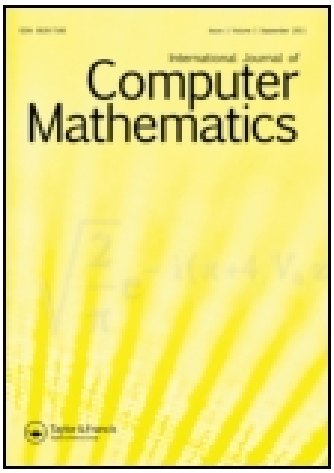

International Journal of Computer Mathematics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gcom20

Circulant and skew-circulant splitting iteration for fractional advection-diffusion equations

Wei Qu[a], Siu-Long Lei[a] & Seak-Weng Vong[a]
[a] Department of Mathematics, University of Macau, Macao, China
Accepted author version posted online: 17 Jan 2014.Published online: 26 Mar 2014.

International Journal of Computer Mathematics, 2014
Vol. 91, No. 10, 2232–2242, http://dx.doi.org/10.1080/00207160.2013.871001

Circulant and skew-circulant splitting iteration for fractional advection–diffusion equations

Wei Qu, Siu-Long Lei* and Seak-Weng Vong

Department of Mathematics, University of Macau, Macao, China

(*Received 12 June 2013; revised version received 20 August 2013; second revision received 15 November 2013; accepted 23 November 2013*)

An implicit second-order finite difference scheme, which is unconditionally stable, is employed to discretize fractional advection–diffusion equations with constant coefficients. The resulting systems are full, unsymmetric, and possess Toeplitz structure. Circulant and skew-circulant splitting iteration is employed for solving the Toeplitz system. The method is proved to be convergent unconditionally to the solution of the linear system. Numerical examples show that the convergence rate of the method is fast.

Keywords: Toeplitz matrix; fractional advection–diffusion equation; fast Fourier transform; circulant and skew-circulant splitting iteration

2010 AMS Subject Classifications: 26A33; 65L12; 65L20; 65F10; 65T50

1. Introduction

In this section, we derive the discretized linear system from fractional advection–diffusion equations (FADEs) with constant coefficients and introduce the background of circulant and skew-circulant splitting (CSCS) iteration for solving the discretized linear system. Let us consider the problem

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= v\frac{\partial u(x,t)}{\partial x} + d_{+}{}_{a}D_{x}^{\alpha}u(x,t) + d_{-}{}_{x}D_{b}^{\alpha}u(x,t) + f(x,t), \\ u(a,t) &= u(b,t) = 0, \quad t \in [0,T], \\ u(x,0) &= u_0(x), \quad x \in [a,b], \end{aligned} \tag{1}$$

where v is a real number, and $d_{\pm} \geq 0$ with $d_{+} + d_{-} \neq 0$. Here ${}_{a}D_{x}^{\alpha}u(x,t)$ and ${}_{x}D_{b}^{\alpha}u(x,t)$ denote the α-order left and right Riemann–Liouville fractional derivatives of $u(x,t)$, respectively, and they are defined as

$${}_{a}D_{x}^{\alpha}u(x,t) = \frac{1}{\Gamma(2-\alpha)}\frac{\partial^2}{\partial x^2}\int_{a}^{x}\frac{u(\xi,t)}{(x-\xi)^{\alpha-1}}\,\mathrm{d}\xi,$$

*Corresponding author. Email: sllei@umac.mo

$$ {}_xD_b^{\alpha}u(x,t)=\frac{1}{\Gamma(2-\alpha)}\frac{\partial^2}{\partial x^2}\int_x^b\frac{u(\xi,t)}{(\xi-x)^{\alpha-1}}\,\mathrm{d}\xi, $$

where $\Gamma(\cdot)$ is the gamma function, and $\alpha\in(1,2)$.

Let N,M be positive integers, and $h=(b-a)/(N+1)$ and $\tau=T/M$ be the space step and time step, respectively. We define a spatial and temporal partition $x_i=a+ih$ for $i=0,1,\ldots,N+1$, $t_m=m\tau$ for $m=0,1,\ldots,M$. Sousa and Li [18] proved that

$$ \begin{aligned} {}_aD_x^{\alpha}u(x_i,t_m)&=\frac{1}{\Gamma(4-\alpha)h^{\alpha}}\sum_{k=0}^{i+1}q_k^{(\alpha)}u(x_{i-k+1},t_m)+\mathcal{O}(h^2),\\ {}_xD_b^{\alpha}u(x_i,t_m)&=\frac{1}{\Gamma(4-\alpha)h^{\alpha}}\sum_{k=0}^{N-i+2}q_k^{(\alpha)}u(x_{i+k-1},t_m)+\mathcal{O}(h^2), \end{aligned} \tag{2} $$

where

$$ q_k^{(\alpha)}=\begin{cases}1, & k=0,\\ -4+2^{3-\alpha}, & k=1,\\ 3^{3-\alpha}-4\times2^{3-\alpha}+6, & k=2,\\ (k+1)^{3-\alpha}-4k^{3-\alpha}+6(k-1)^{3-\alpha}-4(k-2)^{3-\alpha}+(k-3)^{3-\alpha}, & k\ge3.\end{cases} \tag{3} $$

Let

$$ u_i^m\approx u(x_i,t_m),\quad f_i^{m+1/2}=f(x_i,t_{m+1/2}),\quad t_{m+1/2}=\tfrac{1}{2}(t_m+t_{m+1}), $$

for $i=0,1,\ldots,N+1$ and $m=0,1,\ldots,M$. Applying the Crank-Nicolson technique to the FADEs (1) with approximations (2) and the second-order central difference approximation to $\partial u/\partial x$, we have

$$ \begin{aligned} \frac{u_i^{m+1}-u_i^m}{\tau}&=\frac{\nu}{4}\left(\frac{u_{i+1}^{m+1}-u_{i-1}^{m+1}}{h}+\frac{u_{i+1}^m-u_{i-1}^m}{h}\right)\\ &\quad+\frac{1}{2\Gamma(4-\alpha)}\left(\frac{d_+}{h^{\alpha}}\sum_{k=0}^{i+1}q_k^{(\alpha)}u_{i-k+1}^m+\frac{d_+}{h^{\alpha}}\sum_{k=0}^{i+1}q_k^{(\alpha)}u_{i-k+1}^{m+1}\right.\\ &\quad\left.+\frac{d_-}{h^{\alpha}}\sum_{k=0}^{N-i+2}q_k^{(\alpha)}u_{i+k-1}^m+\frac{d_-}{h^{\alpha}}\sum_{k=0}^{N-i+2}q_k^{(\alpha)}u_{i+k-1}^{m+1}\right)+f_i^{m+1/2}, \end{aligned} \tag{4} $$

for $i=1,2,\ldots,N$. The finite difference scheme (4) is second-order accurate in both space and time, and is unconditionally stable [7]. Denote $\nu_{\tau,h,\alpha}=\tau/2\Gamma(4-\alpha)h^{\alpha}$, $\mu_{\tau,h}=\tau/4h$, $u^m=[u_1^m,u_2^m,\ldots,u_N^m]^{\mathrm{T}}$, and $f^{m+1/2}=[f_1^{m+1/2},f_2^{m+1/2},\ldots,f_N^{m+1/2}]^{\mathrm{T}}$. The scheme (4) is expressed in the matrix form

$$ Au^{m+1}=b^m, \tag{5} $$

where the coefficient matrix, see [21],

$$ A=I-\nu\mu_{\tau,h}W-\nu_{\tau,h,\alpha}(d_+Q+d_-Q^{\mathrm{T}}), \tag{6} $$

and the right-hand side vector

$$ b^m=\left(I+\nu\mu_{\tau,h}W+\nu_{\tau,h,\alpha}(d_+Q+d_-Q^{\mathrm{T}})\right)u^m+\tau f^{m+1/2}. \tag{7} $$

Here the $N \times N$ matrices

$$Q = \begin{bmatrix} q_1^{(\alpha)} & q_0^{(\alpha)} & 0 & \cdots & 0 \\ q_2^{(\alpha)} & q_1^{(\alpha)} & q_0^{(\alpha)} & \ddots & \vdots \\ \vdots & q_2^{(\alpha)} & q_1^{(\alpha)} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & q_0^{(\alpha)} \\ q_N^{(\alpha)} & \cdots & \cdots & q_2^{(\alpha)} & q_1^{(\alpha)} \end{bmatrix}, \quad W = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix}, \tag{8}$$

and I is the identity matrix. Note that the matrix Q is Toeplitz (see Definition 2 in next section) and the coefficient matrix A of Equation (5), defined in Equation (6), is a full Toeplitz matrix, which is unsymmetric when $d_+ \neq d_-$. Besides, A is independent of m, i.e. the coefficient matrix of Equation (5) is unchanged in each time level of the finite difference scheme.

FADE emerges from numerous research areas such as modelling chaotic dynamics of classical conservative systems [22], groundwater contaminant transport [2,3], turbulent flow [4,15], and, applications in biology [9], finance [13], image processing [1], and physics [16].

Recently, fast solution methods are developed for solving FADEs. In [21], Toeplitz structure of the coefficient matrix for finite difference scheme is firstly observed, and fast direct solver based on fast Fourier transform (FFT) is developed. In [20], Krylov subspace method with fast matrix-vector multiplication is proposed for solving two dimensional FADEs. Multigrid method [11] and conjugate gradient method on the normal equation with circulant preconditioner [8] are also efficient for solving FADEs. In this paper, an alternative choice is provided for solving FADEs by using CSCS iteration.

CSCS iteration for solving Toeplitz systems was proposed by Ng [10] in 2003. In each iteration of the method, six pairs of FFT and inverse FFT are required to be performed which cost $\mathcal{O}(N \log N)$ operations, where N is the matrix size. Since the implementation of FFT can be highly parallelized, CSCS iteration can be benefitted from parallel computing. Recently, CSCS iteration is employed for solving Toeplitz systems of weakly nonlinear equations [23].

The drawback of CSCS iteration is on the convergence. For general Toeplitz systems, CSCS iteration may not converge. A sufficient condition given in [10] for the convergence of CSCS iteration is that the two splitting matrices are positive stable. The definition of a matrix being positive (negative) stable is given as:

DEFINITION 1 *A matrix is said to be positive* (*negative*) *stable if all its eigenvalues have positive* (*negative*) *real parts.*

The sufficient condition is too strict to be satisfied for general Toeplitz systems. However, for Toeplitz systems (5), which come from finite difference discretization of FADE (1), the two splitting matrices are proved to be positive stable in this paper, and hence the CSCS iteration for solving Equation (5) converges for all initial guess and all $\alpha \in (1, 2)$, where α is the order of fractional derivatives. Numerical examples also show the fast convergence rate of the CSCS iteration.

The paper is organized as follows. In Section 2, CSCS iteration is introduced for solving the Toeplitz system (5). In Section 3, the convergence of the CSCS iteration is proved theoretically. Numerical examples are given in Section 4 to show the efficiency of the CSCS iteration. Finally, concluding remarks are given in Section 5.

2. The CSCS iteration

Firstly, let us give the definitions of Toeplitz, circulant, and skew-circulant matrices.

DEFINITION 2 *An $N \times N$ matrix B is said to be Toeplitz if*

$$B = \begin{bmatrix} b_0 & b_{-1} & \cdots & b_{2-N} & b_{1-N} \\ b_1 & b_0 & b_{-1} & \ddots & b_{2-N} \\ \vdots & b_1 & b_0 & \ddots & \vdots \\ b_{N-2} & \ddots & \ddots & \ddots & b_{-1} \\ b_{N-1} & b_{N-2} & \cdots & b_1 & b_0 \end{bmatrix},$$

i.e. B is constant along its diagonals. The Toeplitz matrix B is said to be circulant if $b_k = b_{k-N}$ for all $k = 1, \ldots, N-1$. If $b_k = -b_{k-N}$ for all $k = 1, \ldots, N-1$, the Toeplitz matrix B is said to be skew-circulant.

It is known that all Toeplitz matrices can be decomposed as a sum of a circulant matrix and a skew-circulant matrix [12]. Note that the matrices Q and W defined in Equation (8) are Toeplitz, and therefore possess the following circulant and skew-circulant splitting

$$Q = \tfrac{1}{2}(C_q + S_q),$$
$$W = \tfrac{1}{2}(C_w + S_w),$$

where

$$C_q = \begin{bmatrix} q_1^{(\alpha)} & q_0^{(\alpha)} + q_N^{(\alpha)} & q_{N-1}^{(\alpha)} & \cdots & q_3^{(\alpha)} & q_2^{(\alpha)} \\ q_2^{(\alpha)} & q_1^{(\alpha)} & q_0^{(\alpha)} + q_N^{(\alpha)} & q_{N-1}^{(\alpha)} & \ddots & q_3^{(\alpha)} \\ \vdots & q_2^{(\alpha)} & q_1^{(\alpha)} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & q_{N-1}^{(\alpha)} \\ q_{N-1}^{(\alpha)} & \ddots & \ddots & \ddots & q_1^{(\alpha)} & q_0^{(\alpha)} + q_N^{(\alpha)} \\ q_N^{(\alpha)} + q_0^{(\alpha)} & q_{N-1}^{(\alpha)} & \cdots & \cdots & q_2^{(\alpha)} & q_1^{(\alpha)} \end{bmatrix}, \tag{9}$$

$$S_q = \begin{bmatrix} q_1^{(\alpha)} & q_0^{(\alpha)} - q_N^{(\alpha)} & -q_{N-1}^{(\alpha)} & \cdots & -q_3^{(\alpha)} & -q_2^{(\alpha)} \\ q_2^{(\alpha)} & q_1^{(\alpha)} & q_0^{(\alpha)} - q_N^{(\alpha)} & -q_{N-1}^{(\alpha)} & \ddots & -q_3^{(\alpha)} \\ \vdots & q_2^{(\alpha)} & q_1^{(\alpha)} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & -q_{N-1}^{(\alpha)} \\ q_{N-1}^{(\alpha)} & \ddots & \ddots & \ddots & q_1^{(\alpha)} & q_0^{(\alpha)} - q_N^{(\alpha)} \\ q_N^{(\alpha)} - q_0^{(\alpha)} & q_{N-1}^{(\alpha)} & \cdots & \cdots & q_2^{(\alpha)} & q_1^{(\alpha)} \end{bmatrix}, \tag{10}$$

$$C_w = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & -1 \\ -1 & 0 & 1 & 0 & \ddots & 0 \\ 0 & -1 & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 1 \\ 1 & 0 & \cdots & 0 & -1 & 0 \end{bmatrix} \quad \text{and} \quad S_w = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 1 \\ -1 & 0 & 1 & 0 & \ddots & 0 \\ 0 & -1 & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 1 \\ -1 & 0 & \cdots & 0 & -1 & 0 \end{bmatrix}. \tag{11}$$

Then the coefficient matrix A of Equation (5), defined in Equation (6), possesses the following circulant and skew-circulant splitting

$$A = C + S,$$

where

$$\begin{aligned} C &= \tfrac{1}{2}(I - \nu_{\tau,h,\alpha} d_+ C_q - \nu_{\tau,h,\alpha} d_- C_q^{\mathrm{T}} - \nu\mu_{\tau,h} C_w), \\ S &= \tfrac{1}{2}(I - \nu_{\tau,h,\alpha} d_+ S_q - \nu_{\tau,h,\alpha} d_- S_q^{\mathrm{T}} - \nu\mu_{\tau,h} S_w). \end{aligned} \tag{12}$$

CSCS iteration for solving the Toeplitz system (5) is defined as follows. Take the initial guess as $u^{m+1,0} = u^m$. For $k = 0, 1, 2, \ldots$, until convergence, solve the following couple of linear systems

$$\begin{aligned} (\sigma I + C)u^{m+1,k+1/2} &= (\sigma I - S)u^{m+1,k} + b^m, \\ (\sigma I + S)u^{m+1,k+1} &= (\sigma I - C)u^{m+1,k+1/2} + b^m, \end{aligned} \tag{13}$$

repeatedly to generate the sequence $\{u^{m+1,k}\}_k$, where σ is a given positive parameter. Then take u^{m+1} as the 'limit' of the sequence $\{u^{m+1,k}\}_k$. An algorithm for the CSCS iteration is given in Algorithm 1.

Algorithm 1 CSCS iteration for solving $Au^{m+1} = b^m$

1. Take the initial guess as $u^{m+1,0} = u^m$
2. Choose the value of $\sigma > 0$
3. For $k = 0, 1, 2, \ldots$, until convergence, Do
4. $\quad r^{m+1,k} = b^m - Au^{m+1,k}$
5. $\quad (\sigma I + C)v^{m+1,k+\frac{1}{2}} = r^{m+1,k}$
6. $\quad u^{m+1,k+\frac{1}{2}} = u^{m+1,k} + v^{m+1,k+\frac{1}{2}}$
7. $\quad r^{m+1,k+\frac{1}{2}} = b^m - Au^{m+1,k+\frac{1}{2}}$
8. $\quad (\sigma I + S)v^{m+1,k+1} = r^{m+1,k+\frac{1}{2}}$
9. $\quad u^{m+1,k+1} = u^{m+1,k+\frac{1}{2}} + v^{m+1,k+1}$
10. EndDo

In the following, we analyse the operation cost of Algorithm 1. It is well-known that circulant matrices and skew-circulant matrices can be diagonalized [6] by the Fourier matrix

$$F = [F_{j,k}], \quad F_{j,k} = \frac{1}{\sqrt{N}} \mathrm{e}^{(2\pi \mathbf{i}/N)(j-1)(k-1)}, \quad 1 \le j, k \le N,$$

and the scaled Fourier matrix

$$\hat{F} = F\Omega, \quad \Omega = \mathrm{diag}\left(1, \mathrm{e}^{-\pi\mathbf{i}/N}, \ldots, \mathrm{e}^{-(N-1)\pi\mathbf{i}/N}\right),$$

respectively. Therefore, we have

$$C = F^* \Lambda_C F \quad \text{and} \quad S = \hat{F}^* \Lambda_S \hat{F}, \tag{14}$$

where Λ_C and Λ_S are diagonal matrices storing the eigenvalues of C and S, respectively. Denote c_1 and s_1 as the first columns of C and S, respectively. From Equation (14), one can deduce that the main diagonal vectors of Λ_C and Λ_S are $\sqrt{N}Fc_1$ and $\sqrt{N}\hat{F}s_1$, see [5] for instant, which can be obtained by using FFTs in $\mathcal{O}(N \log N)$ operations.

In Steps 4 and 7 of Algorithm 1, the matrix–vector product Az, where z is a vector, can be done by using FFT and inverse FFT in $\mathcal{O}(N \log N)$ operations since

$$Az = Cz + Sz = F^* \Lambda_C F z + \hat{F}^* \Lambda_S \hat{F} z.$$

In Steps 5 and 8 of Algorithm 1, the linear systems can also be solved by using FFT and inverse FFT in $\mathcal{O}(N \log N)$ operations since

$$(\sigma I + C)^{-1} z = F^* (\sigma I + \Lambda_C)^{-1} F z \quad \text{and} \quad (\sigma I + S)^{-1} z = \hat{F}^* (\sigma I + \Lambda_S)^{-1} \hat{F} z.$$

The right-hand side vector b^m in Equation (7) can also be performed by using FFT in $\mathcal{O}(N \log N)$ operations by noting that

$$b^m = (2I - A)u^m + \tau f^{m+1/2}.$$

Therefore, the total operation cost in each iteration of Algorithm 1 is $\mathcal{O}(N \log N)$.

3. Convergence of the CSCS iteration

From [10], we know that a sufficient condition for the convergence of CSCS iteration is that the splitting matrices C and S are positive stable. To prove the convergence of the CSCS iteration for solving FADEs (1), we introduce the following two lemmas.

LEMMA 1 [17,18] *For all $\alpha \in (1,2)$, the sequence $\{q_k^{(\alpha)}\}$ defined in Equation (3) satisfies the following properties*:

$$q_1^{(\alpha)} < 0, \quad q_2^{(\alpha)} \begin{cases} > 0, & \alpha \in (\alpha_0, 2), \\ \leq 0, & \alpha \in (1, \alpha_0], \end{cases} \quad q_0^{(\alpha)} \geq q_3^{(\alpha)} \geq q_4^{(\alpha)} \cdots \geq 0, \quad q_0^{(\alpha)} + q_2^{(\alpha)} \geq 0, \quad \sum_{k=0}^{\infty} q_k^{(\alpha)} = 0,$$

where $\alpha_0 \approx 1.5545$.

LEMMA 2 *For all $\alpha \in (1,2)$, the matrices C_q and S_q defined in Equations (9) and (10) are negative stable.*

Proof Denote $\{\lambda_j(C_q), j = 1, 2, \ldots, N\}$ as the spectrum of C_q. From [6],

$$\lambda_j(C_q) = q_1^{(\alpha)} + (q_0^{(\alpha)} + q_N^{(\alpha)})\gamma_j + q_{N-1}^{(\alpha)} \gamma_j^2 + \cdots + q_3^{(\alpha)} \gamma_j^{N-2} + q_2^{(\alpha)} \gamma_j^{N-1},$$

where $\gamma_j = \exp(\mathbf{i}\theta_j)$, $\theta_j = 2\pi(j-1)/N$, $\mathbf{i} = \sqrt{-1}$ and N is the matrix size. Then

$$\begin{aligned} \mathrm{Re}(\lambda_j(C_q)) &= \tfrac{1}{2}(\lambda_j(C_q) + \overline{\lambda_j(C_q)}) \\ &= q_1^{(\alpha)} + (q_0^{(\alpha)} + q_N^{(\alpha)}) \cos\theta_j + q_{N-1}^{(\alpha)} \cos(2\theta_j) + \cdots \\ &\quad + q_3^{(\alpha)} \cos((N-2)\theta_j) + q_2^{(\alpha)} \cos((N-1)\theta_j) \\ &= q_1^{(\alpha)} + (q_0^{(\alpha)} + q_N^{(\alpha)} + q_2^{(\alpha)}) \cos\theta_j + q_{N-1}^{(\alpha)} \cos(2\theta_j) + \cdots \\ &\quad + q_3^{(\alpha)} \cos((N-2)\theta_j). \end{aligned} \tag{15}$$

From Lemma 1, the coefficient of each cosine in Equation (15) is non-negative and we have

$$\mathrm{Re}(\lambda_j(C_q)) \leq \sum_{k=0}^{N} q_k^{(\alpha)} < \sum_{k=0}^{\infty} q_k^{(\alpha)} = 0, \quad \text{for } \forall \alpha \in (1,2).$$

For the skew-circulant matrix S_q, from [6],

$$\lambda_j(S_q) = q_1^{(\alpha)} + (q_0^{(\alpha)} - q_N^{(\alpha)})\zeta_j - q_{N-1}^{(\alpha)}\zeta_j^2 - \cdots - q_3^{(\alpha)}\zeta_j^{N-2} - q_2^{(\alpha)}\zeta_j^{N-1},$$

where $\zeta_j = \exp(\mathbf{i}\phi_j)$ and $\phi_j = (2j-1)\pi/N$. Then we have

$$\begin{aligned}
\mathrm{Re}(\lambda_j(S_q)) &= \frac{1}{2}(\lambda_j(S_q) + \overline{\lambda_j(S_q)}) \\
&= q_1^{(\alpha)} + (q_0^{(\alpha)} - q_N^{(\alpha)})\cos\phi_j - q_{N-1}^{(\alpha)}\cos(2\phi_j) - \cdots \\
&\quad - q_3^{(\alpha)}\cos((N-2)\phi_j) - q_2^{(\alpha)}\cos((N-1)\phi_j) \\
&= q_1^{(\alpha)} + (q_0^{(\alpha)} + q_2^{(\alpha)})\cos\phi_j - q_N^{(\alpha)}\cos\phi_j - q_{N-1}^{(\alpha)}\cos(2\phi_j) - \cdots \\
&\quad - q_3^{(\alpha)}\cos((N-2)\phi_j) \\
&\le q_1^{(\alpha)} + q_0^{(\alpha)} + q_2^{(\alpha)} + q_N^{(\alpha)} + q_{N-1}^{(\alpha)} + \cdots + q_3^{(\alpha)} \\
&< \sum_{k=0}^{\infty} q_k^{(\alpha)} = 0,
\end{aligned}$$

for any $\alpha \in (1,2)$. ■

Remark 1 For $\alpha \in (\alpha_0, 2)$, by noting that $q_2^{(\alpha)} > 0$ (see Lemma 1), Lemma 2 can be proved by applying the Gerschgorin theorem directly to the matrices C_q and S_q.

Now, let $C_q = F^*\Lambda_{C_q}F$, $S_q = \hat{F}^*\Lambda_{S_q}\hat{F}$, $C_w = F^*\Lambda_{C_w}F$, and $S_w = \hat{F}^*\Lambda_{S_w}\hat{F}$. By Equations (12) and (14), we have

$$\begin{aligned}
\Lambda_C &= \tfrac{1}{2}(I - \nu_{\tau,h,\alpha}d_+\Lambda_{C_q} - \nu_{\tau,h,\alpha}d_-\Lambda_{C_q}^* - \nu\mu_{\tau,h}\Lambda_{C_w}), \\
\Lambda_S &= \tfrac{1}{2}(I - \nu_{\tau,h,\alpha}d_+\Lambda_{S_q} - \nu_{\tau,h,\alpha}d_-\Lambda_{S_q}^* - \nu\mu_{\tau,h}\Lambda_{S_w}).
\end{aligned}$$

Note that the matrices C_w and S_w in Equation (11) are skew-symmetric, then

$$\mathrm{Re}(\Lambda_{C_w}) = \mathrm{Re}(\Lambda_{S_w}) = 0,$$

and since $\nu_{\tau,h,\alpha} > 0$, $d_\pm \ge 0$ with $d_+ + d_- \ne 0$, and by Lemma 2, we get that

$$\begin{aligned}
\mathrm{Re}([\Lambda_C]_{jj}) &= \tfrac{1}{2}(1 - \nu_{\tau,h,\alpha}(d_+ + d_-)\mathrm{Re}([\Lambda_{C_q}]_{jj})) > 0, \\
\mathrm{Re}([\Lambda_S]_{jj}) &= \tfrac{1}{2}(1 - \nu_{\tau,h,\alpha}(d_+ + d_-)\mathrm{Re}([\Lambda_{S_q}]_{jj})) > 0,
\end{aligned}$$

for all $j = 1, \ldots, N$. Therefore, matrices C and S are positive stable.

Now, by the Theorem 1 in [10], we get a theorem for the convergence of the CSCS iteration.

THEOREM 1 *For all $\nu_{\tau,h,\alpha} > 0$, the spectral radius $\rho(T(\sigma))$ of the iteration matrix of* (13)

$$T(\sigma) = (\sigma I + S)^{-1}(\sigma I - C)(\sigma I + C)^{-1}(\sigma I - S)$$

is bounded by

$$\eta(\sigma) \equiv \max_{\lambda_j \in \lambda(C)} \frac{|\sigma - \lambda_j|}{|\sigma + \lambda_j|} \cdot \max_{\mu_j \in \lambda(S)} \frac{|\sigma - \mu_j|}{|\sigma + \mu_j|},$$

and it holds that

$$\rho(T(\sigma)) \le \eta(\sigma) < 1, \quad \forall \sigma > 0,$$

i.e. the CSCS iteration converges to the exact solution of the linear system (5) *for all initial guess and all $\sigma > 0$.*

The convergence rate of the CSCS iteration depends on the choice of initial guess and σ. In our calculations, σ is chosen according to the following formula

$$\sigma = \begin{cases} \sqrt{\gamma_{\min}\gamma_{\max} - \zeta_{\max}^2}, & \text{for } \zeta_{\max} < \sqrt{\gamma_{\min}\gamma_{\max}}, \\ \sqrt{\gamma_{\min}^2 + \zeta_{\max}^2}, & \text{for } \zeta_{\max} \geq \sqrt{\gamma_{\min}\gamma_{\max}}, \end{cases} \tag{16}$$

where $\gamma_{\min}$ and $\gamma_{\max}$ are the lower and the upper bounds of the real part of the eigenvalues of the matrices C and S, and $\zeta_{\max}$ is the upper bound of the absolute values of the imaginary part of the eigenvalues of the matrices C and S. The σ in Equation (16) is obtained by minimizing an upper bound of $\rho(T(\sigma))$, see [10], and all quantities inside the radical sign in Equation (16) are positive since the matrices C and S are positive stable. To obtain a better σ, further study is required.

4. Numerical examples

In this section, we illustrate the efficiency of the CSCS iteration for solving FADEs (1). All experiments are performed in MATLAB R2010a on a Dell Vostro 260S desktop with the following configuration: Intel(R) Core(TM) i5 − 2400 CPU 3.10 GHz and 4 GB RAM. In the following numerical tests, we compare the CSCS iteration (13) for solving linear system (5) with the fast biconjugate gradient stabilized (FBICGSTAB) method. The FBICGSTAB method is the biconjugate gradient stabilized method [14] in which all matrix-vector multiplications Av, for any vector v, is done by using FFT in $\mathcal{O}(N \log N)$ operations [20]. Suppose the initial guess $u^{m+1,0}$ is not the solution of Equation (5), the stopping criteria of the FBICGSTAB method and the CSCS iteration are chosen as

$$\frac{\|r^{m+1,k}\|_2}{\|r^{m+1,0}\|_2} = \frac{\|b^m - Au^{m+1,k}\|_2}{\|b^m - Au^{m+1,0}\|_2} < 10^{-7},$$

which is strict enough to maintain the second-order convergence of the finite difference scheme (4), see Tables 1 and 2. In the following tables, 'CPU(s)' denotes the total CPU time in seconds, 'Iter' denotes the *average* number of iterations of the method over all time level for solving the whole

Table 1. Comparisons for solving Example 1 by the FBICGSTAB method and the CSCS iteration at $t = 1$.

		FBICGSTAB				CSCS				
α	N	$\|E\|_\infty$	Rate	Iter	CPU(s)	$\|E\|_\infty$	Rate	σ	Iter	CPU(s)
1.2	2^6	3.0330e−05	–	5.0	0.0347	3.0330e−05	–	0.62	4.0	0.0148
	2^7	8.0076e−06	1.92	5.0	0.0817	8.0076e−06	1.92	0.63	4.0	0.0401
	2^8	2.0531e−06	1.96	4.0	0.1822	2.0553e−06	1.96	0.65	4.0	0.1236
	2^9	5.2056e−07	1.98	4.0	0.5730	5.2042e−07	1.98	0.66	5.0	0.5570
	2^{10}	1.3096e−07	1.99	4.0	1.9612	1.3089e−07	1.99	0.67	5.0	2.0621
1.5	2^6	2.4994e−05	–	13.1	0.0867	2.4994e−05	–	1.36	11.0	0.0355
	2^7	6.1163e−06	2.03	14.7	0.2287	6.1168e−06	2.03	1.53	12.0	0.1066
	2^8	1.4851e−06	2.04	17.3	0.7214	1.4854e−06	2.04	1.69	14.0	0.3871
	2^9	3.5962e−07	2.05	19.6	2.5796	3.5985e−07	2.05	1.81	15.0	1.5503
	2^{10}	8.7073e−08	2.05	21.8	9.7279	8.7128e−08	2.05	1.78	16.0	6.0593
1.8	2^6	2.5819e−05	–	34.4	0.2246	2.5819e−05	–	3.54	27.0	0.0835
	2^7	6.5283e−06	1.98	50.3	0.7717	6.5284e−06	1.98	4.50	36.0	0.3076
	2^8	1.6337e−06	2.00	61.1	2.5129	1.6339e−06	2.00	5.60	45.1	1.1971
	2^9	4.0637e−07	2.01	78.3	10.1429	4.0647e−07	2.01	6.60	56.3	5.6647
	2^{10}	1.0072e−07	2.01	98.6	43.4173	1.0072e−07	2.01	6.55	79.9	29.4979

Table 2. Comparisons for solving Example 2 by the FBICGSTAB method and the CSCS iteration at $t = 1$.

		FBICGSTAB				CSCS				
α	N	$\|E\|_\infty$	Rate	Iter	CPU(s)	$\|E\|_\infty$	Rate	σ	Iter	CPU(s)
1.2	2^6	1.6456e−05	–	4.0	0.0298	1.6456e−05	–	0.53	3.0	0.0132
	2^7	4.1700e−06	1.98	4.0	0.0724	4.1700e−06	1.98	0.53	3.0	0.0372
	2^8	1.0513e−06	1.99	4.0	0.2023	1.0513e−06	1.99	0.53	3.0	0.1192
	2^9	2.6392e−07	1.99	3.0	0.5357	2.6392e−07	1.99	0.53	3.0	0.4399
	2^{10}	6.6125e−08	2.00	3.0	1.8599	6.6126e−08	2.00	0.53	3.0	1.6539
1.5	2^6	1.3607e−05	–	8.0	0.0550	1.3607e−05	–	0.82	6.0	0.0223
	2^7	3.4668e−06	1.97	8.0	0.1326	3.4668e−06	1.97	0.88	7.0	0.0692
	2^8	8.7650e−07	1.98	9.4	0.4212	8.7653e−07	1.98	0.92	7.0	0.2222
	2^9	2.2058e−07	1.99	11.0	1.5721	2.2060e−07	1.99	0.89	7.0	0.8290
	2^{10}	5.5391e−08	1.99	11.1	5.3826	5.5370e−08	1.99	0.64	7.0	3.1002
1.8	2^6	1.1927e−05	–	18.9	0.1246	1.1927e−05	–	1.96	16.0	0.0519
	2^7	3.0908e−06	1.95	26.3	0.4074	3.0909e−06	1.95	2.44	20.0	0.1750
	2^8	7.9376e−07	1.96	30.8	1.2916	7.9380e−07	1.96	2.92	24.0	0.6631
	2^9	2.0255e−07	1.97	36.4	4.8593	2.0258e−07	1.97	3.14	26.0	2.6835
	2^{10}	5.1466e−08	1.98	44.0	19.6817	5.1472e−08	1.98	1.57	46.6	17.4633

FADE problem, i.e.

$$\text{Iter} = \frac{1}{M}\sum_{m=1}^{M} \text{Iter}(m),$$

where Iter(m) is the number of iterations required for solving Equation (5) in the mth time level.

Example 1 Consider FADE (1) on space interval $[a, b] = [0, 1]$ and time interval $[0, T] = [0, 1]$ with diffusion coefficients $d_+ = 0.8$, $d_- = 0.5$, coefficient $v = -0.1$, initial condition $u_0(x) = x^2(1-x)^2$, and source term

$$f(x,t) = -\mathrm{e}^{-t}\left\{x^2(1-x)^2 + 2vx(1-x)(1-2x) + \frac{\Gamma(3)}{\Gamma(3-\alpha)}[d_+x^{2-\alpha} + d_-(1-x)^{2-\alpha}]\right.$$
$$\left. - 2\frac{\Gamma(4)}{\Gamma(4-\alpha)}[d_+x^{3-\alpha} + d_-(1-x)^{3-\alpha}] + \frac{\Gamma(5)}{\Gamma(5-\alpha)}[d_+x^{4-\alpha} + d_-(1-x)^{4-\alpha}]\right\}.$$

The exact solution of this example is $u(x,t) = \mathrm{e}^{-t}x^2(1-x)^2$. For the finite difference discretization, the space step and time step are taken to be $h = 1/(N+1)$ and $\tau = h$, respectively.

Table 1 shows the numerical results of solving Example 1 until $t = 1$. From the table, one can see that the CSCS iteration and the FBICGSTAB method are very efficient. For $\alpha = 1.2$, the performance of the CSCS iteration and the FBICGSTAB method are almost the same. For $\alpha = 1.5$ and 1.8, the CSCS iteration converges faster than the FBICGSTAB method. Table 1 also shows the second-order convergence of the numerical methods, which agrees with the order of convergence of the finite difference scheme (4).

Example 2 Consider FADE (1) on space interval $[a, b] = [0, 1]$ and time interval $[0, T] = [0, 1]$ with diffusion coefficients $d_+ = 0.1$, $d_- = 0.3$, advection coefficient $v = -0.1$, initial condition

$u_0(x) = \sin(1)x^3(1-x)^3$, and source term

$$f(x,t) = \cos(t+1)x^3(1-x)^3 - \sin(t+1)\left\{3vx^2(1-x)^2(1-2x)\right.$$
$$+\frac{\Gamma(4)}{\Gamma(4-\alpha)}[d_+x^{3-\alpha}+d_-(1-x)^{3-\alpha}] - 3\frac{\Gamma(5)}{\Gamma(5-\alpha)}[d_+x^{4-\alpha}+d_-(1-x)^{4-\alpha}]$$
$$\left.+3\frac{\Gamma(6)}{\Gamma(6-\alpha)}[d_+x^{5-\alpha}+d_-(1-x)^{5-\alpha}] - \frac{\Gamma(7)}{\Gamma(7-\alpha)}[d_+x^{6-\alpha}+d_-(1-x)^{6-\alpha}]\right\}$$

The exact solution of this example is $u(x,t) = \sin(t+1)x^3(1-x)^3$. For the finite difference discretization, the space step and time step are taken to be $h = 1/(N+1)$ and $\tau = h$, respectively.

Table 2 shows the performance of the CSCS iteration and the FBICGSTAB method for solving Example 2. Similar to Example 1, the CSCS iteration performs better when $\alpha = 1.5$ and 1.8. For $\alpha = 1.2$, the two methods perform almost the same.

5. Concluding remarks

In this paper, CSCS iteration is employed to solve FADE (1) with second-order accuracy. The complexity in each iteration of the method is $\mathcal{O}(N \log N)$ by using FFTs. The method is proved to be convergent globally and numerical examples reveal that the convergence rate of the CSCS iteration is fast.

We remark that if the second-order finite difference approximation (2) for fractional derivatives is replaced by the one proposed in [19], the convergence of the CSCS iteration can be proved similarly. Besides, the CSCS iteration can be extended to solve high-dimensional FADEs through the alternating direction method proposed in [7]. The CSCS iteration for high-dimensional FADEs could be much benefited from parallel computing.

Acknowledgements

Siu-Long Lei is supported by research grant MYRG071(Y1-L2)-FST13-LSL from University of Macau. Seak-Weng Vong is supported by the Macao Science and Technology Development Fund FDCT/001/2013/A.

References

[1] J. Bai and X. Feng, *Fractional-order anisotropic diffusion for image denoising*, IEEE Trans. Image Process. 16 (2007), pp. 2492–2502.
[2] D. Benson, S.W. Wheatcraft, and M.M. Meerschaert, *Application of a fractional advection-dispersion equation*, Water Resour. Res. 36 (2000), pp. 1403–1413.
[3] D. Benson, S.W. Wheatcraft, and M.M. Meerschaert, *The fractional-order governing equation of Lévy motion*, Water Resour. Res. 36 (2000), pp. 1413–1423.
[4] B.A. Carreras, V.E. Lynch, and G.M. Zaslavsky, *Anomalous diffusion and exit time distribution of particle tracers in plasma turbulence models*, Phys. Plasma 8 (2001), pp. 5096–5103.
[5] R. Chan and X. Jin, *An Introduction to Iterative Toeplitz Solvers*, SIAM, Philadelphia, 2007.
[6] P. Davis, *Circulant Matrices*, John Wiley & Sons, Inc., New York, 1979.
[7] W. Deng and M. Chen, *Efficient numerical algorithms for three-dimensional fractional partial differential equations* (2013). Available at arXiv:1303.4628v1 [math.NA].
[8] S. Lei and H. Sun, *A circulant preconditioner for fractional diffusion equations*, J. Comput. Phys. 242 (2013), pp. 715–725.
[9] R.L. Magin, *Fractional Calculus in Bioengineering*, Begell House Publishers, Connecticut, 2006.
[10] M. Ng, *Circulant and skew-circulant splitting methods for Toeplitz systems*, J. Comput. Appl. Math. 159 (2003), pp. 101–108.

[11] H. Pang and H. Sun, *Multigrid method for fractional diffusion equations*, J. Comput. Phys. 231 (2012), pp. 693–703.
[12] L. Pustylnikov, *The algebraic structure of spaces of Toeplitz and Hankel matrices* (in Russian), Dokl. Akad. Nauk USSR 250 (1980), pp. 556–559.
[13] M. Raberto, E. Scalas, and F. Mainardi, *Waiting-times and returns in high-frequency financial data: An empirical study*, Phys. A 314 (2002), pp. 749–755.
[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[15] M.F. Shlesinger, B.J. West, and J. Klafter, *Lévy dynamics of enhanced diffusion: Application to turbulence*, Phys. Rev. Lett. 58 (1987), pp. 1100–1103.
[16] I.M. Sokolov, J. Klafter, and A. Blumen, *Fractional kinetics*, Phys. Today Nov. (2002), pp. 28–53.
[17] E. Sousa, *A second order explicit finite difference method for the fractional advection diffusion equation*, Comput. Math. Appl. 64 (2012), pp. 3141–3152.
[18] E. Sousa and C. Li, *A weighted finite difference method for the fractional diffusion equation based on the Riemann–Liouville derivative* (2011). Available at arXiv:1109.2345v1 [math.NA].
[19] W. Tian, H. Zhou, and W. Deng, *A class of second order difference approximations for solving space fractional diffusion equations*, Math. Comput. (in press).
[20] H. Wang and T.S. Basu, *A fast finite difference method for two-dimensional space-fractional diffusion equations*, SIAM J. Sci. Comput. 34 (2012), pp. A2444–A2458.
[21] H. Wang, K. Wang, and T. Sircar, *A direct* $\mathcal{O}(N \log^2 N)$ *finite difference method for fractional diffusion equations*, J. Comput. Phys. 229 (2010), pp. 8095–8104.
[22] G.M. Zaslavsky, D. Stevens, and H. Weitzner, *Self-similar transport in incomplete chaos*, Phys. Rev. E 48 (1993), pp. 1683–1694.
[23] M. Zhu and G. Zhang, *On CSCS-based iteration methods for Toeplitz system of weakly nonlinear equations*, J. Comput. Appl. Math. 235 (2011), pp. 5095–5104.