate incomplete LU factorization with no fill-in (ILU(0)) of the banded matrix $I_{N^2}-A_k^{(m+1)}$. Let L and U be the factors of its incomplete ILU(0) factorization, then both of them have $2k-1$ nonzero diagonals only. Thus, $P_k^{(m+1)}:=LU$ is also a sparse matrix. Since the matrix $P_k^{(m+1)}$ can be a good approximation of $I_{N^2}-A_k^{(m+1)}$, it can also be a good approximation of the coefficient matrix $\underline{A}^{(m+1)}$. Thus, instead of solving the linear system (2.3), we can solve the following preconditioned linear system

$$\left(P_k^{(m+1)}\right)^{-1}\underline{A}^{(m+1)}\mathbf{u}^{(m+1)}=\left(P_k^{(m+1)}\right)^{-1}\mathbf{b}^{(m+1)}. \tag{3.1}$$

For the sake of completeness, we state the preconditioned method GMRES(ρ) with residual checking at each iteration for the preconditioned linear system (3.1); see [20]. Here we use the following notations: the symbols $\mathbf{r}_t$ and $\mathbf{r}_p$ denote the current residual of the original linear system and that of the preconditioned linear system, respectively. Let $IterMax$ stand for the maximal number of iteration, and $\epsilon>0$ denote the given relative accuracy of the residual. The GMRES method is restarted after ρ iterations.

Preconditioned GMRES(ρ) Algorithm

At each time step $t^{(m+1)}$, we choose a $\mathbf{u}_0$ as an initial guess for $\mathbf{u}^{(m+1)}$
Set $\mu:=0$, and compute the ILU(0) factorization of $I_{N^2}-A_k^{(m+1)}$: $P^{(m+1)}=LU$
Compute $\mathbf{r}:=\mathbf{b}^{(m+1)}-\underline{A}^{(m+1)}\mathbf{u}^{(m+1)}$, and assign $\mathbf{r}_t:=\mathbf{r}$
While $\mu\le IterMax$ and $\|\mathbf{r}_t\|_2/\|\mathbf{b}^{(m+1)}\|_2>\epsilon$ do
  $\mu:=\mu+1$
  Compute $\mathbf{r}_p:=U^{-1}L^{-1}\mathbf{r}$, $\beta:=\|\mathbf{r}_p\|$, $\mathbf{v}_1:=\mathbf{r}_p/\beta$
  Assign $j:=0$ and $V_1:=\mathbf{v}_1$
  While $j\le\rho$ and $\|\mathbf{r}_t\|_2/\|\mathbf{b}^{(m+1)}\|_2>\epsilon$ do
    $j:=j+1$
    Compute $\mathbf{w}:=U^{-1}L^{-1}\underline{A}^{(m+1)}\mathbf{v}_j$
    For $i=1,\cdots,j$ do
      $h_{i,j}=\mathbf{v}_i^T*\mathbf{w}$
      $\mathbf{w}:=\mathbf{w}-h_{i,j}\mathbf{v}_i$
    Enddo
    Compute $h_{j+1,j}=\|\mathbf{w}\|_2$ and $\mathbf{v}_{j+1}:=\mathbf{w}/h_{j+1,j}$
    Assign $V_{j+1}:=[V_j,\mathbf{v}_{j+1}]$ and $H_j:=[h_{\gamma,\delta}]_{1\le\gamma\le j+1,1\le\delta\le j}$
    Compute $\mathbf{y}_j:=\mathrm{argmin}_{\mathbf{y}}\|\beta\mathbf{e}_1-H_j\mathbf{y}\|_2$
    Compute the residual $\mathbf{r}_t:=\mathbf{r}-LUV_{j+1}H_j\mathbf{y}_j$
  Enddo
  $\mathbf{r}:=\mathbf{r}_t$
  $\mathbf{u}^{(m+1)}:=\mathbf{u}^{(m+1)}+V_j\mathbf{y}_j$
Enddo

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

In the preconditioned GMRES method, the main computational works at each iteration are as follows:

1. Compute $\mathbf{w}=U^{-1}L^{-1}\underline{A}^{(m+1)}\mathbf{v}_j$;
2. Compute $\mathbf{r}_t=\mathbf{r}-LUV_{j+1}H_j\mathbf{y}_j$.

Since $\underline{A}^{(m+1)}$ has a special Toeplitz-like structure, the matrix-vector multiplication $\underline{A}^{(m+1)}\mathbf{v}$ can be calculated very efficiently [10,17]. Notice that

$$\begin{aligned}\underline{A}^{(m+1)}\mathbf{v} &= \mathbf{v}-D_+^{(m+1)}\big(I_{N_2}\otimes A_L^{\alpha,N_1}\big)\mathbf{v}-D_-^{(m+1)}\big(I_{N_2}\otimes A_R^{\alpha,N_1}\big)\mathbf{v}\\ &\quad -E_+^{(m+1)}\big(A_L^{\beta,N_2}\otimes I_{N_1}\big)\mathbf{v}-E_-^{(m+1)}\big(A_R^{\beta,N_2}\otimes I_{N_1}\big)\mathbf{v},\end{aligned}$$

and

$$I_{N_2}\otimes A_R^{\alpha,N_1}=\big(I_{N_2}\otimes A_L^{\alpha,N_1}\big)^T,\qquad A_R^{\beta,N_2}\otimes I_{N_1}=\big(A_L^{\beta,N_2}\otimes I_{N_1}\big)^T.$$

The main costs of computing $\underline{A}^{(m+1)}\mathbf{v}$ are $\big(I_{N_2}\otimes A_L^{\alpha,N_1}\big)\mathbf{v}$, $\big(I_{N_2}\otimes A_R^{\alpha,N_1}\big)\mathbf{v}$, $\big(A_L^{\beta,N_2}\otimes I_{N_1}\big)\mathbf{v}$, and $\big(A_R^{\beta,N_2}\otimes I_{N_1}\big)\mathbf{v}$. The multiplications $\big(I_{N_2}\otimes A_L^{\alpha,N_1}\big)\mathbf{v}$ and $\big(I_{N_2}\otimes A_R^{\alpha,N_1}\big)\mathbf{v}$ can be calculated simultaneously by computing $\big(I_{N_2}\otimes A_L^{\alpha,N_1}\big)(\mathbf{v}+\sqrt{-1}\check{\mathbf{v}})$, where $\check{\mathbf{v}}=(v_N,v_{N-1},\cdots,v_1)^T$ [29]. Similarly, we can also calculate $\big(A_L^{\beta,N_2}\otimes I_{N_1}\big)\mathbf{v}$ and $\big(A_R^{\beta,N_2}\otimes I_{N_1}\big)\mathbf{v}$ simultaneously. By using the FFT, the matrix-vector multiplication $\underline{A}^{(m+1)}\mathbf{v}$ needs $\mathcal{O}(N^2\log N)$ operations only. In addition, since both L and U are sparse triangular matrices with $2k-1$ nonzero diagonals only, the computations of $L\mathbf{x}$, $U\mathbf{x}$, $L^{-1}\mathbf{x}$, and $U^{-1}\mathbf{x}$ require $\mathcal{O}(N^2k)$ operations only. Therefore, computing $\mathbf{w}=U^{-1}L^{-1}\underline{A}^{(m+1)}\mathbf{v}_j$ requires $\mathcal{O}(N^2\log N)$ operations. As for computation of $\mathbf{r}_t$, it can be obtained in $\mathcal{O}(N^2(j+k))$ operations since V_{j+1} and H_j are matrices of sizes $N^2\times(j+1)$ and $(j+1)\times j$.

3.2 Preconditioned CGNR method

Another efficient method to solve the linear system (2.3) is to solve its normal equation by the preconditioned conjugate gradient method, i.e., by the preconditioned CNGR method. The normal equation of (2.3) is

$$\big(\underline{A}^{(m+1)}\big)^T\underline{A}^{(m+1)}\mathbf{u}^{(m+1)}=\big(\underline{A}^{(m+1)}\big)^T\mathbf{b}^{(m+1)}. \tag{3.2}$$

If the diffusion coefficients $d_\pm(x,y,t)$ and $e_\pm(x,y,t)$ are not sufficiently small, then the matrix $\big(\underline{A}^{(m+1)}\big)^T\underline{A}^{(m+1)}$ may be ill-conditioned if the grid sizes are small. In addition, the eigenvalues of this matrix are not clustered around any special point. Thus, the CGNR method converges very slowly. In order to accelerate the convergence, an efficient preconditioner is needed. In this paper, we choose $\big(P_k^{(m+1)}\big)^TP_k^{(m+1)}$ as the preconditioner for the normal equation (3.2).

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

The main points of the preconditioned CGNR method are as follows.

(i) The main works at each iteration are the matrix-vector multiplication $\left(\underline{A}^{(m+1)}\right)^T \underline{A}^{(m+1)}\mathbf{v}$ and the solution of the preconditioned system $\left(P_k^{(m+1)}\right)^T P_k^{(m+1)}\mathbf{w}=\mathbf{z}$.

(ii) Similar to the preconditioned GMRES method, we can calculate the matrix-vector multiplication $\left(\underline{A}^{(m+1)}\right)^T \underline{A}^{(m+1)}\mathbf{v}$ efficiently by using the special Toeplitz-like structure of the coefficient matrices.

3.3 Preconditioners for ADI method

We consider banded preconditioners for the discretized linear systems (2.5) and (2.6). Actually, the preconditioned iterative methods proposed in [20] can be applied to the D'Yakonov ADI scheme (2.5) straightforwardly. For the discretized linear systems (2.6), the preconditioners for the coefficient matrices $\underline{A}_x$ and $\underline{A}_y$ are given by

$$P_k^x = I_{N_2}\otimes W_k^x, \qquad P_k^y = W_k^y \otimes I_{N_1},$$

where

$$\begin{cases} W_k^x = I_{N_1} - D_+\left(A_{N_1,k}^{\alpha,L}+C_{N_1}^{\alpha}\right) - D_-\left(A_{N_1,k}^{\alpha,R}+C_{N_1}^{\alpha}\right), \\ W_k^y = I_{N_2} - E_+\left(A_{N_2,k}^{\beta,L}+C_{N_2}^{\beta}\right) - E_-\left(A_{N_2,k}^{\beta,R}+C_{N_2}^{\beta}\right). \end{cases}$$

Thus, we can solve the following preconditioned linear systems

$$\begin{cases} \left(P_k^x\right)^{-1}\underline{A}_x\mathbf{v}^{(m+1)} = \left(P_k^x\right)^{-1}\widetilde{\mathbf{b}}^{(m+1)}, \\ \left(P_k^y\right)^{-1}\underline{A}_y\mathbf{u}^{(m+1)} = \left(P_k^y\right)^{-1}\mathbf{v}^{(m+1)}, \end{cases}$$

by using GMRES method instead of solving the linear systems (2.6). The normal equations of the linear systems (2.6) are given by

$$\begin{cases} \left(\underline{A}_x\right)^T\underline{A}_x\mathbf{v}^{(m+1)} = \left(\underline{A}_x\right)^T\widetilde{\mathbf{b}}^{(m+1)}, \\ \left(\underline{A}_y\right)^T\underline{A}_y\mathbf{u}^{(m+1)} = \left(\underline{A}_y\right)^T\mathbf{v}^{(m+1)}. \end{cases}$$

For the above linear systems, we choose $\left(P_k^x\right)^T P_k^x$ and $\left(P_k^y\right)^T P_k^y$ as preconditioners, and apply the preconditioned CGNR method to solve these normalized linear systems.

Remark. Let the LU factorizations of W_k^x and W_k^y be given by

$$W_k^x = L_k^x U_k^x, \qquad W_k^y = L_k^y U_k^y.$$

Then the LU factorizations of P_k^x and P_k^y are given by

$$P_k^x = \left(I_{N_2}\otimes L_k^x\right)\left(I_{N_2}\otimes U_k^x\right), \qquad P_k^y = \left(L_k^y\otimes I_{N_1}\right)\left(U_k^y\otimes I_{N_1}\right).$$

Thus, unlike the preconditioners $P_k^{(m+1)}$ for $\underline{A}^{(m+1)}$, the LU factorizations of both P_k^x and P_k^y can be calculated very efficiently.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

4 Numerical results

In this section, we present some numerical examples to illustrate the performance of preconditioned iterative methods versus iterative methods without preconditioners. For the CN-WSGD scheme (2.3), we choose the initial guess as

$$\mathbf{u}_0 = \begin{cases} \mathbf{u}^{(0)}, & m=0, \\ \tilde{\mathbf{u}}^{(m+1)} \equiv 2\mathbf{u}^{(m)} - \mathbf{u}^{(m-1)}, & m>0. \end{cases}$$

Since $\tilde{\mathbf{u}}^{(m+1)}$ is a good approximation of $\mathbf{u}^{(m+1)}$ [41], we set it as an initial guess. The stopping criterion is

$$\frac{\|\mathbf{r}_j\|_2}{\|\mathbf{b}^{(m+1)}\|_2} < 10^{-7},$$

where $\mathbf{r}_j$ is the residual vector after jth iteration. In our numerical experiments, we set the parameter $k=4$.

For the D'Yakonov ADI method (2.6), we choose the initial guesses as

$$\mathbf{v}_0 = \begin{cases} \underline{A}_y \mathbf{u}^{(0)}, & m=0, \\ \tilde{\mathbf{v}}^{(m+1)} \equiv \underline{A}_y \left(2\mathbf{u}^{(m)} - \mathbf{u}^{(m-1)}\right), & m>0, \end{cases}$$

and

$$\mathbf{u}_0 = \begin{cases} \mathbf{u}^{(0)}, & m=0, \\ \tilde{\mathbf{u}}^{(m+1)} \equiv 2\mathbf{u}^{(m)} - \mathbf{u}^{(m-1)}, & m>0. \end{cases}$$

The stopping criterions are

$$\frac{\|\widetilde{\mathbf{r}}_j\|_2}{\|\widetilde{\mathbf{b}}^{(m+1)}\|_2} < 10^{-7}, \qquad \frac{\|\widehat{\mathbf{r}}_j\|_2}{\|\mathbf{v}^{(m+1)}\|_2} < 10^{-7},$$

where $\widetilde{\mathbf{r}}_j$ and $\widehat{\mathbf{r}}_j$ are the residual vectors after jth iteration. For the D'Yakonov ADI method (2.6), we set the parameter $k=8$.

In the following tables, we use "N_1" and "N_2" to denote the numbers of spatial partitions in x-direction and y-direction respectively, and use "M" to denote the number of temporal partition. The maximal error between the true solution and the numerical solution at the last time step is denoted by "Error". In order to illustrate the efficiency of our preconditioners, we also present the performance of the GMRES(20) method and the CGNR method for comparison. All numerical tests are run by using MATLAB R2010a on a Dell computer Intel(R) Core(TM)i7-2600 Duo of 3.40 GHz CPU and 16GB RAM.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

Example 4.1. Consider Eq. (1.1) with $\alpha=1.8$, $\beta=1.9$. The left-sided and right-sided diffusion coefficients are given by

$$d_+(x,y,t)=4(1+t)x^{1.8}(1+y), \qquad d_-(x,y,t)=4(1+t)(1-x)^{1.8}(1+y),$$
$$e_+(x,y,t)=4(1+t)(1+x)y^{1.9}, \qquad e_-(x,y,t)=4(1+t)(1+x)(1-y)^{1.9}.$$

The spatial domain is $\Omega=(0,1)\times(0,1)$ and the time interval is $[0,T]=[0,1]$. The initial condition is $u_0(x,y)=x^3y^3(1-x)^3(1-y)^3$. The source term is given by

$$\begin{aligned} f(x,y,t)=-e^{-t}\Big[&\Big(x^3(1-x)^3y^3(1-y)^3\Big)+4(1+t)\Big(\tfrac{\Gamma(4)}{\Gamma(2.2)}\big(x^3+(1-x)^3\big)-\tfrac{3\Gamma(5)}{\Gamma(3.2)}\big(x^4+(1-x)^4\big)\\ &+\tfrac{3\Gamma(6)}{\Gamma(4.2)}\big(x^5+(1-x)^5\big)-\tfrac{\Gamma(7)}{\Gamma(5.2)}\big(x^6+(1-x)^6\big)\Big)y^3(1-y)^3(1+y)\\ &+4(1+t)\Big(\tfrac{\Gamma(4)}{\Gamma(2.1)}\big(y^3+(1-y)^3\big)-\tfrac{3\Gamma(5)}{\Gamma(3.1)}\big(y^4+(1-y)^4\big)\\ &+\tfrac{3\Gamma(6)}{\Gamma(4.1)}\big(y^5+(1-y)^5\big)-\tfrac{\Gamma(7)}{\Gamma(5.1)}\big(y^6+(1-y)^6\big)\Big)x^3(1-x)^3(1+x)\Big]. \end{aligned}$$

The solution to the fractional diffusion equation is

$$u(x,y,t)=e^{-t}x^3(1-x)^3y^3(1-y)^3.$$

Numerical results for Example 4.1 are shown in Tables 1-4. Tables 1-2 show numerical results for solving the CN-WSGD scheme (2.3), including the average numbers of iterations required by GMRES(20), PGMRES(20), CGNR, and PCGNR methods, the accuracy of numerical solutions, and the CPU times required by different methods. Tables 3-4 present numerical results for solving the D'Yakonov ADI scheme (2.5).

From Tables 1-4, we see that the GMRES(20) and the CGNR methods require much more iterations than the PGMRES(20) and the PCGNR methods, especially the CGNR method. The numbers of iterations required by the PGMRES(20) and the PCGNR methods increase slowly (compared to the iterative methods without preconditioners) as N_1 and N_2 increase. For the GMRES and GMRES(20) methods, the CN-WSGD scheme is more efficient than the D'Yakonov ADI scheme in terms of CPU times.

Table 1: The average numbers of iterations by different methods for Example 4.1: Eq. (2.3).

$N_1=N_2=M$	GMRES	PGMRES	CGNR	PCGNR	Error
2^4	48.750	11.000	181.625	18.688	1.1386×10^{-6}
2^5	81.594	12.406	599.875	26.969	3.0206×10^{-7}
2^6	157.750	14.250	1658.094	37.266	7.6918×10^{-8}
2^7	273.914	17.055	3984.930	47.633	1.9381×10^{-8}
2^8	416.625	20.477	7160.086	56.469	4.8489×10^{-9}

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

Table 2: The CPU times of different methods for Example 4.1: Eq. (2.3).

$N_1=N_2=M$	GMRES	PGMRES	CGNR	PCGNR
2^4	0.219 s	0.093 s	1.092 s	0.156 s
2^5	1.357 s	0.405 s	11.965 s	0.843 s
2^6	14.181 s	2.449 s	204.891 s	7.316 s
2^7	165.407 s	19.562 s	3366.755 s	66.768 s
2^8	2480.888 s	206.950 s	62435.475 s	786.132 s

Table 3: The average numbers of iterations by different methods for Example 4.1: Eq. (2.5).

$N_1=N_2=M$	GMRES	PGMRES	CGNR	PCGNR	Error
2^4	19.904	3.385	16.000	6.850	1.0456×10^{-6}
2^5	77.073	4.024	37.119	8.146	2.9276×10^{-7}
2^6	149.846	4.650	108.289	8.961	7.5861×10^{-8}
2^7	275.846	5.611	318.806	8.918	1.9195×10^{-8}
2^8	465.764	6.608	1001.709	8.678	4.7942×10^{-9}

Table 4: The CPU times of different methods for Example 4.1: Eq. (2.5).

$N_1=N_2=M$	GMRES	PGMRES	CGNR	PCGNR
2^4	1.9190 s	0.0296 s	1.4040 s	0.0730 s
2^5	19.7970 s	1.4820 s	12.9480 s	3.8220 s
2^6	159.3390 s	8.9230 s	154.1910 s	19.7810 s
2^7	1268.1420 s	68.9360 s	1993.7620 s	110.8850 s
2^8	10536.1130 s	834.1170 s	31884.4220 s	1122.0430 s

Example 4.2. Consider Eq. (1.1) with $\alpha=1.8$, $\beta=1.9$. The left-sided and right-sided diffusion coefficients are given by

$$d_+(x,y,t)=6x^{1.8}, \qquad d_-(x,y,t)=6(1-x)^{1.8},$$
$$e_+(x,y,t)=6y^{1.9}, \qquad e_-(x,y,t)=6(1-y)^{1.9}.$$

The spatial domain is $\Omega=(0,1)\times(0,1)$ and the time interval is $[0,T]=[0,1]$. The initial condition is $u_0(x,y)=x^3y^3(1-x)^3(1-y)^3$. The source term is given by

$$\begin{aligned} f(x,y,t)=-e^{-t}\Big[&\Big(x^3(1-x)^3y^3(1-y)^3\Big)+6\Big(\tfrac{\Gamma(4)}{\Gamma(2.2)}\big(x^3+(1-x)^3\big)-\tfrac{3\Gamma(5)}{\Gamma(3.2)}\big(x^4+(1-x)^4\big)\\ &+\tfrac{3\Gamma(6)}{\Gamma(4.2)}\big(x^5+(1-x)^5\big)-\tfrac{\Gamma(7)}{\Gamma(5.2)}\big(x^6+(1-x)^6\big)\Big)y^3(1-y)^3\\ &+6\Big(\tfrac{\Gamma(4)}{\Gamma(2.1)}\big(y^3+(1-y)^3\big)-\tfrac{3\Gamma(5)}{\Gamma(3.1)}\big(y^4+(1-y)^4\big)\\ &+\tfrac{3\Gamma(6)}{\Gamma(4.1)}\big(y^5+(1-y)^5\big)-\tfrac{\Gamma(7)}{\Gamma(5.1)}\big(y^6+(1-y)^6\big)\Big)x^3(1-x)^3\Big]. \end{aligned}$$

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

Table 5: The average numbers of iterations by different methods for Example 4.2: Eq. (2.3).

$N_1=N_2=M$	GMRES	PGMRES	CGNR	PCGNR	Error
2^4	36.000	11.000	72.688	18.875	1.1465×10^{-6}
2^5	63.694	12.250	240.469	25.344	2.9185×10^{-7}
2^6	113.234	13.813	751.734	33.688	7.4451×10^{-8}
2^7	173.008	15.531	2035.477	41.430	1.8769×10^{-8}
2^8	231.719	17.465	3665.578	44.465	4.7132×10^{-9}

Table 6: The CPU times of different methods for Example 4.2: Eq. (2.3).

$N_1=N_2=M$	GMRES	PGMRES	CGNR	PCGNR
2^4	0.203 s	0.094 s	0.515 s	0.172 s
2^5	1.123 s	0.374 s	5.351 s	0.858 s
2^6	10.437 s	2.403 s	93.413 s	6.724 s
2^7	109.590 s	19.219 s	1588.021 s	58.891 s
2^8	1387.772 s	192.848 s	31837.512 s	708.398 s

Table 7: The average numbers of iterations by different methods for Example 4.2: Eq. (2.6).

$N_1=N_2=M$	GMRES	PGMRES	CGNR	PCGNR	Error
2^4	16.000	7.000	29.562	7.000	3.0886×10^{-6}
2^5	32.000	8.063	65.219	8.094	7.8083×10^{-7}
2^6	182.125	9.031	176.375	9.094	1.9642×10^{-7}
2^7	311.805	11.023	509.500	11.063	4.9238×10^{-8}
2^8	476.441	19.008	1496.227	11.020	1.2286×10^{-8}

The solution to the fractional diffusion equation is

$$u(x,y,t)=e^{-t}x^3(1-x)^3y^3(1-y)^3.$$

Numerical results for Example 4.2 are shown in Tables 5-8. Tables 5-6 show numerical results for solving the CN-WSGD scheme (2.3), including the average numbers of iterations required by GMRES(20), PGMRES(20), CGNR, and PCGNR methods, the accuracy of numerical solutions, and the CPU times required by different methods. Tables 7-8 present numerical results for solving the D'Yakonov ADI scheme (2.6).

From Tables 5-8, we see again that the GMRES(20) and the CGNR methods require much more iterations than the PGMRES(20) and the PCGNR methods. The numbers of iterations required by the PGMRES(20) and the PCGNR methods increase slowly as N_1 and N_2 increase. For the CGNR and PCGNR method, the D'Yakonov ADI scheme is more efficient than the CN-WSGD scheme in terms of CPU times and number of iterations. In the other hand, numerical solutions obtained by using the CN-WSGD scheme are slightly better than those obtained by using the ADI scheme.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

Table 8: The CPU times of different methods for Example 4.2: Eq. (2.6).

$N_1 = N_2 = M$	GMRES	PGMRES	CGNR	PCGNR
2^4	0.094 s	0.062 s	0.156 s	0.0780 s
2^5	0.514 s	0.265 s	0.999 s	0.3119 s
2^6	11.404 s	1.654 s	12.730 s	1.9970 s
2^7	137.015 s	13.026 s	218.011 s	14.929 s
2^8	1872.572 s	161.772 s	6380.770 s	132.351 s

Acknowledgments

The authors would like to thank the referees for their very valuable comments and suggestions which made the paper more presentable. X. Jin was supported by the research grant MYRG098(Y2-L3)-FST13-JXQ from University of Macau, and F. Lin was supported by the National Natural Science Foundation of China (Project No. 11271238).

References

[1] O. Axelsson, Iterative Solution Methods, Cambridge University Press, 1996.
[2] J. Bai and X. Feng, Fractional-order anisotropic diffusion for image denoising, IEEE Tran. Image Process., 16 (2007) 2492–2502.
[3] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. V. der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, 1994.
[4] T. S. Basu and H. Wang, A fast second-order finite difference method for space-fractional diffusion equations, Int. J. Numer. Anal. Model., 9 (2012) 658–666.
[5] D. Benson, S. W. Wheatcraft and M. M. Meerchaert, Applications of a fractional advection-dispersion equation, Water Resour. Res., 36 (2000) 1403–1413.
[6] D. Benson, S. W. Wheatcraft and M. M. Meerschaert, The fractional-order governing equation of Lévy motion, Water Resour. Res., 36 (2000) 1413–1423.
[7] D. Benson, M. Kovács and M. M. Meerchaert, Numerical solutions for fractional reaction-diffusion equations, Comput. Math. Appl., 55 (2008) 2212–2226.
[8] B. A. Carreras, V. E. Lynch and G. M. Zaslavsky, Anomalous diffusion and exit time distribution of particle tracers in plasma turbulence model, Phys. Plasmas, 8 (2001) 5096–5103.
[9] R. Chan, Q. Chang and H. Sun, Multigrid method for ill-conditioned symmetric Toeplitz systems, SIAM J. Sci. Statist. Comput., 19 (1998) 516–529.
[10] R. Chan and X. Jin, An Introduction to Iterative Toeplitz Solvers, SIAM, Philadelphia, 2007.
[11] R. Chan and M. Ng, Conjugate gradient methods for Toeplitz systems, SIAM Rev., 38 (1996) 427–482.
[12] M. Chen and W. Deng, Fourth order difference approximations for space Riemann-Liouville derivatives based on weighted and shifted Lubich difference operators, Commun. Comput. Phys., 16 (2014) 516–540.
[13] W. Deng and M. Chen, Efficient numerical algorithms for three-dimensional fractional partial differential equations, J. Comp. Math., 32 (2014) 371–391.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

[14] V. J. Ervin, N. Heuer and J. P. Roop, Numerical approximation of a time dependent, nonlinear, space-fractional diffusion equation, SIAM J. Numer. Anal., 45 (2007) 572–591.
[15] N. J. Ford and A. C. Simpson, The numerical solution of fractional differential equations: speed versus accuracy, Numer. Algorithms, 26 (2001) 333–346.
[16] X. Ji and H. Tang, High-order accurate Runge-Kutta (local) discontinuous Galerkin methods for one- and two-dimensional fractional diffusion equations, Numer. Math. Theor. Meth. Appl., 5 (2012) 333–358.
[17] X. Jin, Preconditioning Techniques for Toeplitz Systems, Higher Education Press, Beijing, 2010.
[18] T. A. M. Langlands, B. I. Henry and S. L. Wearne, Fractional cable equation models for anomalous electrodiffusion in nerve cells: infinite domain solutions, J. Math. Biol., 59 (2009) 761–808.
[19] S. Lei and H. Sun, A circulant preconditioner for fractional diffusion equation, J. Comput. Phys., 242 (2013) 715–725.
[20] F. Lin, S. Yang and X. Jin, Preconditioned iterative methods for fractional diffusion equation, J. Comput. Phys., 256 (2014) 109–117.
[21] F. Liu, V. Anh and I. Turner, Numerical solution of the space fractional Fokker-Planck equation, J. Comput. Appl. Math., 166 (2004) 209–219.
[22] R. L. Magin, Fractional Calculus in Bioengineering, Begell House Publishers, 2006.
[23] F. Mainardi, Fractals and fractional calculus continuum mechanics, Springer Verlag, 1997, pp. 291–348.
[24] M. M. Meerschaert and C. Tadjeran, Finite difference approximations for fractional advection-dispersion flow equations, J. Comput. Phys., 172 (2004) 65–77.
[25] M. M. Meerschaert and C. Tadjeran, Finite difference approximations for two-sided space-fractional partial differential equations, Appl. Numer. Math., 56 (2006) 80–90.
[26] 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.
[27] D. A. Murio, Implicit finite difference approximation for time fractional diffusion equations, Comput. Math. Appl., 56 (2008) 1138–1145.
[28] K. B. Oldham, Fractional differential equations in electrochemistry, Adv. Engrg. Software, 41 (2010) 9–12 (Civil-Comp Special Issue).
[29] H. Pang and H. Sun, Multigrid iterative methods for fractional diffusion equations, J. Comput. Phys., 231 (2012) 693–703.
[30] I. Podlubny, Fractional Differential Equations, Academic Press, New York, 1999.
[31] M. Raberto, E. Scalas and F. Mainardi, Waiting-times and returns in high-frequency financial data: an empirical study, Physica A, 314 (2002) 749–755.
[32] J. Ren and Z. Sun, Efficient and stable numerical methods for multi-term time fractional sub-diffusion equations, East Asian Journal on Applied Mathematics, 4 (2014) 242–266.
[33] K. Ritchie, X. Shan, J. Kondo, K. Iwasawa, T. Fujiwara and A. Kusumi, Detection of non-Brownian diffusion in the cell membrane in single molecule tracking, Biophysical J., 88 (2005) 2266–2277.
[34] J. P. Roop, Computational aspects of FEM approximation of fractional advection dispersion equations on bounded domains in R^2, J. Comput. Appl. Math., 193 (2006) 243–268.
[35] H. Scher and E. W. Montroll, Anomalous transit-time dispersion in amorphous solids, Phys. Rev. B, 12 (1975) 2455–2477.
[36] I. M. Sokolov, J. Klafter and A. Blumen, Fractional kinetics, Phys. Today (Nov. 2002), pp. 48–54.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press

[37] E. Sousa, Finite difference approximates for a fractional advection diffusion problem, J. Comput. Phys., 228 (2009) 4038–4054.
[38] C. Tadjeran, M. M. Meerschaert and H. P. Scheffler, A second-order accurate numerical approximation for the fractional diffusion equations, J. Comput. Phys., 213 (2006) 205–213.
[39] W. Tian, H. Zhou and W. Deng, A class of second order difference approximation for solving space fractional diffusion equations, Math. Comp., 84 (2015) 1703–1727.
[40] H. Wang and K. Wang, An $O(N\log^2 N)$ alternating-direction finite difference method for two-dimensional fractional diffusion equations, J. Comput. Phys., 230 (2011) 7830–7839.
[41] 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.
[42] K. Wang and H. Wang, A fast characteristic finite difference method for fractional advection-diffusion equations, Adv Water Resour., 34 (2011) 810–816.
[43] H. Wang and T. S. Basu, A fast finite difference method for two-dimensional space-fractional diffusion equations, SIAM J. Sci. Comput., 34 (2012) 2444–2458.
[44] N. Zhang, W. Deng and Y. Wu, Finite difference/element method for a two-dimensional modified fractional diffusion equation, Adv. Appl. Math. Mech., 4 (2012) 496–518.

https://doi.org/10.4208/cicp.120314.230115a Published online by Cambridge University Press