Turkish Journal of Mathematics

Volume 40 | Number 6 Article 13

1-1-2016

On the NPHSS-KPIK iteration method for low-rank complex Sylvester equations arising from time-periodic fractional diffusion equations

MIN-LI ZENG

GUO-FENG ZHANG

TÜBİTAK

Turkish Journal of Mathematics

http://journals.tubitak.gov.tr/math/

Research Article

Turk J Math
(2016) 40: 1325 – 1339
© TÜBİTAK
doi:10.3906/mat-1510-93

On the NPHSS-KPIK iteration method for low-rank complex Sylvester equations arising from time-periodic fractional diffusion equations

Min-Li ZENG[1,2], **Guo-Feng ZHANG**[2,*]
[1]School of Mathematics and Statistics, Lanzhou University, Lanzhou, P.R. China
[2]School of Mathematics, Putian University, Putian, P.R. China

Received: 25.10.2015 • **Accepted/Published Online:** 16.02.2016 • **Final Version:** 02.12.2016

Abstract: Based on the Hermitian and skew-Hermitian (HS) splitting for non-Hermitian matrices, a nonalternating preconditioned Hermitian and skew-Hermitian splitting-Krylov plus inverted Krylov subspace (NPHSS-KPIK) iteration method for solving a class of large and low-rank complex Sylvester equations arising from the two-dimensional time-periodic fractional diffusion problem is established. The local convergence condition is proposed and the optimal parameter is given. Numerical experiments are used to show the efficiency of the NPHSS-KPIK iteration method for solving the Sylvester equations arising from the time-periodic fractional diffusion equations.

Key words: Sylvester equation, Krylov-plus-inverted-Krylov subspace method, time-periodic fractional diffusion equation, NPHSS method, low-rank

1. Introduction

Consider the following Sylvester equation

$$AU + UB = C, \tag{1.1}$$

where $A \in \mathbb{C}^{m\times m}$, $B \in \mathbb{C}^{n\times n}$, and $C \in \mathbb{C}^{m\times n}$ are given complex matrices. Assume

(A_1) A, B, and C are large matrices;

(A_2) $A = W_1 + \mathbf{i}T_1$, $B = W_2 + \mathbf{i}T_2$, with W_1 and W_2 being both symmetric positive matrices, at least one of T_1 and T_2 being a nonzero matrix;

(A_3) $C = FG^T$ has much lower rank than the problem size, i.e. $F \in \mathbb{C}^{m\times s}$, $G \in \mathbb{C}^{n\times s}$, where $s \ll m$ and $s \ll n$.

The Sylvester equation of the form (1.1) arises in numerous applications, such as control and system theory, image restoration, and the discretized approximation of fractional diffusion equations. We refer to [3] for a detailed description. If $C \neq 0$, according to [3], when A and $-B$ do not have disjoint spectra, the Sylvester equation (1.1) has a unique solution under the assumptions (A_1)–(A_3).

The standard methods for numerical solution of the Sylvester equation (1.1) are the Bartels–Stewart [7] and the Hessenberg–Schur methods [9]. They consist of transforming the matrices A and B into triangular

*Correspondence: gf_zhang@lzu.edu.cn

2000 *AMS Mathematics Subject Classification:* 65F10, 65N22.
This work was supported by the National Natural Science Foundation of China (No.11271174, 11471175, 11511130015), the Natural Science Foundation of Fujian Province (No.2016J05016), and the Scientific Research Project of Putian University (No.2015061, 2016021, 2016075).

Hessenberg form by an orthogonal similarity transformation or a full Schur decomposition and then solving the resulting system of linear equations directly by a backward substitution. The Bartels–Stewart and the Hessenberg–Schur methods are applicable and effective for general Sylvester equations of reasonably small sizes.

When A and B are large and sparse, iterative methods such as Smith's method [22], the alternating direction implicit (ADI) method [25], the successive over-relaxation (SOR) method [24], and the matrix splitting methods [10] are often the methods of choices for efficiently and accurately solving the Sylvester equation (1.1). Those iterative methods are efficient when A and B are both Hermitian. However, when the matrix A or B is not Hermitian, the convergence of those iterative methods may be theoretically not guaranteed, even if both of A and B are either asymptotically stable or N-stable, or the skew-Hermitian part of A or B is dominantly strong.

When A and B are positive semidefinite, and at least one of them is positive definite, based on the Hermitian and skew-Hermitian (HS) splittings [5], Bai [3] proposed a Hermitian and skew-Hermitian splitting (HSS) iteration method. The HSS iteration method for solving Sylvester equations is decomposed into a sequence of subproblems about two coupled Sylvester equations with respect to shifted Hermitian positive definite matrices and shifted skew-Hermitian matrices, respectively. The HSS iteration method is an efficient and robust solver for the large sparse Sylvester equations with non-Hermitian and positive definite matrices. For more details about the solvers for the Sylvester equation, we refer to [26, 29] and some references therein.

It is well known that the Sylvester equation (1.1) can be rewritten mathematically equivalent to the linear system $(I \otimes A + B^T \otimes I)x = c$, where the vectors x and c contain the concatenated columns of the matrices X and C, respectively, and $\otimes$ denotes the Kronecker product. However, this is a numerically poor way to get the solution X, because the linear system matrix is of size $mn \times mn$, which is an extremely large number.

When the right-hand side matrix C is lower rank than the problem dimension, Simoncini et al. [21] proposed a Krylov-plus-inverted-Krylov (KPIK) subspace method for solving large-scale Lyapunov equations. The KPIK subspace method projects the problem onto a much smaller approximation space; then the reduced problem can be solved by means of a direct Schur decomposition method. For more details, we refer to [12].

In this paper, we concentrate on an efficient solver for the discretized Sylvester equation arising from two-dimensional time-periodic fractional diffusion equations. The resulting system is a complex discretization Sylvester equation, which is obtained by applying the shifted Grünwald finite difference discretization both of x-direction and y-direction space-fractional diffusion terms. According to the special structure and extremely great size properties of the discretized coefficient matrix, we use the project method to rewrite the original discretization system as a reduced Sylvester equation, which can be directly solved by Schur decomposition method. The iterative generation of the extended space requires solving systems with coefficient matrices A and B, which are both complex symmetric, non-Hermitian, and of great size. Therefore, based on the single-step HSS iteration scheme proposed by Li and Wu et al.[13] for solving the linear systems at each step of the KPIK method, we name the new method the NPHSS-KPIK method. The local convergent properties for the NPHSS-KPIK method are proposed and the optimal parameters are given. Numerical experiments also show that the NPHSS-KPIK iteration method is an efficient and robust solver for the low-rank complex Sylvester equation (1.1).

The remainder of this paper is organized as follows. In Section 2, we review the discretized Sylvester matrix equation from the two-dimensional time-periodic fractional diffusion equations (FDEs). Then we derive the NPHSS-KPIK iteration method for the Sylvester equation. In Section 3, we study the local convergence properties and the optimal choice of the iterative parameter for the NPHSS-KPIK method. Numerical exper-

iments are presented in Section 4 to illustrate the efficiency of the NPHSS-KPIK iteration method for solving two-dimensional time-periodic FDEs. In Section 5, some conclusions are given to end this paper.

Throughout the paper, we use $\mathbf{i} = \sqrt{-1}$ to denote the imaginary unit. $\|\cdot\|_2$ is defined as the 2-norm of a vector or a matrix. M^T and M^* denote the transpose and the conjugate transpose of the matrix M, respectively. Range(M) indicates the space spanned by the columns of M.

2. The discretized system and the NPHSS-KPIK method

Consider the following two-dimensional time-periodic FDEs

$$\begin{cases} \dfrac{\partial u(x,y,t)}{\partial t} - {}_xD_R^{\beta_1}u(x,y,t) - {}_yD_R^{\beta_2}u(x,y,t) = f(x,y,t), \quad a_x \le x \le b_x, \quad a_y \le y \le b_y, t \in [0,T], \\ u(a_x,y,t) = u(b_x,y,t) = u(x,a_y,t) = u(x,b_y,t) = 0, \quad 0 \le t \le T, \\ u(x,y,0) = 0, \quad a_x \le x \le b_x, a_y \le y \le b_y, \end{cases} \tag{2.1}$$

where $f(x,y,t)$ is the source and sink term, the differentiation parameter $\beta_1, \beta_2 \in (1,2)$. Assume there is a time-periodic solution to the system (2.1), i.e. $u(x,y,t) = u(x,y)e^{\mathbf{i}\omega t}$ with $\omega = \frac{2\pi M}{T}$ for some $M \in \mathbb{Z}$. Here the Riesz derivatives ${}_xD_R^{\beta_1}u(x,y)$ and ${}_yD_R^{\beta_2}u(x,y)$ are defined as the weighted average of the left- and right-sided derivative [17]. That is,

$$ {}_xD_R^{\beta_1}u(x,y,t) = \frac{1}{2}\left(\frac{\partial^{\beta_1}u(x,y,t)}{\partial_+ x^{\beta_1}} + \frac{\partial^{\beta_1}u(x,y)}{\partial_- x^{\beta_1}}\right),$$

$$ {}_yD_R^{\beta_2}u(x,y,t) = \frac{1}{2}\left(\frac{\partial^{\beta_2}u(x,y,t)}{\partial_+ y^{\beta_2}} + \frac{\partial^{\beta_2}u(x,y)}{\partial_- y^{\beta_2}}\right).$$

The left and right fractional derivatives are defined in the Grünwald–Letnikov form [16]

$$\frac{\partial^{\beta_1}u(x,y,t)}{\partial_+ x^{\beta_1}} = \lim_{h_x\to 0}\frac{1}{h_x^{\beta_1}}\sum_{k=0}^{\lfloor (x-a_x)/h_x\rfloor} g_k^{(\beta_1)}u(x-kh_x,y,t),$$

$$\frac{\partial^{\beta_1}u(x,y,t)}{\partial_- x^{\beta_1}} = \lim_{h_x\to 0}\frac{1}{h_x^{\beta_1}}\sum_{k=0}^{\lfloor (b_x-x)/h_x\rfloor} g_k^{(\beta_1)}u(x+kh_x,y,t),$$

$$\frac{\partial^{\beta_2}u(x,y,t)}{\partial_+ y^{\beta_2}} = \lim_{h_y\to 0}\frac{1}{h_y^{\beta_2}}\sum_{k=0}^{\lfloor (y-a_y)/h_y\rfloor} g_k^{(\beta_2)}u(x,y-kh_y,t),$$

$$\frac{\partial^{\beta_2}u(x,y,t)}{\partial_- y^{\beta_2}} = \lim_{h_y\to 0}\frac{1}{h_y^{\beta_2}}\sum_{k=0}^{\lfloor (b_y-y)/h_y\rfloor} g_k^{(\beta_2)}u(x,y+kh_y,t),$$

where $\lfloor x\rfloor$ denotes the largest integer that is not greater than x and the coefficients $g_k^{(\beta)}$ $(k = 1,2,\cdots.)$ are defined as

$$g_0^{(\beta)} = 1, \quad g_k^{(\beta)} = \frac{(-1)^k}{k!}\beta(\beta-1)\cdots(\beta-k+1).$$

The FDEs of form (2.1) arise in variety of research areas such as modeling chaotic dynamics of classical conservative systems [27], turbulent flow [20], and groundwater contaminant transport [8], and applications in finance [18], image processing [2], physics [23], and biology [14].

Because the solution $u(x,y,t)$ of Eq. (2.1) is time-periodic, then $u(x,y,t)$ and $f(x,y,t)$ are both time-harmonic, i.e. $u(x,y,t)=u(x,y)e^{\mathbf{i}\omega t}$ and $f(x,y,t)=f(x,y)e^{\mathbf{i}\omega t}$, where $u(x,y)$ and $f(x,y)$ solve the following time-independent fractional differential equation:

$$\mathbf{i}\omega u(x,y)-{}_xD_R^{\beta_1}u(x,y)-{}_yD_R^{\beta_2}u(x,y)=f(x,y),\quad a_x\le x\le b_x,\quad a_y\le y\le b_y. \tag{2.2}$$

Let $h_x=\frac{b_x-a_x}{N_x+1}$, $h_y=\frac{b_y-a_y}{N_y+1}$ be the mesh grid of the $x-$direction and $y-$direction, respectively. Use the spatial partitions:

$$x_i=a_x+ih_x,\quad y_j=a_y+jh_y,\quad i=0,1,\cdots,N_x+1;\quad j=0,1,\cdots,N_y+1.$$

Denote $u_{i,j}=u(x_i,y_j),\quad f_{i,j}=f(x_i,y_j)$. At each node (x_i,y_j), we employ the shifted Grünwald approximations [15] as

$$\begin{aligned}
\frac{\partial^{\beta_1}u(x_i,y_j)}{\partial_+x^{\beta_1}}&=\frac{1}{h_x^{\beta_1}}\sum_{k=0}^{i+1}g_k^{(\beta_1)}u_{i-k+1,j}+O(h_x),\\
\frac{\partial^{\beta_1}u(x_i,y_j)}{\partial_-x^{\beta_1}}&=\frac{1}{h_x^{\beta_1}}\sum_{k=0}^{N_x-i+2}g_k^{(\beta_1)}u_{i+k-1,j}+O(h_x),\\
\frac{\partial^{\beta_2}u(x_i,y_j)}{\partial_+y^{\beta_2}}&=\frac{1}{h_y^{\beta_2}}\sum_{k=0}^{j+1}g_k^{(\beta_2)}u_{i,j-k+1}+O(h_y),\\
\frac{\partial^{\beta_2}u(x_i,y_j)}{\partial_-y^{\beta_2}}&=\frac{1}{h_y^{\beta_2}}\sum_{k=0}^{N_y-j+2}g_k^{(\beta_2)}u_{i,j+k-1}+O(h_y).
\end{aligned}$$

Then Eq. (2.2) leads to the following finite difference scheme:

$$-\frac{1}{2h_x^{\beta_1}}\Big(\sum_{k=0}^{i+1}g_k^{(\beta_1)}u_{i-k+1,j}+\sum_{k=0}^{N_x-i+2}g_k^{(\beta_1)}u_{i+k-1,j}\Big)-\frac{1}{2h_y^{\beta_2}}\Big(\sum_{k=0}^{j+1}g_k^{(\beta)}u_{i,j-k+1}+\sum_{k=0}^{N_y-j+2}g_k^{(\beta)}u_{i,j+k-1}\Big)=f_{i,j}. \tag{2.3}$$

According to the boundary condition $u_{0,j}=u_{N_x+1,j}=u_{i,0}=u_{i,N_y+1}=0$ and let

$$U=\begin{pmatrix}u_{1,1}&u_{1,2}&\cdots&u_{1,N_y}\\u_{2,1}&u_{2,2}&\cdots&u_{2,N_y}\\\vdots&\vdots&\vdots&\vdots\\u_{N_x,1}&u_{N_x,2}&\cdots&u_{N_x,N_y}\end{pmatrix},\quad C=\begin{pmatrix}f_{1,1}&f_{1,2}&\cdots&f_{1,N_y}\\f_{2,1}&f_{2,2}&\cdots&f_{2,N_y}\\\vdots&\vdots&\vdots&\vdots\\f_{N_x,1}&f_{N_x,2}&\cdots&f_{N_x,N_y}\end{pmatrix};$$

then we can rewrite the finite difference scheme (2.3) into a matrix form

$$\mathbf{i}\omega U+(L_{\beta_1}^{N_x})^TU+U(L_{\beta_2}^{N_y})^T=C, \tag{2.4}$$

where $L_\beta^N=\frac{1}{2}(G_\beta^N+(G_\beta^N)^T)$ is a Toeplitz matrix, and

$$G_\beta^N=-\begin{pmatrix}g_1^{(\beta)}&g_0^{(\beta)}&0&\cdots&0\\g_2^{(\beta)}&g_1^{(\beta)}&g_0^{(\beta)}&\cdots&0\\g_3^{(\beta)}&\ddots&\ddots&\ddots&\vdots\\\vdots&\ddots&\ddots&\ddots&g_0^{(\beta)}\\g_N^{(\beta)}&\cdots&g_3^{(\beta)}&g_2^{(\beta)}&g_1^{(\beta)}\end{pmatrix},\quad (\beta,N)\in\{(\beta_1,N_x),(\beta_2,N_y)\}.$$

The linear equation (2.4) can be rewritten as

$$(\frac{1}{2}\mathbf{i}\omega I_{N_x} + L_{\beta_1}^{N_x})^T U + U(\frac{1}{2}\mathbf{i}\omega I_{N_y} + L_{\beta_2}^{N_y})^T = C. \tag{2.5}$$

For simplicity, we will omit the subscription N_x and N_y of the identity matrix when it is easy to distinguish. Therefore, we reformulate matrix equation (2.5) into the form of (1.1), with

$$A = (\frac{1}{2}\mathbf{i}\omega I + L_{\beta_1}^{N_x})^T = \frac{1}{2}\mathbf{i}\omega I + L_{\beta_1}^{N_x};\ B = (\frac{1}{2}\mathbf{i}\omega I + L_{\beta_2}^{N_y})^T = \frac{1}{2}\mathbf{i}\omega I + L_{\beta_2}^{N_y}. \tag{2.6}$$

Obviously, A and B are both non-Hermitian positive definite matrices [28].

Based on the low-rank property of the right-hand side matrix $C = FG^T$ defined by Eq. (1.1), we need to seek an approximation $\hat{U} \approx U$, such that $\hat{U} = V_a Y V_b^T$ for some matrix Y. Here V_a and V_b have much fewer columns than rows. Then a general projection method for Eq. (1.1) will be considered first.

Given two approximation spaces Range (V_a) and Range (V_b), an approximation $\hat{U} = V_a Y V_b^T$ is determined by requiring that the residual $R = A\hat{U} + \hat{U}B - C$ satisfies $V_a^T R V_b = 0$.

If V_a and V_b have orthogonal columns, by substituting $\hat{U} = V_a Y V_b^T$, we can obtain the reduced matrix equation as

$$(V_a^T A V_a)Y + Y(V_b^T B V_b) = V_a^T F G^T V_b.$$

Denote $H_a = V_a^T A V_a$, $H_b = V_b^T B V_b$, $F_a = V_a^T F$, $G_b = V_b^T G$; then we obtain a matrix equation of small size as

$$H_a Y + Y H_b = F_a G_b^T,$$

which can be efficiently solved by the Bartels–Stewart method. However, different choices of Range (V_a) and Range (V_b) will lead to different approximate solutions. According to [12], for Range (V_a), we can choose the Krylov-Plus-Inverted-Krylov (KPIK) subspace, which is an approximation space generated by powers of A and A^{-1}. Then the space Range (V_a) can be generated as

$$\text{Range}(V_a) = \text{span}([F, A^{-1}F, AF, A^{-2}F, A^2F, A^{-3}F, \cdots]).$$

By the same strategy, we can choose the space for Range (V_b) as

$$\text{Range}(V_b) = \text{span}([G, B^{-1}G, BG, B^{-2}G, B^2G, B^{-3}G, \cdots]).$$

Both of the spaces Range (V_a) and Range (V_b) can be expanded until the approximate solution $\hat{U}$ is sufficiently good. For more details about the KPIK subspace method, we refer to [21].

Therefore, from a computational standpoint, when we are expanding the space in the iterative generation, we have to solve systems with coefficient matrices A and B, which are both large and complex non-Hermitian matrices. According to [13], we will solve those systems by the NPHSS iteration method, and we name the corresponding method an NPHSS-KPIK method. At the end of this section, we will remark on the main steps in the NPHSS-KPIK iteration method for solving the Sylvester equation (1.1) as the following algorithm.

Algorithm 2.1 *(The NPHSS-KPIK iteration method)*
Given tolerances $\varepsilon_{out} > 0$ and $\varepsilon_{in} > 0$, an integer parameter k_1, and set $k = 0$, $m = k_1$. Use the ***NPHSS***

iteration method *to solve* $AZ_a = F$ *and* $BZ_b = G$ *to obtain* $\tilde{Z}_a \approx A^{-1}F$ *until* $\|A\tilde{Z}_a - F\|/\|F\| < \varepsilon_{in}$ *and* $\tilde{Z}_b \approx B^{-1}G$ *until* $\|B\tilde{Z}_b - G\|/\|G\| < \varepsilon_{in}$. *Set* $V_{a,1} = gram\_sh([F, \tilde{Z}_a])$, $V_{a,0} = [\,]$; $V_{b,1} = gram\_sh([G, \tilde{Z}_b])$, $V_{b,0} = [\,]$;

For $j = 1, 2, \cdots, m$, *do Step 1–Step 9:*

Step 1. $V_{a,j} = [V_{a,j-1}, V_{a,j}]$, $V_{b,j} = [V_{b,j-1}, V_{b,j}]$;

Step 2. Set $H_{a,j} = V_{a,j}^T A V_{a,j}$ *and* $F_{a,j} = V_{a,j}F$; $H_{b,j} = V_{b,j}^T A V_{b,j}$ *and* $G_{b,j} = V_{b,j}G$;

Step 3. Solve $H_{a,j}Y_j + Y_j H_{b,j} = F_{a,j}G_{b,j}^T$ *to obtain* Y_j;

Step 4. Compute the upper bound r_j *for the residual norm according to (Theorem 3,[12]);*

Step 5. If $r_j < \varepsilon_{out}$, *then* $\hat{U} = V_{a,j}Y_jV_{b,j}^T$ *and stop. If not convergent, go to Step 8;*

Step 6. Set $V_{a,j}^{(1)}$: *first* s *(here* s *is defined by condition* (A_3) *of Eq. (1.1)) columns of* $V_{a,j}$; $V_{a,j}^{(2)}$: *second* s *columns of* $V_{a,j}$; $V_{b,j}^{(1)}$: *first* s *columns of* $V_{b,j}$; $V_{b,j}^{(2)}$: *second* s *columns of* $V_{b,j}$;

Step 7. Use the ***NPHSS iteration method*** *to obtain the approximations of* $\tilde{Z}_a \approx A^{-1}V_{a,j}^{(2)}$ *and* $\tilde{Z}_b \approx B^{-1}V_{b,j}^{(2)}$ *until converge. Set* $V'_{a,j+1} = [AV_{a,j}^{(1)}, \tilde{Z}_a]$ *and* $V'_{b,j+1} = [BV_{b,j}^{(1)}, \tilde{Z}_b]$;

Step 8. $\hat{V}_{a,j+1} \leftarrow$ *orthogonalize* $V'_{a,j+1}$ *w.r. to* $V_{a,j}$; $\hat{V}_{b,j+1} \leftarrow$ *orthogonalize* $V'_{b,j+1}$ *w.r. to* $V_{b,j}$;

Step 9. Set $V_{a,j+1} = gram\_sh(\hat{V}_{a,j+1})$ *and* $V_{b,j+1} = gram\_sh(\hat{V}_{b,j+1})$, $j = j + 1$, *turn to Step 1.*

end

We remark in the NPHSS-KPIK iteration method that the function *gram_sh* constructs an orthogonal basis and a Hessenberg matrix; for more details, we refer to [21].

3. The local convergence analysis

The theoretical analysis about the convergence property of the KPIK method can be found in [12, 21]. In this section, we only consider the local convergent property of the NPHSS-KPIK iteration method in Algorithm 2.1.

From Algorithm 2.1, we find that, at each iterative step of the NPHSS-KPIK iteration method, one needs to solve the following complex linear system:

$$Kx = b, \tag{3.1}$$

where $K \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$. When the coefficient matrix K is with a dominant indefinite symmetric part, Bai and Qiu proposed a class of splitting minimal residual (SMINRES) and preconditioned SMINRES (PSMINRES) method by making use of the inner/outer iteration technique [6]. When the coefficient matrix K is with a dominant symmetric positive definite part, Axelsson further introduced the nested-splitting conjugate gradient (NSCG) and preconditioned NSCG (PNSCG) schemes, which are a class of nested iteration schemes. These schemes employ the classical conjugate gradient methods as inner iteration to approximate each outer iterate [1]. Those methods are efficient when they are used to solve the linear systems with a dominant indefinite or definite symmetric part.

Furthermore, if we let $W = \frac{1}{2}(K + K^*)$ and $\mathbf{i}T = \frac{1}{2}(K - K^*)$ be the Hermitian parts and skew-Hermitian parts of K, respectively, then we can split the coefficient matrix K as $(\alpha P + W) - (\alpha P - \mathbf{i}T)$, where α is a given positive constant and $P \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix. According to [13], we can

propose the nonalternating preconditioned HSS (NPHSS) iteration method for solving the system (3.1) as the following algorithm:

Algorithm 3.1 *(The NPHSS iteration method [13]) Given an initial guess* $x^{(0)}$, *a positive constant* α, *and a symmetric positive definite matrix* $P \in \mathbb{R}^{n\times n}$, *for* $k = 1, 2, \cdots$, *until* $\{x^{(k)}\}$ *converges, compute*

$$(\alpha P + W)x^{(k+1)} = (\alpha P - \boldsymbol{i}T)x^{(k)} + b,$$

or equivalently

$$x^{(k+1)} = x^{(k)} + (\alpha P + W)^{-1}(b - Kx^{(k)}).$$

The convergence result for the NPHSS iteration method can be given as the following theorem.

Theorem 3.1 *[13] Suppose* α *to be a positive constant and* P *to be a symmetric positive definite matrix; then the iterative matrix of the NPHSS iteration method is*

$$M(P;\alpha) = (\alpha P + W)^{-1}(\alpha P - \boldsymbol{i}T),$$

and its spectral radius $\rho(M(P;\alpha))$ *is bounded by*

$$\delta(\alpha) = \frac{\sqrt{\alpha^2 + \sigma_{\max}^2}}{\alpha + \lambda_{\min}},$$

where $\sigma_{\max}$ *is the maximum singular value of the matrix* $\boldsymbol{i}P^{-1}T$, *and* $\lambda_{\min}$ *is the minimum eigenvalue of the matrix* $P^{-1}W$. *Moreover, if*

$$\alpha > \frac{\sigma_{\max}^2 - \lambda_{\min}^2}{2\lambda_{\min}},$$

then the upper bound $\delta(\alpha) < 1$. *Particularly, if* $\sigma_{\max} \le \lambda_{\min}$, *then the NPHSS iteration method is unconditionally convergent.*

Next, we will give the quasi-optimal α to minimize the upper bound $\delta(\alpha)$.

Theorem 3.2 *[13] Let* $\sigma_{\max}$, $\lambda_{\min}$ *be defined by Theorem 3.1; then the optimal parameter* α *is* $\alpha^* = \frac{\sigma_{\max}^2}{\lambda_{\min}}$ *and the corresponding upper bound for* $\rho(M(P;\alpha))$ *is*

$$\delta(\alpha^*) = \frac{\sigma_{\max}}{\sqrt{\sigma_{\max}^2 + \lambda_{\min}^2}}.$$

Remark 3.3 *According to [1, 6] and based on the splitting* $K = (\alpha P + W) - (\alpha P - \boldsymbol{i}T)$, *we can also solve the linear equations (3.1) approximatively by the minimal residual method or the conjugate gradient (CG) method. As* $\alpha P + W$ *is symmetric positive definite, then, at each step of iteration, a system of linear equations with coefficient matrix* $\alpha P + W$ *can be solved by the CG method. For more details, we refer to [1].*

4. Numerical results

In this section, we use the following example to examine the numerical behaviour of the NPHSS-KPIK iteration method. Our tests are performed in MATLAB R2012a on an Intel Core i7-3770 CPU 3.40 GHz and 8.00 GB of RAM, with machine precision 10^{-16}. When we use the NPHSS iteration method to solve the system at the inner iteration, because the system matrix is symmetric positive definite, then we will solve the system exactly by Cholesky factorization. In all the numerical experiments, the stopping criteria for all the inner iteration method are the maximum iteration counts exceeding by 1000.

The tolerance (ε_{out} in Algorithm 2.1) for the KPIK iteration method is set to be 10^{-8}. The iteration counts (denoted by 'it') containing the average iterations for solving systems A and B, respectively, and the elapsed CPU time to get the numerical solutions of the problems in seconds (denoted by 'CPU') are reported in the tables. The absolute error norm (denoted by 'RES') reported in all the tables is defined by

$$\mathrm{RES} = \|A\hat{U} + \hat{U}B - C\|_2,$$

where $\hat{U}$ is the numerical solution obtained by the corresponding iteration method.

Example 1 *The initial-boundary value problem of two-dimensional time-periodic FDE (2.1) is considered with* $a_x = a_y = 0$, $b_x = b_y = 1$ *and* $T = 1$. *The source term is given by* $f(x, y, t) = 100\sin(10x)\cos(y)e^{\mathbf{i}\omega t}$, *where* ω *is the problem parameter.*

By applying the shifted Grünwald approximations to discretize the fractional spatial derivative and substituting $u(x, y, t) = u(x, y)e^{\mathbf{i}\omega t}$ and $f(x, y, t) = f(x, y)e^{\mathbf{i}\omega t}$ into the above equation, we can obtain the system (2.5) according to Section 2. In order to test the efficiency of the NPHSS-KPIK iteration method, we will rewrite the equation (2.5) as $AU + UB = C$ with $A = \frac{1}{2}\mathbf{i}\omega I + L_{\beta_1}^{N_x}$; $B = \frac{1}{2}\mathbf{i}\omega I + L_{\beta_2}^{N_y}$. The low-rank right-hand side matrix C has the form

$$C = 100\begin{pmatrix} \sin(10h_x) \\ \sin(20h_x) \\ \sin(30h_x) \\ \vdots \\ \sin(10N_xh_x) \end{pmatrix}\begin{pmatrix} \cos(h_y) & \cos(2h_y) & \cos(3h_y) & \cdots & \cos(N_yh_y) \end{pmatrix} := FG^T,$$

where $F \in \mathbb{R}^{N_x \times s}$, $G \in \mathbb{R}^{N_y \times s}$ with $s = 1$.

We will replace the inner iterative method, i.e. NPHSS iteration method in Algorithm 3.1, instead of some other methods, such as GMRES method [19], MINRES method [11], HSS iteration method [5], and MHSS iteration method [4], and the corresponding methods are named GMRES-KPIK method, MINRES-KPIK method, HSS-KPIK method, and MHSS-KPIK method, respectively.

When we use the HSS-KPIK method and the MHSS-KPIK method, the optimal parameters obtained by Bai in [4, 5] are used as

$$\overline{\alpha} = \sqrt{\lambda_{\min}\lambda_{\max}},$$

where $\lambda_{\min}$, $\lambda_{\max}$ are the minimum eigenvalue and maximum eigenvalue of the Hermitian parts of the matrix A or B, respectively. When the NPHSS-KPIK method is used, we set the preconditioner P as an identity

matrix and use the optimal parameter $\alpha^* = \frac{\sigma_{\max}^2}{\lambda_{\min}}$ [13], where $\sigma_{\max}$ and $\lambda_{\min}$ are the maximum singular-values of the skew-Hermitian parts and the maximum eigenvalue of the Hermitian parts, respectively.

We compare the NPHSS-KPIK iteration method with the GMRES-KPIK method, the MINRES-KPIK method, the HSS-KPIK method, and the MHSS-KPIK method according to different mesh grids. We test the problem when the mesh grid is $2^{N_x+1} \times 2^{N_y+1}$, the size of the matrix A is $2^{N_x+1} \times 2^{N_x+1}$, the size of the matrix B is $2^{N_y+1} \times 2^{N_y+1}$, and the size of the unknown matrix U is $2^{N_x+1} \times 2^{N_y+1}$, whose total number of unknowns is $2^{N_x+1} \times 2^{N_y+1}$.

We will report the iteration counts for the NPHSS iteration method for solving the linear systems with coefficient matrix being A and B, and denote the average iteration counts by it_A and it_B, respectively. We test the methods according to different choices of mesh grids and of different derivative parameters, i.e.

$$(\beta_1, \beta_2) \in \{(1.3, 1.3), (1.3, 1.7), (1.7, 1.3), (1.9, 1.9)\}.$$

We remark the results by '-' in Table 1–Table 4, if the iteration counts achieve the maximum iteration count 1000.

From Table 1–Table 4, we can see that the GMRES-KPIK method and the MINRES-KPIK method are independent of the derivative parameters β_1, β_2 and the parameter ω. Furthermore, because of the large and complex properties of the matrices A and B, the GMRES-KPIK method and the MINRES-KPIK method are sensitive to the mesh grids. The iteration counts and CPU time increase rapidly when the mesh grids become small.

The HSS iteration method and the MHSS iteration method are famous for their efficiency and unconditionality convergent properties for general complex systems. Then we can find from Table 1–Table 4 that the HSS iteration method and MHSS iteration method are independent of the parameter ω and are both more efficient than the GMRES-KPIK method and MINRES-KPIK method. However, we can also find that when the derivative parameters β_1 and β_2 are closer to 2, they become less efficient. Furthermore, they both depend greatly on the mesh grids.

However, it can be seen from Table 1–Table 4 that the NPHSS-KPIK method is independent of the mesh grids and it is the most efficient. Furthermore, the NPHSS-KPIK method becomes more efficient when the derivative parameters β_1 and β_2 are closer to 2.

5. Conclusions

In this paper, we derive an NPHSS-KPIK iteration method for the low-rank Sylvester equations arising from the time-periodic fractional diffusion equations. When the right-hand side matrix of the Sylvester equation is low-rank, the Krylov-plus-inverted-Krylov subspace method can be efficiently used to reformulate the original large Sylvester equation into a reduced matrix equation, which is of small size and can be solved directly by Bartels–Stewart and Hessenberg–Schur decomposition method. However, when we use the NPHSS-KPIK method for solving the Sylvester equation arising from the discretization of two-dimensional time-periodic FDEs, we have to solve systems with the coefficient matrix being A and B, which are large and non-Hermitian. According to the special structure of the matrices A and B, we consider the NPHSS iteration method [13]. The main advantage of the NPHSS iteration method for solving linear systems is that one only needs to solve a system with a symmetric positive definite coefficient matrix. Experiments are also used to illustrate the feasibility and the efficiency of the NPHSS-KPIK iteration method. It can be seen from the experiments that the NPHSS-KPIK iteration method is independent of the mesh grids. Furthermore, when the derivative parameters β_1 and β_2 are closer to 2, the NPHSS-KPIK method becomes more efficient.

Table 1. Numerical results for the proposed methods ($\beta_1 = \beta_2 = 1.3$).

ω	N_x+1	N_y+1		GMRES-KPIK	MINRES-KPIK	HSS-KPIK	MHSS-KPIK	NPHSS-KPIK
0.1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(62,32)	(62,32)	(268,169)	(268,169)	(6,6)
			CPU	0.39	0.1	0.009	0.009	**0.001**
			RES	3.02E-05	3.02E-05	3.02E-05	3.02E-05	3.02E-05
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(107,62)	(107,62)	(423,268)	(423,268)	(6,6)
			CPU	1.7	0.29	0.04	0.04	**0.002**
			RES	9.80E-06	7.58E-06	5.59E-06	5.58E-06	5.59E-06
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(175,107)	(175,107)	(665,423)	(666,423)	(6,6)
			CPU	7.48	1.03	0.2	0.17	**0.006**
			RES	1.91E-06	1.91E-06	2.40E-06	2.37E-06	1.91E-06
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(279,175)	(279,175)	(-,665)	(-,666)	(6,6)
			CPU	35.85	9.62	1.34	0.95	**0.02**
			RES	5.69E-06	5.69E-06	9.12E-06	9.12E-06	5.69E-06
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(443,279)	(443,279)	(-,-)	(-,-)	(6,6)
			CPU	191.94	68.58	9.12	7.49	**0.12**
			RES	8.13E-06	1.15E-05	0.0088	0.0088	1.15E-05
1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(62,32)	(62,32)	(268,169)	(270,171)	(16,16)
			CPU	0.39	0.1	0.009	0.008	**0.001**
			RES	3.02E-05	3.02E-05	3.02E-05	3.02E-05	3.02E-05
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(107,62)	(107,62)	(423,268)	(425,270)	(16,16)
			CPU	1.7	0.29	0.04	0.04	**0.002**
			RES	5.76E-06	7.21E-06	5.59E-06	5.58E-06	5.61E-06
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(174,107)	(174,107)	(665,423)	(667,425)	(16,16)
			CPU	7.54	1.02	0.2	0.15	**0.006**
			RES	2.00E-06	2.00E-06	2.16E-06	2.27E-06	1.91E-06
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(279,174)	(279,174)	(-,665)	(-,667)	(16,16)
			CPU	35.95	9.58	1.57	0.96	**0.03**
			RES	5.69E-06	5.67E-06	8.04E-06	8.51E-06	5.69E-06
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(442,279)	(442,279)	(-,-)	(-,-)	(16,16)
			CPU	194.42	70.36	8.98	6.59	**0.13**
			RES	8.34E-06	1.30E-05	0.0092	0.0094	1.15E-05

Table 2. Numerical results for the proposed methods ($\beta_1 = 1.3, \beta_2 = 1.7$).

ω	N_x+1	N_y+1		GMRES-KPIK	MINRES-KPIK	HSS-KPIK	MHSS-KPIK	NPHSS-KPIK
0.1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(62,32)	(62,32)	(268,245)	(268,245)	(6,4)
			CPU	0.39	0.1	0.01	0.01	**0.001**
			RES	5.45E-05	5.45E-05	5.45E-05	5.45E-05	5.45E-05
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(107,64)	(107,64)	(423,440)	(423,440)	(6,4)
			CPU	1.72	0.29	0.07	0.06	**0.002**
			RES	1.88E-05	1.46E-05	1.07E-05	1.07E-05	1.08E-05
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(175,121)	(175,121)	(665,791)	(666,791)	(6,4)
			CPU	8.35	1.13	0.35	0.31	**0.006**
			RES	1.21E-06	1.21E-06	2.86E-06	2.85E-06	1.68E-06
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(279,221)	(279,221)	(-,-)	(-,-)	(6,4)
			CPU	42.57	10.27	2.36	1.86	**0.02**
			RES	4.45E-06	4.44E-06	0.0012	0.0012	3.65E-06
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(443,401)	(443,401)	(-,-)	(-,-)	(6,4)
			CPU	234.14	78.23	9.7	8.3	**0.11**
			RES	1.01E-05	1.04E-05	0.7687	0.7688	1.09E-05
1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(62,32)	(62,32)	(268,169)	(270,246)	(16,8)
			CPU	0.39	0.1	0.009	0.01	**0.001**
			RES	5.45E-05	5.45E-05	5.45E-05	5.45E-05	5.45E-05
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(107,64)	(107,64)	(423,440)	(425,440)	(16,8)
			CPU	1.71	0.29	0.07	0.06	**0.002**
			RES	1.11E-05	1.38E-05	1.08E-05	1.07E-05	1.08E-05
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(174,121)	(174,121)	(665,791)	(667,792)	(16,8)
			CPU	8.45	1.12	0.38	0.31	**0.006**
			RES	1.21E-06	1.21E-06	2.84E-06	2.81E-06	1.40E-06
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(279,221)	(279,221)	(-,-)	(-,-)	(16,8)
			CPU	42.52	10.19	2.36	1.86	**0.03**
			RES	4.44E-06	4.44E-06	0.0013	0.0013	3.65E-06
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(442,401)	(442,401)	(-,-)	(-,-)	(16,8)
			CPU	234.14	78.23	9.7	8.3	**0.12**
			RES	1.01E-05	1.04E-05	0.7653	0.7663	6.63E-06

Table 3. Numerical results for the proposed methods ($\beta_1 = 1.7, \beta_2 = 1.3$).

ω	N_x+1	N_y+1		GMRES-KPIK	MINRES-KPIK	HSS-KPIK	MHSS-KPIK	NPHSS-KPIK
0.1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(62,32)	(62,32)	(440,169)	(440,169)	(4,6)
			CPU	0.41	0.1	0.009	0.009	**0.001**
			RES	1.45E-05	1.45E-05	1.45E-05	1.45E-05	1.45E-05
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(121,62)	(121,62)	(791,268)	(791,268)	(4,6)
			CPU	2.13	0.31	0.04	0.04	**0.002**
			RES	1.91E-05	1.91E-05	1.91E-05	1.89E-05	1.88E-05
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(221,107)	(221,107)	(-,423)	(-,423)	(4,6)
			CPU	11.91	1.28	0.2	0.17	**0.006**
			RES	2.65E-06	2.65E-06	6.21E-04	6.21E-04	2.65E-06
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(401,175)	(401,175)	(-,665)	(-,666)	(4,6)
			CPU	71.62	14.87	1.39	1.08	**0.03**
			RES	6.53E-06	6.53E-06	0.3818	0.3819	6.19E-06
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(725,279)	(725,279)	(-,-)	(-,-)	(4,6)
			CPU	474.45	121.92	9.13	7.49	**0.12**
			RES	1.09E-05	2.20E-05	18.4557	18.4563	1.07E-05
1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(64,32)	(64,32)	(440,169)	(440,171)	(8,16)
			CPU	0.41	0.1	0.009	0.008	**0.001**
			RES	1.45E-05	1.45E-05	1.45E-05	1.45E-05	1.45E-05
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(121,62)	(121,62)	(791,268)	(792,270)	(8,16)
			CPU	2.09	0.32	0.04	0.04	**0.002**
			RES	1.91E-05	1.91E-05	1.89E-05	1.80E-05	1.89E-05
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(221,107)	(221,107)	(-,423)	(-,425)	(8,16)
			CPU	11.8	1.3	0.21	0.15	**0.007**
			RES	2.64E-06	2.65E-06	6.23E-04	6.26E-04	2.64E-06
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(401,174)	(401,174)	(-,665)	(-,667)	(8,16)
			CPU	70.77	14.95	1.41	0.96	**0.03**
			RES	6.53E-06	6.53E-06	0.3834	0.3839	6.53E-06
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(725,279)	(725,279)	(-,-)	(-,-)	(8,16)
			CPU	472.11	122.63	9.2	6.57	**0.13**
			RES	1.08E-05	2.34E-05	18.5259	18.5259	1.07E-05

Table 4. Numerical results for the proposed methods ($\beta_1 = \beta_2 = 1.9$).

ω	N_x+1	N_y+1		GMRES-KPIK	MINRES-KPIK	HSS-KPIK	MHSS-KPIK	NPHSS-KPIK
0.1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(64,32)	(64,32)	(623,325)	(623,325)	(4,4)
			CPU	0.42	0.1	0.01	0.01	**0.001**
			RES	2.55E-04	2.55E-04	2.55E-04	2.55E-04	2.55E-04
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(128,64)	(128,64)	(-,623)	(-,623)	(4,4)
			CPU	2.35	0.33	0.11	0.09	**0.002**
			RES	3.60E-04	3.60E-04	3.59E-04	3.60E-04	3.60E-04
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(253,128)	(253,128)	(-,-)	(-,-)	(4,4)
			CPU	16.58	1.51	0.54	0.45	**0.006**
			RES	1.06E-04	8.67E-05	0.0536	0.0536	8.66E-05
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(491,253)	(491,253)	(-,-)	(-,-)	(4,4)
			CPU	124.46	20.54	2.3	1.85	**0.03**
			RES	6.21E-05	6.21E-05	5.0331	5.0333	4.68E-05
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(950,491)	(949,491)	(-,-)	(-,-)	(4,4)
			CPU	¿1000	208.32	10.06	8.31	**0.12**
			RES	1.38E-05	1.12E-04	83.8967	83.896	1.14E-05
1	2^7	2^6	$(\text{it}_A, \text{it}_B)$	(62,32)	(62,32)	(623,325)	(623,325)	(7,7)
			CPU	0.39	0.1	0.01	0.01	**0.001**
			RES	2.55E-04	2.55E-04	2.55E-04	2.55E-04	2.55E-04
	2^8	2^7	$(\text{it}_A, \text{it}_B)$	(128,64)	(128,64)	(-,623)	(-,623)	(7,7)
			CPU	2.38	0.34	0.11	0.09	**0.002**
			RES	3.60E-04	3.60E-04	3.59E-04	3.60E-04	3.60E-04
	2^9	2^8	$(\text{it}_A, \text{it}_B)$	(253,128)	(253,128)	(-,-)	(-,-)	(7,7)
			CPU	16.63	1.49	0.53	0.46	**0.006**
			RES	8.68E-05	8.67E-05	0.0537	0.0538	8.66E-05
	2^{10}	2^9	$(\text{it}_A, \text{it}_B)$	(491,253)	(491,253)	(-,-)	(-,-)	(7,7)
			CPU	124.43	21.74	2.29	1.86	**0.03**
			RES	6.22E-05	6.22E-05	5.0218	5.0232	3.61E-05
	2^{11}	2^{10}	$(\text{it}_A, \text{it}_B)$	(950,491)	(949,491)	(-,-)	(-,-)	(7,7)
			CPU	¿1000	202.66	9.78	8.54	**0.13**
			RES	8.58E-06	1.13E-04	83.6608	83.6711	1.14E-05

Acknowledgments

The authors would like to thank the anonymous referee for his/her careful reading of the manuscript and useful comments and improvements.

References

[1] Axelsson O, Bai ZZ, Qiu SX. A class of nested iteration schemes for linear systems with a coefficient matrix with a dominant positive definite symmetric part. Numer Algorithms 2004; 35: 351-372.

[2] Bai J, Feng XC. Fractional-order anisotropic diffusion for image denoising. IEEE Trans. Image Process 2007; 16: 2492–2502.

[3] Bai ZZ. On Hermitian and skew-Hermitian splitting iteration methods for continuous Sylvester equations. J Comput Math 2011; 29: 185-198.

[4] Bai ZZ, Benzi M, Chen F. Modified HSS iteration methods for a class of complex symmetric linear systems. Computing 2010; 87: 93-111.

[5] Bai ZZ, Golub GH, Ng MK. Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. SIAM J Matrix Anal Appl 2003; 24: 603-626.

[6] Bai ZZ, Qiu SX. Splitting-MINRES methods for linear systems with the coefficient matrix with a dominant indefinite symmetric part. Math Numer Sinica 2002; 24: 113-128 (in Chinese).

[7] Bartels RH, Stewart GW. Algorithm 432: Solution of the matrix equation AX+XB=C. Comm ACM 1972; 15: 820-826.

[8] Benson DA, Wheatcraft SW, Meerschaert MM. Application of a fractional advection-dispersion equation. Water Res Research 2000; 36: 1403-1412.

[9] Golub G, Nash S, Loan CV. A Hessenberg-Schur method for the problem AX+XB=C. IEEE Trans Automat Control 1979; 24: 909-913.

[10] Gu CQ, Xue HY. A shift-splitting hierarchical identification method for solving Lyapunov matrix equations. Linear Algebra Appl 2009; 430: 1517-1530.

[11] Harman HH, Jones WH. Factor analysis by minimizing residuals (minres). Psychometrika 1966; 31: 351-368.

[12] Jbilou K. Low rank approximate solutions to large Sylvester matrix equations. Appl Math Comput 2006; 177: 365-376.

[13] Li CX, Wu SL. A single-step HSS method for non-Hermitian positive definite linear systems. Appl Math Lett 2015; 44: 26-29.

[14] Magin RL. Fractional Calculus in Bioengineering. New York, NY, USA: Begell House Publishers, 2006.

[15] Meerschaert MM, Tadjeran C. Finite difference approximations for two-sided space-fractional partial differential equations. Appl Numer Math 2006; 56: 80-90.

[16] Oldham KB, Spanier J. The Fractional Calculus. New York, NY, USA: Academic Press, 1974.

[17] Podlubny I, Chechkin A, Skovranek T, Chen YQ, Jara BMV. Matrix approach to discrete fractional calculus II: Partial fractional differential equations. J Comput Phys 2009; 228: 3137-3153.

[18] Raberto M, Scalas E, Mainardi F. Waiting-times and returns in high-frequency financial data: an empirical study. Phys A: Statistical Mechanics and its Applications 2002; 314: 749-755.

[19] Saad Y, Schultz MH. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Statist Comput 1986; 7: 856-869.

[20] Shlesinger MF, West BJ, Klafter J. Lévy dynamics of enhanced diffusion: Application to turbulence. Phys Rev Lett 1987; 58: 1100.

[21] Simoncini V. A new iterative method for solving large-scale Lyapunov matrix equations. SIAM J Sci Comput 2007; 29: 1268-1288.

[22] Smith RA. Matrix equation XA+BX=C. SIAM J Appl Math 1968; 16: 198-201.

[23] Sokolov IM, Klafter J, Blumen A. Fractional kinetics. Phys Today 2002; 55: 48-54.

[24] Starke G, Niethammer W. SOR for AX-XB=C. Linear Algebra Appl 1991; 154: 355–375.

[25] Wachspress EL. Iterative solution of the Lyapunov matrix equation. Appl Math Lett 1988; 1: 87-90.

[26] Wang X, Li WW, Mao LZ. On positive-definite and skew-Hermitian splitting iteration methods for continuous Sylvester equation AX+ XB= C. Comput Math Appl 2013; 66: 2352-2361.

[27] Zaslavsky GM, Stevens D, Weitzner H. Self-similar transport in incomplete chaos. Phys Rev E(3) 1993; 48: 1683.

[28] Zeng ML, Zhang GF. Incomplete circulant and skew-circulant splitting iteration method for time-dependent space fractional diffusion equations. Jpn J Ind Appl Math 2016; 33: 251-268.

[29] Zheng QQ, Ma CF. On normal and skew-Hermitian splitting iteration methods for large sparse continuous Sylvester equations. J Comput Appl Math 2014; 268: 145-154.