Journal of Computational Physics 303 (2015) 203–211

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A fast direct method for block triangular Toeplitz-like with tri-diagonal block systems from time-fractional partial differential equations [☆]

Rihuan Ke [a], Michael K. Ng [b,*], Hai-Wei Sun [c]

[a] School of Mathematics, South China Normal University, China
[b] Department of Mathematics, Hong Kong Baptist University, Hong Kong
[c] Department of Mathematics, University of Macau, Macao

ARTICLE INFO

Article history:
Received 1 May 2015
Received in revised form 10 September 2015
Accepted 24 September 2015
Available online xxxx

Keywords:
Block triangular Toeplitz-like matrix
Direct methods
Divide-and-conquer strategy
Fast Fourier transform
Fractional partial differential equations

ABSTRACT

In this paper, we study the block lower triangular Toeplitz-like with tri-diagonal blocks system which arises from the time-fractional partial differential equation. Existing fast numerical solver (e.g., fast approximate inversion method) cannot handle such linear system as the main diagonal blocks are different. The main contribution of this paper is to propose a fast direct method for solving this linear system, and to illustrate that the proposed method is much faster than the classical block forward substitution method for solving this linear system. Our idea is based on the divide-and-conquer strategy and together with the fast Fourier transforms for calculating Toeplitz matrix–vector multiplication. The complexity needs $\mathcal{O}(MN\log^2 M)$ arithmetic operations, where M is the number of blocks (the number of time steps) in the system and N is the size (number of spatial grid points) of each block. Numerical examples from the finite difference discretization of time-fractional partial differential equations are also given to demonstrate the efficiency of the proposed method.

1. Introduction

Consider a linear system

$$\mathbf{A}\mathbf{u} = \mathbf{b}, \tag{1.1}$$

where $\mathbf{A}$ is the block lower triangular Toeplitz-like with tri-diagonal block (BL3TB-like) matrix of the form

$$\mathbf{A} = \begin{bmatrix} A_1^{(1)} & & & \\ A_2 & A_1^{(2)} & & \\ \vdots & \ddots & \ddots & \\ A_M & \cdots & A_2 & A_1^{(M)} \end{bmatrix}, \tag{1.2}$$

☆ The research was supported by research grants HKRGC GRF 12301214, MYRG102(Y2-L3)-FST13-SHW from University of Macau and 105/2012/A3 from FDCT of Macau.

* Corresponding author.
E-mail addresses: krhuan@gmail.com (R. Ke), mng@math.hkbu.edu.hk (M.K. Ng), HSun@umac.mo (H.-W. Sun).

http://dx.doi.org/10.1016/j.jcp.2015.09.042
0021-9991/

in which $A_1^{(1)}, A_1^{(2)}, \ldots, A_1^{(M)}$, and A_j $(j = 2, \ldots, M)$ are N-by-N tri-diagonal matrices, $\mathbf{u}$ is the unknown vector, and $\mathbf{b}$ is the right hand side vector. We remark that $A_1^{(1)}, A_1^{(2)}, \ldots, A_1^{(M)}$ are not necessarily the same. Such a linear system arises from the finite difference discretization of time-fractional partial differential equation; see [14,16,39,38,35,13,28,34] and Section 3. In the literature, there are many applications in time-fractional partial differential equations [26], for instances, chaotic dynamics of classical conservative systems [32], groundwater contaminant transport [2,3], turbulent flow [7,29], biology [23], finance [27], image processing [1], and physics [30]. Recent numerical methods for time-fractional partial differential equations can be found in [38,35,13,28,34,10–12,20,33,37].

Toeplitz matrices emerge from numerous topics such as signal and image processing, numerical solutions of partial differential equations and integral equations, as well as queueing networks; see [8,9] and the references therein. In particular, the triangular Toeplitz matrix plays a key role in the displacement representation of general Toeplitz matrices, which is fundamental in the study of structured matrices and polynomial computations; see [4,5,19,25].

Traditionally, the block forward substitution (BFS) method [18] can be straightforwardly applied to solve (1.1) in $\mathcal{O}(M^2N)$ arithmetic operations with $\mathcal{O}(MN)$ storage requirement. In order to reduce the computational cost, Lu, Pang, and Sun [22] recently proposed an approximate inversion method for solving (1.1) with $A_1^{(k)} = A_1$ for $1 \le k \le M$, where the coefficient matrix therefore becomes a block lower triangular Toeplitz with tri-diagonal block (BL3TB) matrix. Their idea is to make use of block-Toeplitz structure and approximate the BL3TB matrix by the block ϵ-circulant matrix that can be block-diagonalized by the fast Fourier transform (FFT) into a diagonal block matrix, in which each diagonal block is still tri-diagonal. The total computational complexity by their method is of $\mathcal{O}(MN \log M)$ arithmetic operations that is much cheaper than $\mathcal{O}(M^2N)$ arithmetic operations by the BFS method; see more details in [22]. Nevertheless, their approximate inversion method will be no longer available if $\mathbf{A}$ in (1.1) is not exact BL3TB; i.e., $A_1^{(k)}$ in (1.2) are different.

Another direct method for solving BL3TB system is the so-called block divide-and-conquer method [4,15]. Let $\mathbf{A}_{2k}$ be a BL3TB matrix and partitioned as

$$\mathbf{A}_{2k} = \begin{bmatrix} \mathbf{A}_k & \mathbf{0} \\ \mathbf{B}_k & \mathbf{A}_k \end{bmatrix}, \tag{1.3}$$

where $\mathbf{A}_k$ is still BL3TB and $\mathbf{B}_k$ is block Toeplitz. Thus, we have [4,15]

$$\mathbf{A}_{2k}^{-1} = \begin{bmatrix} \mathbf{A}_k^{-1} & \mathbf{0} \\ \mathbf{C}_k & \mathbf{A}_k^{-1} \end{bmatrix} \quad \text{with} \quad \mathbf{C}_k = -\mathbf{A}_k^{-1}\mathbf{B}_k\mathbf{A}_k^{-1}. \tag{1.4}$$

If $\mathbf{A}_k^{-1}$ is known, the only task is to compute $\mathbf{C}_k$, which is also a block Toeplitz matrix. Finally, $\mathbf{A}_M^{-1}$ can be recursively obtained from the inverse of a very small size matrix. In other words, the divide-and-conquer method is to compute the inverse of $\mathbf{A}_M$ exactly. The computational cost is of $\mathcal{O}(N^2M \log M + N^3M)$ and storage requirement is of $\mathcal{O}(N^2M)$ [4] since the inverse of a tri-diagonal matrix is usually dense. Therefore, the divide-and-conquer method may not be better than the BFS method for solving the BL3TB system if the block size N is large. Moreover, the divide-and-conquer method also cannot be applied for the BL3TB-like matrix if $A_1^{(k)}$ in (1.2) are different.

The main contribution of this paper is to propose a fast direct method for solving (1.2). Existing fast numerical solver (e.g., fast approximate inversion method) cannot handle such linear system as the main diagonal blocks are different. We illustrate that the proposed method is much faster than the classical block forward substitution method for solving this linear system. Our idea is to combine the BFS method with the divide-and-conquer strategy, to solve the BL3TB-like system in (1.1). As the divide-and-conquer method, the BL3TB-like matrix (1.2) is partitioned analogously to (1.3). Unlike the divide-and-conquer method, the inverse of the large size matrix is not calculated as that in (1.4). In our proposal, the partition is employed to reduce the original linear system into two half-size linear systems. Then the BFS method is exploited to solve both linear systems together. Meanwhile, the FFT can be applied to speed up the computation of the right hand side in the second half-size linear system as it contains a block Toeplitz matrix. Both half-size BL3TB-like matrices could be further reduced the matrix size by half until the matrices with small enough size are reached. Finally, the solution of (1.1) is recursively obtained. The computational complexity of the proposed method is of $\mathcal{O}(MN \log^2 M)$ operations which is cheaper than $\mathcal{O}(M^2N)$ operations of the classical BFS method. We remark that our method is of $\mathcal{O}(MN)$ storage requirement and it can be applied to the general BL3TB-like system. Numerical examples are given to illustrate the efficiency of the proposed method.

The outline of this paper is given as follows. In Section 2, we present the proposed algorithm. In Section 3, we consider coefficient matrices constructed by the finite difference discretization of time-fractional partial differential equations. In Section 4, we report experimental results and compare the proposed method with the other testing methods. Finally, concluding remarks are given in Section 5.

2. The proposed method

Firstly, we briefly introduce the BFS method for solving the block lower triangular system. Let

$$\mathbf{u} := \begin{bmatrix} u^{(1)} \\ u^{(2)} \\ \vdots \\ u^{(M)} \end{bmatrix} \quad \text{and} \quad \mathbf{b} := \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(M)} \end{bmatrix},$$

where $u^{(i)}$ and $b^{(i)}$ $(i = 1, 2, \cdots, M)$ are vectors of size N. The BFS method solves $u^{(1)}, u^{(2)}, \cdots u^{(M)}$ one by one via solving the following linear systems:

$$\begin{aligned} A_1^{(1)} u^{(1)} &= b^{(1)}, \\ A_1^{(2)} u^{(2)} &= b^{(2)} - A_2 u^{(1)}, \\ &\vdots \\ A_1^{(M)} u^{(M)} &= b^{(M)} - \sum_{i=1}^{M-1} A_{i+1} u^{(M-i)}. \end{aligned} \tag{2.1}$$

Obviously, each tri-diagonal linear system in (2.1) can be directly solved in $\mathcal{O}(N)$ operations. Nevertheless, it requires to compute the right hand side vector and the computational cost is $\mathcal{O}(kMN)$ operations for the kth equation in (2.1). Therefore, the overall computational complexity is of $\mathcal{O}(M^2N)$ operations by the BFS method.

We note that the main workload of the BFS method is in the calculation of the right hand side. Therefore, the key is to find an efficient way to compute the right hand side faster. To this end, in the following, we exploit the divide-and-conquer strategy in the BFS method to reduce the complexity, see [21].

Without loss of generality, we assume that $M > 1$ is even. Let $k = M/2$. The matrix $\mathbf{A}$ can be partitioned as follows:

$$\mathbf{A} = \left[\begin{array}{ccc|ccc} A_1^{(1)} & & & & & \\ \vdots & \ddots & & & & \\ A_k & \cdots & A_1^{(k)} & & & \\ \hline A_{k+1} & \cdots & A_2 & A_1^{(k+1)} & & \\ \vdots & & \vdots & \vdots & \ddots & \\ A_M & \cdots & A_{k+1} & A_k & \cdots & A_1^{(M)} \end{array}\right] := \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} \end{bmatrix},$$

in which $\mathbf{0}$ is a zero matrix. Accordingly, $\mathbf{u}$ and $\mathbf{b}$ are partitioned into $\mathbf{u} =: \begin{bmatrix} \mathbf{v}^{(1)} \\ \mathbf{v}^{(2)} \end{bmatrix}$ and $\mathbf{b} =: \begin{bmatrix} \mathbf{p}^{(1)} \\ \mathbf{p}^{(2)} \end{bmatrix}$ respectively. Therefore, the linear system (1.1) is reduced into two half-size linear systems equivalently

$$\begin{cases} \mathbf{B}\mathbf{v}^{(1)} = \mathbf{p}^{(1)}, \\ \mathbf{D}\mathbf{v}^{(2)} = \mathbf{p}^{(2)} - \mathbf{C}\mathbf{v}^{(1)}. \end{cases} \tag{2.2}$$

We note that both $\mathbf{B}$ and $\mathbf{D}$ are BL3TB-like, while $\mathbf{C}$ is a block Toeplitz matrix. Suppose $\mathbf{v}^{(1)}$ is calculated. Since $\mathbf{C}$ is a block Toeplitz matrix, the matrix–vector product $\mathbf{C}\mathbf{v}^{(1)}$ can be computed efficiently by the FFT. More specifically, $\mathbf{C}$ is firstly extended into an MN-by-MN block circulant matrix $\tilde{\mathbf{C}}$ where its first block column is given by

$$\mathbf{S} = \begin{bmatrix} A_{k+1} \\ A_{k+2} \\ \vdots \\ A_M \\ 0 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}.$$

The block circulant matrix $\tilde{\mathbf{C}}$ can be diagonalized into diagonal block matrix by the discrete Fourier transform matrix, i.e.,

$$\tilde{\mathbf{C}} = (F^* \otimes I_N)\mathrm{diag}(\Lambda_1, \Lambda_2, \cdots, \Lambda_M)(F \otimes I_N),$$

in which F is the discrete Fourier transform matrix of size M, I_N is the N-by-N identity matrix, $\otimes$ is the Kronecker product and $\Lambda_1, \cdots, \Lambda_M$ are $N \times N$ matrices determined by

$$\begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_M \end{bmatrix} = \sqrt{M}(F \otimes I_N)\mathbf{S}; \tag{2.3}$$

see [9]. The computational cost of $\Lambda_1, \cdots, \Lambda_M$ is of $\mathcal{O}(NM \log M)$ operations. Moreover, since A_j are tri-diagonal matrices, so are Λ_j. It follows that the matrix–vector multiplication $\mathbf{C}\mathbf{v}^{(1)}$ can be computed via

$$\tilde{\mathbf{C}} \begin{bmatrix} \mathbf{v}^{(1)} \\ \mathbf{0} \end{bmatrix} = (F^* \otimes I_N)\mathrm{diag}(\Lambda_1, \Lambda_1, \cdots, \Lambda_M)(F \otimes I_N) \begin{bmatrix} \mathbf{v}^{(1)} \\ \mathbf{0} \end{bmatrix}.$$

The cost of this matrix–vector multiplication is of $\mathcal{O}(NM \log M)$ operations. In (2.2), suppose the first linear system $\mathbf{B}\mathbf{v}^{(1)} = \mathbf{p}^{(1)}$ is solved, then the second linear system $\mathbf{D}\mathbf{v}^{(2)} = \mathbf{p}^{(2)} - \mathbf{C}\mathbf{v}^{(1)}$ can be computed after the right hand side vector is obtained. Since both $\mathbf{B}$ and $\mathbf{D}$ are BL3TB-like matrices, the same procedure can be applied to solve both linear systems recursively. The proposed algorithm, we call that as the DC-BFS method, for solving BL3TB-like linear systems in (1.1) is summarized in Algorithm `DCBFS`.

Algorithm: `DCBFS` $(\{A_1^{(j)}\}_{j=1}^m, \{A_j\}_{j=2}^m, \mathbf{b}, \mathbf{u})$
Input: $\{A_1^{(j)}\}_{j=1}^m$, $\{A_j\}_{j=2}^m$, $\mathbf{b}$
Output: $\mathbf{u}$
Step 1: If $m = 1$, then solve $A_1^{(1)}\mathbf{u} = \mathbf{b}$; Otherwise
Step 2: Call `DCBFS`$(\{A_1^{(j)}\}_{j=1}^{m/2}, \{A_j\}_{j=2}^{m/2}, \mathbf{b}[1:m/2], \mathbf{u}[1:m/2])$
Step 3: Compute $\mathbf{b}' = \mathbf{b}[m/2+1:m] - \texttt{BlockToep}(\{A_j\}_{j=2}^m, \mathbf{u}[1:m/2])$
Step 4: Call `DCBFS`$(\{A_1^{(j)}\}_{j=m/2+1}^{m}, \{A_j\}_{j=2}^{m/2}, \mathbf{b}', \mathbf{u}[m/2+1:m])$

In the above algorithm, $\mathbf{x}[1:m/2]$ refers to the first half of the partition of $\mathbf{x}$, $\mathbf{x}[m/2+1:m]$ refers to the remaining half of the partition of $\mathbf{x}$, and $\texttt{BlockToep}(\{A_j\}_{j=2}^m, \mathbf{u}[1:m/2])$ refers to the fast block Toeplitz matrix–vector multiplication using FFT. When m, the number of block, is greater than one, we employ the divide-and-conquer strategy to reduce that into the two smaller block systems with size by half. To study the total complexity, we suppose that Θ_m is the number of operations required for solving an m-block linear system. Its computational cost can be estimated as follows:

$$\Theta_m = \mathcal{O}(Nm \log m) + 2\Theta_{m/2}.$$

When the number of block is equal to one, we only need to solve a tri-diagonal linear system and the cost is of $\mathcal{O}(N)$ operations, i.e., $\Theta_1 = \mathcal{O}(N)$. According to this recursive formula, we obtain $\Theta_M = \mathcal{O}(NM \log^2 M)$ for the M-by-M block lower triangular Toeplitz-like with N-by-N tri-diagonal block system in (1.1). It is clear it is significantly less than the cost of BFS method which is of $\mathcal{O}(NM^2)$ operations.

We remark that the proposed method can be applied for the general BL3TB-like linear system (1.1); i.e., the diagonal blocks are not necessarily the same.

3. Applications to time-fractional partial differential equations

In this section, we consider linear systems in (1.1) arising from the discretization of time-fractional partial differential equations.

3.1. The fractional sub-diffusion equation

Consider the following nonlinear fractional sub-diffusion equations [14,16,22]

$$\begin{aligned} &\frac{\partial u(x,t)}{\partial t} = {}_0\mathcal{D}_t^{1-\gamma}\left[K\frac{\partial^2 u(x,t)}{\partial x^2}\right] + f(x,t),\ x \in (a,b),\ t \in (0,T], \\ &u(a,t) = \psi_1(t),\ u(b,t) = \psi_2(t),\ t \in (0,T], \\ &u(x,0) = \phi(x),\ x \in [a,b], \end{aligned} \tag{3.1}$$

where $0 \le \gamma \le 1$ and ${}_0\mathcal{D}_t^{1-\gamma}$ denotes the Riemann–Liouville fractional derivative of order $1-\gamma$ defined by

$${}_0\mathcal{D}_t^{1-\gamma} f(t) = \frac{1}{\Gamma(\gamma)}\frac{\partial}{\partial t}\int_0^t \frac{f(\tau)}{(t-\tau)^{1-\gamma}}d\tau,$$

K is the diffusion coefficient, $\phi(x)$, $\psi_1(t)$, $\psi_2(t)$ and $f(x,t)$ are known smooth functions. We note that the diffusion coefficient K here can be either constant or time-dependent. The latter case has received much attention in recent years, see [24,31,17,6] for example.

For the fractional sub-diffusion equations in (3.1), linear systems in (1.1) arise from the discretization by finite difference schemes. As an example, Gao and Sun [16] derived a compact finite difference scheme, which has $2-\gamma$ order and fourth-order accuracy in temporal and spatial variables respectively.

Let $\Delta x = (b-a)/(N+1)$ and $\Delta t = T/M$ with mesh points $(x_i, t_k) = (a + i\Delta x, k\Delta t)$ for $i = 0, 1, \cdots, N+1$ and $t = 0, 1, \cdots, M$, and let the approximate solution be $u_i^k \approx u(x_i, t_k)$. The Gao–Sun scheme for the equation (3.1) can be written as follows:

$$\begin{aligned}
&\alpha_1^{(k)} u_{i-1}^k + \beta_1^{(k)} u_i^k + \eta_1^{(k)} u_{i+1}^k - \sum_{j=1}^{k-1} (\alpha_{k-j+1} u_{i-1}^j + \beta_{k-j+1} u_i^j + \eta_{k-j+1} u_{i+1}^j) \\
&\quad = \tilde{b}_i^k, \ 1 \le i \le N, \ 1 \le k \le M, \\
&u_0^k = \psi_1(t_k), \ u_{M+1}^k = \psi_2(t_k), \ 1 \le k \le M, \\
&u_i^0 = \phi(x_i), 0 \le i \le N+1,
\end{aligned} \tag{3.2}$$

where

$$\begin{cases}
\alpha_1^{(i)} = \eta_1^{(i)} = \frac{1}{12} - K(t_i) \frac{\Delta t^{\gamma} \Gamma(2-\gamma)}{\Delta x^2}, \\
\beta_1^{(i)} = \frac{5}{6} + 2K(t_i) \frac{\Delta t^{\gamma} \Gamma(2-\gamma)}{\Delta x^2}, \\
\alpha_j = \eta_j = \frac{1}{12}(a_{j-2} - a_{j-1}), \\
\beta_j = \frac{5}{6}(a_{j-2} - a_{j-1}), \\
a_{j-2} = (j-1)^{(1-\gamma)} - (j-2)^{(1-\gamma)},
\end{cases}$$

for $1 \le i \le M$, $2 \le j \le M$ and $\tilde{b}_i^k$ contains the nonhomogeneous term and the initial condition. More details of the discretization could be found in [16,22].

The equations (3.2) can be written as the following matrix form:

$$A_1^{(k)} u^k - \sum_{j=1}^{k-1} A_{k-j+1} u^j = b^k, \ k = 1, 2, \cdots, M, \tag{3.3}$$

where

$$A_1^{(k)} = \begin{bmatrix}
\beta_1^{(k)} & \eta_1^{(k)} & & & \\
\alpha_1^{(k)} & \beta_1^{(k)} & \eta_1^{(k)} & & \\
 & \ddots & \ddots & \ddots & \\
 & & \alpha_1^{(k)} & \beta_1^{(k)} & \eta_1^{(k)} \\
 & & & \alpha_1^{(k)} & \beta_1^{(k)}
\end{bmatrix}, \ A_j = \begin{bmatrix}
\beta_j & \eta_j & & & \\
\alpha_j & \beta_j & \eta_j & & \\
 & \ddots & \ddots & \ddots & \\
 & & \alpha_j & \beta_j & \eta_j \\
 & & & \alpha_j & \beta_j
\end{bmatrix}$$

for $k = 1, 2, \cdots, M$, $j = 2, 3, \cdots, M$, $u^k = [u_1^k, u_2^k, \cdots, u_N^k]^T$ and

$$b^k = [\tilde{b}_1^k - \sum_{i=1}^{k} \alpha^{(k-i)} u_0^i, \tilde{b}_2^k, \cdots, \tilde{b}_{N-1}^k, \tilde{b}_N^k - \sum_{i=1}^{k} \eta^{(k-i)} u_{N+1}^i]^T, \ 1 \le k \le M.$$

It is easy to see that (3.3) as a linear system in $\mathbf{u} = [u_1^1, u_2^1, \cdots, u_N^1, u_1^2, u_2^2, \cdots, u_N^M]^T$ possesses block lower triangular Toeplitz-like with tri-diagonal block structure. The proposed method can be applied to solve it at a cost of $\mathcal{O}(NM\log^2 M)$ operations.

3.2. The two-dimensional fractional sub-diffusion equation

Now we consider the two-dimensional fractional sub-diffusion equation

$$\begin{aligned}
&\frac{\partial^{\alpha} u(x,y,t)}{\partial t^{\alpha}} = \Delta u(x,y,t) + f(x,y,t), \ (x,y) \in \Omega, t \in (0,T], \\
&u(x,y,t) = \psi(x,y,t), \ (x,y) \in \partial\Omega, \ \ t \in (0,T], \\
&u(x,y,0) = \phi(x,y), \ (x,y) \in \Omega \cup \partial\Omega,
\end{aligned} \tag{3.4}$$

where $\Omega = (0, R_x) \times (0, R_y)$ is a finite rectangular domain, $\partial\Omega$ is the boundary, Δ is the Laplacian and $\frac{\partial^\alpha u(x,y,t)}{\partial t^\alpha}$ is the Caputo derivative of order α defined by

$$\frac{\partial^\alpha u(x,y,t)}{\partial t^\alpha} = \begin{cases} \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{\partial u(x,y,\eta)}{\partial \eta} \frac{d\eta}{(t-\eta)^\alpha}, & 0 < \alpha < 1, \\ \frac{\partial u(x,y,t)}{\partial t}, & \alpha = 1. \end{cases} \tag{3.5}$$

Many numerical approaches have been designed to solve the two-dimensional fractional sub-diffusion equation. Among them, we are particularly interested in the alternating direction implicit (ADI) schemes, which are proved to have unconditional stability and H^1 norm convergence. The ADI schemes treat a multi-dimensional problem as several independent one-dimensional problems separately and therefore have the advantage of low computational complexities.

Let $\Delta_x = R_x/(N_1 + 1)$, $\Delta_y = R_y/(N_2 + 1)$, $\Delta_t = T/(M + 1)$ be the spatial and time steps with mesh point $(x_i, y_j, t_k) = (i\Delta_x, j\Delta_y, k\Delta_t)$, for $i = 0, 1, \cdots, N_1$, $j = 0, 1, \cdots, N_2$, $k = 0, 1, \cdots, M$. The ADI scheme given by Zhang and Sun [36] for the two-dimensional fractional sub-diffusion equation has the following matrix form

$$[(I - T_1) \otimes (I - T_2)]u^{(m)} = \sum_{k=1}^{m-1} (a_{n-k-1} - a_{n-k})(u^{(k)} + (T_1 \otimes T_2)u^{(k)}) + b^{(m)}, \; m = 1, 2, \cdots, M, \tag{3.6}$$

where $a_j = (j+1)^{1-\alpha} - j^{1-\alpha}$, $b^{(m)}$ is a N_1N_2-vector containing the initial and boundary conditions, T_1 and T_2 are $N_1 \times N_1$ and $N_2 \times N_2$ tri-diagonal matrices of the form

$$\Delta_t^\alpha \Gamma(2-\alpha) \begin{bmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}.$$

The $u^{(m)}$ in (3.6) is an approximation of u at time point m, i.e.,

$$u^{(m)} \approx \begin{bmatrix} (u(x_1, y_1, t_m), u(x_1, y_2, t_m), \cdots, u(x_1, y_{N_2}, t_m))^T \\ \vdots \\ (u(x_{N_1}, y_1, t_m), u(x_{N_1}, y_2, t_m), \cdots, u(x_{N_1}, y_{N_2}, t_m))^T \end{bmatrix}.$$

The linear system (3.6) can be solved by the time-marching method. Moreover, as $(I - T_1) \otimes (I - T_2) = ((I - T_1) \otimes I)(I \otimes (I - T_2))$, for $m = 1, 2, \cdots, M$, the linear equations (3.6) (at time point t_m) can be solved with a two-step strategy:

$$\begin{cases} [(I - T_1) \otimes I]v^{(m)} = \sum_{k=1}^{m-1} (a_{n-k-1} - a_{n-k})(u^{(k)} + (T_1 \otimes T_2)u^{(k)}) + b^{(m)}, \\ [I \otimes (I - T_2)]u^{(m)} = v^{(m)}. \end{cases} \tag{3.7}$$

Both equations of (3.7) are tri-diagonal linear systems of size N_1N_2, which can be solved efficiently with $\mathcal{O}(N_1N_2)$ operations, provided that the vectors on the right hand side are computed in advance. However, it needs an average of $\mathcal{O}(N_1N_2M)$ operations to obtain a RHS vector and the time-marching method for the system (3.6) requires an overall computational complexity of $\mathcal{O}(N_1N_2M^2)$. By making use of block Toeplitz structure in (3.6), the proposed method can be employed to solve the equations with a lower cost at $\mathcal{O}(N_1N_2M\log^2 M)$.

We remark that the approximate inversion method [22] fails to deal with this two-dimensional fractional sub-diffusion equations efficiently. Even the coefficient matrix can be diagonalized into a block diagonal matrix with $M \times M$ blocks by using fast Fourier transforms, the structure of diagonal blocks are very complicated. Here it is no longer tensor product structure of two tri-diagonal matrices. Therefore, the two-step strategy cannot be applied to obtain the solution efficiently.

4. Numerical examples

In this section, we give three examples to illustrate the performance of the proposed method (DC-BFS) for the one-dimensional and two-dimensional fractional sub-diffusion equations. All numerical experiments are carried out on an Intel(R) Core(TM)2 2.66 GHz quad processor machine with 4 GB RAM.

Example 1. (See [16,22].) We consider the equation (3.1) with $a = 0$, $b = 1$, $T = 1$, $\gamma = 3/4$, $K = 1$, and

$$f(x,t) = \exp(x)\left[(1+\gamma)t^\gamma - \frac{\Gamma(2+\gamma)}{\Gamma(1+2\gamma)}t^{2\gamma}\right].$$

The initial condition is given by $\psi(x) = 0$ and the boundary conditions are $\phi_1(t) = t^{1+\gamma}$, $\phi_2(t) = et^{1+\gamma}$. For this equation, we have the exact solution $u(x,t) = e^x t^{1+\gamma}$. Fixing the spatial grid number $N = 256$ and setting the temporal time step $M = 2^7, 2^8, \cdots, 2^{16}$, we solve the equation by (3.2) with the block forward substitution method (BFS) or the time-marching

Table 1
BFS method, DC-BFS method and AI method for Example 1 ($N = 256$).

M	BFS method		DC-BFS method		AI method	
	Error	CPU	Error	CPU	Error	CPU
2^7	1.273e−04	0.227	1.273e−04	0.089	1.273e−04	0.034
2^8	5.422e−05	0.788	5.422e−05	0.190	5.422e−05	0.068
2^9	2.297e−05	2.896	2.297e−05	0.409	2.297e−05	0.139
2^{10}	9.700e−06	11.180	9.700e−06	0.903	9.702e−06	0.280
2^{11}	4.089e−06	44.633	4.089e−06	1.989	4.091e−06	0.566
2^{12}	1.722e−06	176.940	1.722e−06	4.270	1.724e−06	1.148
2^{13}	7.246e−07	705.074	7.246e−07	9.289	7.264e−07	2.259
2^{14}	–	>2000	3.048e−07	20.068	3.067e−07	4.536
2^{15}	–	>2000	1.282e−07	43.390	2.515e−07	9.111
2^{16}	–	>2000	5.391e−08	93.408	7.988e−08	18.312

Table 2
BFS method and DC-BFS method for Example 2 ($N = 256$).

M	BFS method	DC-BFS method	BFS/DC-BFS method
	CPU	CPU	Error
2^{10}	11.299	0.905	9.678e−06
2^{11}	44.531	2.002	4.080e−06
2^{12}	176.801	4.342	1.718e−06
2^{13}	713.923	9.424	7.230e−07
2^{14}	2848.940	20.207	3.042e−07
2^{15}	>2 hours	43.391	1.279e−07
2^{16}	>2 hours	93.483	5.380e−08

method, the proposed method (DC-BFS) and the approximate inversion method (AI method) [22] respectively. The accuracy and the CPU time of these three methods are compared in Table 1, in which 'Error' denotes the relative maximum error between the exact solution and the numerical solution defined by

$$\frac{\max_{1\le i\le N, 1\le k\le M} |u_i^k - u(x_i, t_k)|}{\max_{1\le i\le N, 1\le k\le M} |u(x_i, t_k)|},$$

and 'CPU' represents the CPU time in seconds for solving the corresponding discretized system.

In the table, we can see that the DC-BFS method and the BFS method have the same accuracy, while the DC-BFS method has a higher efficiency. When the spatial grid number is equal or greater than 2^{14}, it takes the BFS method more than 2000 seconds to solve the problem, while the DC-BFS method can be finished in about one minute. The CPU time of the AI method is the least among the three methods, but it is slightly less accurate especially when M is large.

Example 2. In this example, we consider the equation (3.1) with a time-varying diffusion coefficient $K(t) = 1 + t^2$, $a = 0$, $b = 1$, $T = 1$, $\gamma = 3/4$, the source term

$$f(x,t) = \exp(x)\left[(1+\gamma)t^{\gamma} - \frac{\Gamma(2+\gamma)}{\Gamma(1+2\gamma)}t^{2\gamma} - \frac{\Gamma(4+\gamma)}{\Gamma(3+2\gamma)}t^{2\gamma+2}\right],$$

the boundary conditions $\phi_1(t) = t^{1+\gamma}$, $\phi_2(t) = et^{1+\gamma}$ and initial condition $\psi(x) = 0$. The exact solution of this equation is $u(x,t) = e^x t^{1+\gamma}$. The discretized system (3.2) by Gao–Sun's scheme can be solved by the time-marching method as well as the proposed method. However, the approximate inversion method [22] cannot be applied to solve this equation with time-variant coefficients, as the discretized linear system does not have a purely block-Toeplitz structure. Here we only test the time-marching method and our proposed method DC-BFS. We fix the spatial grid number $N = 256$ and set the temporal time step $M = 2^7, 2^8, \cdots, 2^{16}$ respectively. Again, we compare the relative maximum error and the CPU time in seconds of the two methods in a table (see Table 2). We terminate the time-marching method when the computational time is more than two hours.

According to Table 2, the DC-BFS method solves the fractional sub-diffusion equation as precisely as the BFS method does. Nevertheless, due to the divide and conquer strategy, the DC-BFS method has a great improvement in efficiency. It takes more than 2 hours for the BFS method to solve the fractional sub-diffusion equation with 2^{15} time steps, but the DC-BFS method requires less than one minute.

Example 3. (See [36].) We consider the two dimensional fractional sub-diffusion problem (3.4) with the exact solution $u(x, y, t) = \sin(x)\sin(y)t^2$ in the domain $\Omega \times [0, 1/2]$ where $\Omega = [0, \pi] \times [0, \pi]$. The order of fractional derivative $\alpha = 3/4$, the forcing term is

$$f(x, y, t) = \sin(x)\sin(y)(\frac{2}{\Gamma(3-\alpha)} + 2t^2),$$

Table 3
BFS method and DC-BFS method for Example 3 ($N_1 = N_2 = 64$).

M	BFS method	DC-BFS method	BFS/DC-BFS method
	CPU	CPU	Error
2^8	11.752	5.399	4.081e−04
2^9	38.655	11.289	2.159e−04
2^{10}	137.852	23.630	1.331e−04
2^{11}	516.810	49.617	9.750e−05
2^{12}	2027.260	103.888	8.230e−05
2^{13}	>2 hours	215.093	7.581e−05

and the boundary and initial conditions are

$$\psi(x, y, t) = u(x, y, t), \ (x, y) \in \partial\Omega,$$

$$\phi(x, y) = 0, \ (x, y) \in \Omega.$$

We test how well the BFS method and the DC-BFS method work on this time-fractional equation. If we employ the AI method, the block diagonalization of the block ϵ-circulant matrix with the diagonal blocks with the off-diagonal blocks, and hence the diagonal blocks of the resulting matrix are no longer tensor products of two tri-diagonal matrices. In this case, the AI method cannot be used to solve the system efficiently. Therefore, the AI method is not used in this experiment.

Table 3 shows the relative maximum error (Error) and the CPU time (CPU) of these methods for $N_1 = 64$, $N_2 = 64$ and $M = 2^5, 2^6, \cdots, 2^{10}$ respectively. According to our tests, the numerical solutions of (3.4) by the two methods have the same accuracy. The DC-BFS method, however, shows great advantage when M is large, due to the $\log^2 M$ factor in its computational complexity.

5. Concluding remarks

In this paper, we have developed a fast solver for block lower triangular Toeplitz-like systems. The proposed method takes advantage of the block-Toeplitz structure in a recursive manner to solve the system efficiently. We show that the computational complexity of the proposed method is much lower at $\mathcal{O}(NM\log^2 M)$, compared to $\mathcal{O}(NM^2)$ for the time-marching method. In applications, we use the proposed method to numerically solve the fractional time-dependent partial differential equations. The numerical examples have shown that the cost of the proposed method is cheaper than the time-marching method in terms of CPU time.

References

[1] J. Bai, X. Feng, Fractional-order anisotropic diffusion for image denoising, IEEE Trans. Image Process. 16 (2007) 2492–2502.
[2] D. Benson, S. Wheatcraft, M. Meerschaert, Application of a fractional advection–dispersion equation, Water Resour. Res. 36 (2000) 1403–1412.
[3] D. Benson, S. Wheatcraft, M. Meerschaert, The fractional-order governing equation of Lévy motion, Water Resour. Res. 36 (2000) 1413–1423.
[4] D. Bini, G. Latouche, B. Meini, Numerical Methods for Structured Markov Chains, Oxford University Press, New York, 2005.
[5] D. Bini, V. Pan, Polynomial division and its computational complexity, J. Complex. 2 (1986) 179–203.
[6] M. Bologna, A. Svenkeson, B.J. West, P. Grigolini, Diffusion in heterogeneous media: an iterative scheme for finding approximate solutions to fractional differential equations with time-dependent coefficients, J. Comput. Phys. (2014).
[7] B. Carreras, V. Lynch, G. Zaslavsky, Anomalous diffusion and exit time distribution of particle tracers in plasma turbulence models, Phys. Plasmas 8 (2001) 5096–5103.
[8] R. Chan, X. Jin, An Introduction to Iterative Toeplitz Solvers, SIAM, Philadelphia, 2007.
[9] R. Chan, M. Ng, Conjugate gradient methods for Toeplitz systems, SIAM Rev. 38 (1996) 427–482.
[10] C. Chen, F. Liu, V. Anh, I. Turner, Numerical schemes with high spatial accuracy for a variable-order anomalous subdiffusion equation, SIAM J. Sci. Comput. 32 (2010) 1740–1760.
[11] C. Chen, F. Liu, I. Turner, V. Anh, A Fourier method for the fractional diffusion equation describing sub-diffusion, J. Comput. Phys. 227 (2007) 886–897.
[12] C. Chen, F. Liu, I. Turner, V. Anh, Y. Chen, Numerical approximation for a variable-order nonlinear reaction-subdiffusion equation, Numer. Algorithms 63 (2013) 265–290.
[13] S. Chen, F. Liu, X. Jiang, I. Turner, V. Anh, A fast semi-implicit difference method for a nonlinear two-sided space-fractional diffusion equation with variable diffusivity coefficients, Appl. Math. Comput. 257 (2015) 591–601.
[14] M. Cui, Compact finite difference method for the fractional diffusion equation, J. Comput. Phys. 228 (2009) 7792–7804.
[15] D. Commenges, M. Monsion, Fast inversion of triangular Toeplitz matrices, IEEE Trans. Autom. Control 29 (1984) 250–251.
[16] G. Gao, Z. Sun, A compact finite difference scheme for the fractional sub-diffusion equations, J. Comput. Phys. 230 (2011) 586–595.
[17] R. Garra, E. Orsingher, F. Polito, Fractional diffusions with time-varying coefficients, arXiv preprint, arXiv:1501.04806, 2015.
[18] G. Golub, C. Loan, Matrix Computations, 3rd edition, Johns Hopkins University Press, Baltimore, 1996.
[19] T. Kailath, S. Kung, M. Morf, Displacement ranks of matrices and linear equations, J. Math. Anal. Appl. 68 (1979) 395–407.
[20] T. Langlands, B. Henry, The accuracy and stability of an implicit solution method for the fractional diffusion equation, J. Comput. Phys. 205 (2005) 719–736.
[21] F. Lin, W. Ching, M. Ng, Fast inversion of triangular Toeplitz matrices, Theor. Comput. Sci. 315 (2004) 511–523.
[22] X. Lu, H. Pang, H. Sun, Fast approximate inversion of block triangular Toeplitz matrices with application to sub-diffusion equations, Numer. Linear Algebra Appl. (2015), http://dx.doi.org/10.1002/nla.1972.
[23] R. Magin, Fractional Calculus in Bioengineering, Begell House Publishers, 2006.
[24] E. Orsingher, F. Polito, Some results on time-varying fractional partial differential equations and birth-death processes, in: Proceedings of the XIII International EM Conference on Eventological Mathematics and Related Fields, Krasnoyarsk, 2009, pp. 23–27.

[25] V. Pan, Structured Matrices and Polynomials: Unified Superfast Algorithms, Springer, New York, 2001.
[26] I. Podlubny, Fractional Differential Equations, Academic Press, New York, 1999.
[27] M. Raberto, E. Scalas, F. Mainardi, Waiting-times and returns in high-frequency financial data: an empirical study, Physica A 314 (2002) 749–755.
[28] S. Shen, F. Liu, Q. Liu, V. Anh, Numerical simulation of anomalous infiltration in porous media, Numer. Algorithms 68 (2015) 443–454.
[29] M. Shlesinger, B. West, J. Klafter, Lévy dynamics of enhanced diffusion: application to turbulence, Phys. Rev. Lett. 58 (1987) 1100–1103.
[30] I. Sokolov, J. Klafter, A. Blumen, Fractional kinetics, Phys. Today 55 (2002) 48–54.
[31] J. Wu, K.M. Berland, Propagators and time-dependent diffusion coefficients for anomalous diffusion, Biophys. J. 95 (2008) 2049–2052.
[32] G. Zaslavsky, D. Stevens, H. Weitzner, Self-similar transport in incomplete chaos, Phys. Rev. E 48 (1993) 1683–1694.
[33] F. Zeng, C. Li, F. Liu, High-order explicit–implicit numerical methods for nonlinear anomalous diffusion equations, Eur. Phys. J. Spec. Top. 222 (2013) 1885–1900.
[34] F. Zeng, C. Li, F. Liu, I. Turner, The use of finite difference/element approximations for solving the time-fractional subdiffusion equation, SIAM J. Sci. Comput. 35 (2013) 2976–3000.
[35] F. Zeng, C. Li, F. Liu, I. Turner, Numerical algorithms for time-fractional subdiffusion equation with second-order accuracy, SIAM J. Sci. Comput. 37 (2015) 55–78.
[36] Y. Zhang, Z. Sun, Alternating direction implicit schemes for the two-dimensional fractional sub-diffusion equation, J. Comput. Phys. 230 (2011) 8713–8728.
[37] Y. Zhang, Z. Sun, H. Wu, Error estimates of Crank–Nicolson-type difference schemes for the sub-diffusion equation, SIAM J. Numer. Anal. 49 (2011) 2302–2322.
[38] M. Zheng, F. Liu, I. Turner, V. Anh, A novel high order space–time spectral method for the time-fractional Fokker–Planck equation, SIAM J. Sci. Comput. 37 (2015) 701–724.
[39] P. Zhuang, F. Liu, V. Anh, I. Turner, New solution and analytical techniques of the implicit numerical method for the anomalous subdiffusion equation, SIAM J. Numer. Anal. 46 (2008) 1079–1095.