7th CSRC Conference Numerical Methods for Fractional-Derivative Problems

On τ matrix-based approximate inverse preconditioning technique for diagonal-plus-Toeplitz linear systems from spatial fractional diffusion equations

Min-Li Zeng[1] Jun-Feng Yang[2] Guo-Feng Zhang[3]

[1]School of Mathematics and Finance, Putian University

[2]Department of Mathematics, Nanjing University

[3]School of Mathematics and Statistics, Lanzhou University

July 26th-29th, 2023

Individual Resume– 曾闽丽

- Education Experience
  - PhD, Lanzhou University, 2012.09-2015.06
  - Master, Lanzhou University, 2003.09-2006.06
  - Bachelor, Lanzhou University, 1999.09-2003.06
- Research Experience
  - Nanjing University, 2021.02-2022.12
  - University of Macau, 2018.12-2019.01/2019.07-2019.08/2020.11
  - Chinese Academy of Sciences, 2019.08
  - University of Birminghan, UK, 2017.07-2018.07
  - University of NewOrleans, USA, 2016.08-2016.12
  - Fujian Normal University, 2015.09-2016.06
- Work Experience
  - School of Mathematics and Finance, Putian University
- Research Interests
  - Algorithms for large-scaled PDE constrained optimization problems
  - Fast solvers for structured discretized systems from FDEs
  - Matrix-splitting based algorithms and preconditioners for large sparse structured linear systems

Outline

Research Background

People use the FDEs to provide an adequate and accurate description for these anomalous diffusion, for example,

- modeling chaotic dynamic of classical conservation systems;
- groundwater contaminant transport;
- turbulent flow;
- applications in biology, finance, image processing, physics and flow in human meniscus;
- etc.

Research Background

A closed-form analytical solution is usually not available, and numerical methods have become an important means for reliably and effectively computing approximate solution for fractional diffusion equations.

There are many research results on numerical methods for FDEs. To obtain the numerical solutions, it usually needs two steps,

- Step 1. Discretization: finite difference methods, finite element methods, finite volume methods, spectral methods, etc.

Research Background

A closed-form analytical solution is usually not available, and numerical methods have become an important means for reliably and effectively computing approximate solution for fractional diffusion equations.

There are many research results on numerical methods for FDEs. To obtain the numerical solutions, it usually needs two steps,

- Step 1. Discretization: finite difference methods, finite element methods, finite volume methods, spectral methods, etc.
- Step 2. Solution of the discrete systems: direct methods, (preconditioned) iteration methods, etc.

Step 1. Discretization

The finite difference method is a powerful tool and widely used to solve the DEs as well as the FDEs in science and engineering, which is also easy to be understood. Meanwhile, the implementation of the finite difference scheme is simple and easy to be put into practice in computer programs. The work on the finite difference method for FDEs are very rich.

---

[1] Meerschaert M M, Tadjeran C. J., Comput. Appl. Math. 172(1)(2004)65-77.
[2] Meerschaert M M, Tadjeran C., Appl. Numer. Math. 56(1)(2006) 80-90.
[3] Wang H, Wang K, Sircar T., J. Comput. Phys. 229(21) (2010) 8095-8104.
[4] Çelik C, Duman M., J. Comput. Phys. 231(4)(2012) 1743-1750.
[5] Sousa E, Li C., Appl. Numer. Math. 90 (2015) 22-37.

Step 1. Discretization

The finite difference method is a powerful tool and widely used to solve the DEs as well as the FDEs in science and engineering, which is also easy to be understood. Meanwhile, the implementation of the finite difference scheme is simple and easy to be put into practice in computer programs. The work on the finite difference method for FDEs are very rich.

Classic finite difference schemes for FDEs:

- Grünwald-Letnikov formula is often unconditionally unstable [1]
- Shifted Grünwald formula is often unconditionally stable [2]
- The shifted finite difference formulas of Grünwald-Letnikov type: the coefficient matrices have Toeplitz-like structure [3]
- Fractional centered difference formula is unconditionally stable [4]
- Weighted average finite difference scheme [5]
- etc.

[1] Meerschaert M M, Tadjeran C. J., Comput. Appl. Math. 172(1)(2004)65-77.
[2] Meerschaert M M, Tadjeran C., Appl. Numer. Math. 56(1)(2006) 80-90.
[3] Wang H, Wang K, Sircar T., J. Comput. Phys. 229(21) (2010) 8095-8104.
[4] Çelik C, Duman M., J. Comput. Phys. 231(4)(2012) 1743-1750.
[5] Sousa E, Li C., Appl. Numer. Math. 90 (2015) 22-37.

Fast solution methods for the discretization systems

Preconditioned iteration methods for one-dimensional (1D) spatial fractional diffusion equations:

- Circulant preconditioner [6]
- Approximate inverse preconditioner and scaled-circulant preconditioner [7,8]
- Banded preconditioners and splitting preconditioner [9,10]
- Structure preserving preconditioners [11]
- Splitting iteration methods [12,13,14]

---

[6] Lei S L, Sun H W., J. Comput. Phys. 242(2013) 715-725.
[7] Pan JY, Ke RH, Ng MK and Sun HW., SIAM J. Sci. Comput. 36(6)(2014) A2698-A2719.
[8] Bai Z Z, Lu K Y, Pan J Y., Numer. Linear Algebra Appl. 2017, 24(4)(2017)e2093.
[9] Lin F R, Yang S W, Jin X Q., J. Comput. Phys. 256 (2014) 109-117.
[10] Lin XL, Ng MK, Sun HW., SIAM J. Matrix Anal. A. 38(4)(2017)1580-1614.
[11] Donatelli M, Mazza M, Serra-Capizzano S., J. Comput. Phys. 307(2016) 262-279.
[12] Bai ZZ, Lu KY, Pan JY., Numer. Linear Algebra Appl. 24(4)(2017) e2093.
[13] Bai ZZ, Lu KY., BIT 59(1)(2019)1-33.
[14] Bai ZZ, Lu KY., Numer. Linear Algebra Appl. 27(1)(2020) e2274.

Fast solution methods for the discretization systems

Preconditioned iteration methods for two-dimensional (2D) spatial fractional diffusion equations:

- Banded preconditioners [15]
- Fast matrix splitting preconditioners [16]
- ADI-based matrix splitting preconditioning method [17]
- Multilevel circulant preconditioner [18]
- τ-preconditioner[19]
- Kronecker product splitting preconditioner[20]
- etc.

---

[15] Jin XQ, Lin FR, Zhao Z., Commun. Comput. Phys. 18(2)(2015)469-488.
[16] Bai ZZ, Lu KY., J. Comput. Phys. 404 (2020) 109117.
[17] Tang SP, Huang YM., Comput. Math. Appl. 144(2023)210-220.
[18] Lei SL, Chen X, Zhang X., E. Asian J. Appl. Math. 6(2)(2016)109-130.
[19] Lin X, Huang X, Ng M K, Sun HW., Numer. Algorithms 92(1)(2023)795-813.
[20] Chen H, Lv W, Zhang T., J. Comput. Phys. 360(2018)1-14.

Research Object

Consider the numerical solutions for spatial fractional diffusion equations (FDEs) of the form

$$\begin{cases} d(x,t)\dfrac{\partial u(x,t)}{\partial t} = \dfrac{\partial u(x,t)}{\partial_+ x^\beta} + \dfrac{\partial u(x,t)}{\partial_- x^\beta} + f(x,t), & (x,t)\in\Omega\times(0,T], \\ u(x,t)=0, & (x,t)\in\partial\Omega\times(0,T], \\ u(x,0)=u_0(x), & x\in\Omega, \end{cases} \tag{1.1}$$

where $d(x,t)$ is a prescribed nonnegative function, the fractional order $\beta\in(1,2)$, $f(x,t)$ is the source term.

Research Object

Here the left Riemann-Liouville (R-L)and the right R-L fractional derivatives $\frac{\partial u(x,t)}{\partial_+ x^\beta}$ and $\frac{\partial u(x,t)}{\partial_- x^\beta}$ for $\Omega = [x_L, x_R]$ are defined in the Grünwald-Letnikov form as[21] :

$$\frac{\partial u(x,t)}{\partial_+ x^\beta} = \lim_{h\to 0^+} \frac{1}{h^\beta} \sum_{k=0}^{\lfloor (x-x_L)/h \rfloor} g_k^{(\beta)} u(x-kh,t),$$

$$\frac{\partial u(x,t)}{\partial_- x^\beta} = \lim_{h\to 0^+} \frac{1}{h^\beta} \sum_{k=0}^{\lfloor (x_R-x)/h \rfloor} g_k^{(\beta)} u(x+kh,t),$$

where $\lfloor x \rfloor$ denotes the floor of x, and the coefficient $g_k^{(\beta)}$ are defined as follows:

$$g_0^{(\beta)} = 1, \qquad g_k^{(\beta)} = \frac{(-1)^k}{k!}\beta(\beta-1)\dots(\beta-k+1), \quad k = 1,2,\dots. \tag{1.2}$$

[21] I. Podlubny., IEEE Trans. Autom. Control, 44(1):208 – 214, 1999.

Discretization

Let

$$u_i^{(m)} = u(x_i, t_m), \quad d_i^{(m)} = d(x_i, t_m) \quad \text{and} \quad f_i^{(m)} = f(x_i, t_m).$$

Using the shifted Grünwald-Letnikov approximations [22, 23]:

$$\frac{\partial^\beta u(x_i, t_m)}{\partial_+ x^\beta} = \frac{1}{h^\beta} \sum_{k=0}^{i+1} g_k^{(\beta)} u(x_{i-k+1}, t_m) + \mathcal{O}(h),$$

$$\frac{\partial^\beta u(x_i, t_m)}{\partial_- x^\beta} = \frac{1}{h^\beta} \sum_{k=0}^{N-i+2} g_k^{(\beta)} u(x_{i+k-1}, t_m) + \mathcal{O}(h),$$

and a standard first-order time difference quotient, we obtain the following implicit finite difference scheme, which is unconditionally stable ($i = 1, 2, \ldots, N, m = 0, 1, 2, \ldots, M-1$):

$$d_i^{(m+1)} \frac{u_i^{(m+1)} - u_i^{(m)}}{\Delta t} - \frac{1}{h^\beta} \sum_{k=0}^{i+1} g_k^{(\beta)} u_{i-k+1}^{(m+1)} - \frac{1}{h^\beta} \sum_{k=0}^{N-i+2} g_k^{(\beta)} u_{i+k-1}^{(m+1)} = f_i^{(m+1)}. \quad (2.1)$$

[22] M. M. Meerschaert and C. Tadjeran., J. Comput. Appl. Math. 172(1):65 – 77, 2004.
[23] M. M. Meerschaert and C. Tadjeran., Appl. Numer. Math., 56(1):80 – 90, 2006.

Discretization

By employing the boundary condition $u_0^{(m)} = u_{N+1}^{(m)} = 0$ and the denotation

$$\mathbf{u}^{(m)} = [u_1^{(m)}, u_2^{(m)}, \ldots, u_N^{(m)}]^T, \quad \mathbf{f}^{(m)} = [f_1^{(m)}, f_2^{(m)}, \ldots, f_N^{(m)}]^T,$$

the numerical scheme (2.1) can be rewritten in matrix form as

$$A^{(m+1)}\mathbf{u}^{(m+1)} = (D^{(m+1)} + T)\mathbf{u}^{(m+1)} = \triangle t\mathbf{f}^{(m+1)} + D^{(m+1)}\mathbf{u}^{(m)}, \quad m = 0, 1, 2, \ldots, M-1, \tag{2.2}$$

where $D^{(m+1)} = \mathrm{diag}(d_1^{(m+1)}, d_2^{(m+1)}, \ldots, d_n^{(m+1)})$ is a diagonal matrix and

$$T = \frac{\triangle t}{h^\beta}(T_\beta + T_\beta^T)$$

with

$$T_\beta = -\begin{pmatrix} g_1^{(\beta)} & g_0^{(\beta)} & 0 & \cdots & 0 & 0 \\ g_2^{(\beta)} & g_1^{(\beta)} & g_0^{(\beta)} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ g_{N-1}^{(\beta)} & \ddots & \ddots & \ddots & & g_0^{(\beta)} \\ g_N^{(\beta)} & g_{N-1}^{(\beta)} & \cdots & \cdots & g_2^{(\beta)} & g_1^{(\beta)} \end{pmatrix}$$

Discretization

It can be easily seen from the previous section that, at each temporal step, one requires to solve the discretized linear system (2.2), whose coefficient matrix $A^{(m+1)} = D^{(m+1)} + T$ is a diagonal-plus-Toeplitz matrix.
Therefore, we focus on an efficient preconditioning technique for the matrix $A \in \mathbb{R}^{N\times N}$ of the form

$$A = D + T, \tag{2.3}$$

where $D \in \mathbb{R}^{N\times N}$ is a diagonal matrix of nonnegative diagonal elements, and $T \in \mathbb{R}^{N\times N}$ is a SPD Toeplitz matrix.

Existing splitting methods and preconditioners

- The DTS Iteration Method and the DCS preconditioner[24]:

$$\begin{cases} (\alpha I + D)u^{k+\frac{1}{2}} = (\alpha I - T)u^k + b, \\ (\alpha I + T)u^{k+1} = (\alpha I - D)u^{k+\frac{1}{2}} + b. \end{cases}$$

$$M_{DCS} = \frac{1}{2\alpha}(\alpha I + D)(\alpha I + C).$$

- The diagonal and circulant or skew-circulant splitting preconditioners[25];
- The lopsided scaled DTS preconditioner[26];
- The dominant Hermitian splitting iteration method[27];
- etc.

---

[24] Z.-Z. Bai, K.-Y. Lu, and J.-Y. Pan, Numer. Linear Algebra Appl., 24(4):e2093, 2018.
[25] K.-Y. Lu, Comput. Appl. Math. 37: 4196 – 4218, 2018.
[26] S.-P. Tang, Y.-M. Huang, Appl Math Lett 131, 108022, 2022.
[27] K.-Y. Lu, D.-X. Xie, F. Chen, G.V. Muratova, Appl. Numer. Math. 164: 15-28, 2021.

Existing iteration methods and preconditioners

Pan, Ke, Ng and Sun introduce the preconditioner P_1 with [28]

$$P_1^{-1} = \sum_{i=1}^{N} e_i e_i^T K_i^{-1}$$

based on the fact $e_i^T A = e_i^T K_i$, where e_i is the ith column of the identity matrix I, $K_i = d_i I + T$, $i = 1, 2, \ldots, N$. One can use a circulant matrix, such as the Strang circulant matrix [29], to approximate the Toeplitz matrix K_i. Therefore, we can obtain the preconditioner P_C with

$$P_C^{-1} = \sum_{i=1}^{N} e_i e_i^T C_i^{-1},$$

where $C_i = d_i I + C, \quad i = 1, 2, \ldots, N$, with C being the Strang circulant approximation of the Toeplitz matrix T.

[28] J.-Y. Pan, R.-H. Ke, M. K. Ng, and H.-W. Sun., SIAM J. Sci. Comput., 36(6):A2698 – A2719, 2014.

[29] R. H. Chan and G. Strang., SIAM J. Sci. Stat. Comput., 10(1):104 – 119, 1989.

τ matrix

Because T is an SPD Toeplitz matrix in the linear system (2.2) [30], we can use the τ matrix $\tau(T)$, which is discussed in [31] to approximate the matrix T.
As the first column of T is $-(2g_1^{(\beta)}, g_0^{(\beta)}+g_2^{(\beta)}, g_3^{(\beta)}, \ldots, g_N^{(\beta)})^T$, then the τ matrix $\tau(T)$ can be obtained by using the Hankel correction[32] :

$$\tau(T) = T - HC(T), \tag{3.1}$$

where $HC(T)$ is the Hankel matrix with the first column and the last column being

$$-(g_3^{(\beta)}, g_4^{(\beta)}, \ldots, g_N^{(\beta)}, 0, 0)^T \quad \text{and} \quad -(0, 0, g_N^{(\beta)}, \ldots, g_4^{(\beta)}, g_3^{(\beta)})^T,$$

respectively.

---

[30] Z.-Z. Bai, K.-Y. Lu, and J.-Y. Pan, Numer. Linear Algebra Appl., 24(4):e2093, 2018.
[31] S. Serra., Math. Comp., 68(226):793 – 803, 1999.
[32] D. Bini and F. Benedetto, In Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures, pages 220 – 223. ACM, 1990.

Main idea

Therefore, we can easily obtain that the first column of the τ matrix $\tau(T)$ is

$$-(2g_1^{(\beta)} - g_3^{(\beta)}, g_0^{(\beta)} + g_2^{(\beta)} - g_4^{(\beta)}, g_3^{(\beta)} - g_5^{(\beta)}, \ldots, g_{N-2}^{(\beta)} - g_N^{(\beta)}, g_{N-1}^{(\beta)}, g_N^{(\beta)})^T. \quad (3.2)$$

It's known that the τ matrix $\tau(T)$ can be diagonalized as,

$$\tau(T) = S_N \Lambda S_N, \quad S_N = \left(\sqrt{\frac{2}{N+1}} \cdot \sin \frac{\pi ij}{N+1}\right), \quad i,j = 1,2,\ldots,N,$$

and

$$\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N),$$

where $\lambda_i = \frac{\sum_{j=1}^{N} \tau_j \sin(j\xi_i)}{\sin \xi_i}$ with $(\tau_1, \tau_2, \ldots, \tau_N)^T$ being the first column of $\tau(T)$ and $\xi_i = \frac{\pi i}{N+1}$, S_N is the discrete sine transform matrix of order N.

τ matrix

Obviously, $T_j = d_j I + \tau(T)$ is also a τ matrix. Replacing the SPD Toeplitz matrix T by the τ-matrix T_i, we obtain the τ matrix-based preconditioner P_2 with

$$P_2^{-1} = \sum_{i=1}^{N} e_i e_i^T T_i^{-1}. \tag{3.3}$$

Consider a small number $l << N$ of values $\{\tilde{x}_t\}_{t=1}^{l} \subset \{x_i\}_{i=1}^{N}$ such that it covers most of the range of values of $\{x_i\}_{i=1}^{N}$.
Define the function $q_k(x) = \frac{1}{\lambda_k + d(x)}$, where $\lambda_k \in sp(\tau(T)) = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$, $k = 1, 2, \ldots, N$.
Let

$$p_k(x) = \phi_1(x) q_k(\tilde{x}_1) + \phi_2(x) q_k(\tilde{x}_2) + \ldots + \phi_l(x) q_k(\tilde{x}_l) = \sum_{s=1}^{l} \phi_s(x) q_k(\tilde{x}_s)$$

be the piecewise linear interpolation for $q_k(x)$ based on the $l (l << N)$ points.

τ matrix based approximate inverse preconditioner

Substituting $T_j = S\Lambda_j S$, $j = 1, 2, \ldots, N$, where S is the discrete sine transform matrix and Λ_j is a diagonal matrix, whose diagonals are eigenvalues of T_j, follows the interpolation to approximate T_j^{-1}:

$$
\begin{aligned}
T_j^{-1} = S\Lambda_j^{-1}S &= S\mathrm{diag}(q_1(x_j), q_2(x_j), \ldots, q_N(x_j))S \\
&\approx S\mathrm{diag}(p_1(x_j), p_2(x_j), \ldots, p_N(x_j))S \\
&= S\mathrm{diag}(\sum_{s=1}^{l} \phi_s(x_j)q_1(\tilde{x}_s), \sum_{s=1}^{l} \phi_s(x_j)q_2(\tilde{x}_s), \ldots, \sum_{s=1}^{l} \phi_s(x_j)q_N(\tilde{x}_s))S \\
&= S\mathrm{diag}(\sum_{s=1}^{l} r_{s,j}q_1(\tilde{x}_s), \sum_{s=1}^{l} r_{s,j}q_2(\tilde{x}_s), \ldots, \sum_{s=1}^{l} r_{s,j}q_N(\tilde{x}_s))S \\
&= S\sum_{s=1}^{l} r_{s,j}\mathrm{diag}(q_1(\tilde{x}_s), q_2(\tilde{x}_s), \ldots, q_N(\tilde{x}_s))S \\
&\triangleq S(\sum_{s=1}^{l} r_{s,j}\tilde{\Lambda}_s^{-1})S,
\end{aligned}
$$

where $r_{s,j} = \phi_s(x_j)$.

τ matrix based approximate inverse preconditioner

By substituting the approximation of T_i^{-1} from above into (3.3), it follows the approximation of P_2^{-1} as

$$\begin{aligned} P_3^{-1} &= \sum_{i=1}^{N} e_i e_i^T S(\sum_{s=1}^{l} r_{s,i}\tilde{\Lambda}_s^{-1})S \\ &= \sum_{s=1}^{l}(\sum_{i=1}^{N} e_i e_i^T r_{s,i})S\tilde{\Lambda}_s^{-1})S \\ &= \sum_{s=1}^{l} \Phi_s S\tilde{\Lambda}_s^{-1}S, \end{aligned}$$

where

$$\Phi_s = \mathrm{diag}(\varphi_s(x_1), \varphi_s(x_2), \ldots, \varphi_s(x_N))$$

and

$$\tilde{\Lambda}_s^{-1} = \mathrm{diag}(q_1(\tilde{x}_s), q_2(\tilde{x}_s), \ldots, q_N(\tilde{x}_s)),$$

$s = 1, 2, \ldots, l$, are diagonal matrices.

Spectral properties of the preconditioned matrix $P_3^{-1}A$

Definition

Let $A=(a_{i,j})_{i,j\in I}$ be a matrix, where $I=\mathbb{Z},\mathbb{N}$ or $\{1,2,\ldots,N\}$, then we say A belongs to the class $\mathcal{L}_s$, if

$$|a_{i,j}|\leq\frac{c}{(1+|i-j|)^s}$$

for $s>1$, and some constant $c>0$.

Lemma

It holds[a]*:*

$$g_k^{(\beta)}=(1-\frac{\beta+1}{k})g_{k-1}^{(\beta)},\ k=1,2,\ldots,$$
$$g_0^{(\beta)}=1,\ g_1^{(\beta)}=-\beta<0,\ 1>g_2^{(\beta)}>g_3^{(\beta)}>\ldots>0\quad\textit{and}$$
$$\sum_{k=0}^{\infty}g_k^{(\beta)}=0,\quad\sum_{k=0}^{m}g_k^{(\beta)}<0,\quad 1\leq m<\infty.$$

Spectral properties of the preconditioned matrix $P_3^{-1}A$

Lemma

$$g_k^{(\beta)} = \frac{1}{\Gamma(-\beta)k^{\beta+1}}(1+\mathcal{O}(\frac{1}{k})),$$

where $\Gamma(x)$ *is the Gamma function*[a].

[a] H. Wang, K.-X. Wang, and T. Sircar., J. Comput. Phys., 229(21):8095 – 8104, 2010.

Theorem

$$|L_{m,n}| \leq \frac{c_0}{(1+|m-n|)^{\beta+1}},$$

where c_0 *is a positive constant and* $L = (L_{m,n})_{m,n\in I}$ *with* $I = \{1,2,\ldots,N\}$ *can be* T, A, K_i, T_β, T^{-1}, A^{-1}, K_i^{-1} *and* T_β^{-1}[a].

[a] S. Jaffard., volume 7, pages 461 – 476. Elsevier, 1990.

Spectral properties of the preconditioned matrix $P_3^{-1}A$

Lemma

Let $\beta \in (1,2)$ and $L \in \mathscr{L}_{\beta+1}$. Then $\exists$ a constant ϖ s.t. $\|L\|_\infty \leq \varpi$.

Theorem

Let $T_i = d_i I + \tau(T)$ be defined previously, then we have

$$\|T_i^{-1}\|_\infty \leq \frac{1}{c_0},$$

where $c_0 = \min_{1\leq i\leq N}\{d_i + \frac{\triangle t}{h^\beta}\cdot \min\{g_2^{(\beta)}, g_{N-k+2}^{(\beta)}\}\}$.

In the following of this section, we will concentrate on the spectral properties of the preconditioned $P_3^{-1}A$. As

$$P_3^{-1} - A^{-1} = P_3^{-1} - P_2^{-1} + P_2^{-1} - P_1^{-1} + P_1^{-1} - A^{-1},$$

then we will focus on the properties of $P_3^{-1} - P_2^{-1}$, $P_2^{-1} - P_1^{-1}$ and $P_1^{-1} - A^{-1}$.

The approximation property of P_1 and A

Theorem

Suppose $d(x) \in C^1[x_L, x_R]$. For any given $\varepsilon > 0$, $\exists$ a constant c_1 and $\exists$ an integer N_1, such that $l \geq N_1$ and

$$||P_1^{-1} - A^{-1}||_\infty \leq c_1 \max_{1 \leq i \leq N} \triangle(x_i, l) + \varepsilon,$$

where $\triangle(x_i, l) = \max_{i-l \leq k \leq i+l} |x_k - x_i| = (l-1)h$.

The approximation property of P_2 and P_1

Theorem

The approximation P_2^{-1} to P_1^{-1} satisfies

$$P_2^{-1} - P_1^{-1} = E_2 + F_2,$$

where E_2 and F_2 are of small norm and of low rank, respectively, i.e., $||E_2|| < c_2 \cdot \varepsilon$ and $rank(F_2) \leq 4\varsigma$.

The approximation property of P_2 and A

Theorem

Let P_2 and A be defined previously, then $\exists N_2$, such that for $N > N_2$, it holds

$$P_2^{-1} - A^{-1} = E_{P_2} + F_{P_2},$$

where E_{P_2} and F_{P_2} are of small norm and of low rank, respectively.

The approximation property of P_3 and P_2

Theorem

Assume $l \ll N$, then we can rewrite P_3^{-1}, P_2^{-1} and $P_3^{-1} - P_2^{-1}$ as $X + Y + Z$ with X of off-diagonal decay property, Y of a small norm matrix, and Z of a low rank matrix.

The approximation property of P_3 and A

Theorem

Suppose ϖ is sufficiently small and $l \ll N$. Denote by $\varpi = \max_{1\le i\le N} \max_{1\le j\le N}\{|p_{\lambda_j}(d_i) - q_{\lambda_j}(d_i)|\}$. Then for any given $\varepsilon > 0$, $\exists N_3 > 0$ (independent of N) such that

$$P_3^{-1} - P_2^{-1} = E_3 + F_3,$$

where E_3 and F_3 satisfy $||E_3||_\infty \le \varpi(2N_3+1)+\epsilon$ and of a low rank matrix, respectively.

Theorem

Let P_3 and A be defined previously, then $\exists$ an integer N_3, such that for $N > N_3$, it holds

$$P_3^{-1} - A^{-1} = E_{P_3} + F_{P_3},$$

where E_{P_3} and F_{P_3} are of small norm and of low rank, respectively.

Experimental settings

The preconditioners used in our experiments are the identity matrix preconditioner (denoted as 'I'), the preconditioner P_{DCS}, which is defined as

$$P_{DCS} = \frac{1}{2\alpha}(\alpha I + D)(\alpha I + C),$$

and the preconditioner B_3, which is defined as

$$B_3^{-1} = \sum_{j=1}^{l} \Phi_j F \tilde{\Lambda}_j^{-1} F^*,$$

where

$$\Phi_j = \mathrm{diag}(\varphi_j(x_1), \varphi_j(x_2), \ldots, \varphi_j(x_N))$$

with $\varphi_j(x_i)(i = 1, 2, \ldots, N)$ being the interpolation coefficients and $\tilde{\Lambda}_j^{-1} = \mathrm{diag}(q_1(\tilde{x}_j), q_2(\tilde{x}_j), \ldots, q_N(\tilde{x}_j))$, $j = 1, 2, \ldots, l$.

Experimental settings

- The circulant matrix corresponding to the preconditioners B_3 and P_{DCS} is chosen as the Strang's circulant approximations of T.
- To be accordance with the preconditioner B_3, we choose the interpolation nodes in P_3 as

$$\tilde{x}_j = \lfloor \frac{N-1}{\tilde{l}-1} \rfloor \cdot j + 1, \quad j = 0, 1, 2, \ldots, \tilde{l} - 1,$$

  where $\lfloor x \rfloor$ denotes the floor of x.
- All the tests are performed in MATLAB R2017a [version 9.2.0.538062] in double precision, on a personal computer with 2.40GHz central processing unit (Intel(R) Core(TM) 2 Duo CPU), 4.00 GB memory and Windows 64-bit operating system.
- The initial guess is chosen to be zero vector and the iteration is terminated once the current iterate $u^{(k)}$ satisfies

$$\mathrm{RES} = \frac{||b - Au^{(k)}||_2}{||b||_2} < 10^{-6}.$$

Example 1

Example

Consider the FDE (1.1) with the exact solution $u(x,t) = t^2 x^4 (2-x)^4$, $[x_L, x_R] = [0,2]$, *and* $T = 1$. *The diffusion coefficient is given by*

$$d(x,t) = \frac{e^{-(0.8x+\kappa)}}{(1+t)},$$

and the source term is

$$f(x,t) = 2tx^4(2-x)^4 d(x,t) - \sigma_\beta t^2 \sum_{i=5}^{9} \frac{q_i \Gamma(i)(x^{i-1-\beta}) + (2-x)^{i-1-\beta})}{\Gamma(i-\beta)},$$

$$q_5 = 16, q_6 = -32, q_7 = 24, q_8 = -8, q_9 = 1.$$

The parameter κ *is a given constant and* $\sigma_\beta = -\frac{1}{2\cos(\frac{\pi\beta}{2})} > 0$.

Numerical results

Table: Numerical results for preconditioned GMRES iteration methods for Example 1 ($\kappa = 12, \beta = 1.2$).

	Pre.:	I	P_{DCS}	B_3		P_3	
N				$l=20$	$l=24$	$l=20$	$l=24$
2^8	IT	116	6	6	6	4	**4**
	CPU	0.05	0.008	0.008	0.008	0.008	**0.008**
2^9	IT	186	7	6	6	4	**4**
	CPU	0.14	0.02	0.02	0.02	0.02	**0.02**
2^{10}	IT	290	7	7	7	5	**5**
	CPU	0.44	0.03	0.03	0.04	0.03	**0.03**
2^{11}	IT	444	7	7	7	5	**5**
	CPU	1.8	0.06	0.07	0.07	0.06	**0.06**
2^{12}	IT	677	7	7	7	5	**5**
	CPU	150.41	0.26	0.31	0.38	0.33	**0.28**
2^{13}	IT	1029	7	7	7	5	**5**
	CPU	492.34	0.56	0.65	0.65	0.61	**0.62**

Numerical results

Table: Numerical results for preconditioned GMRES iteration methods for Example 1 ($\kappa = 1, \beta = 1.2$).

	Pre.:	I	P_{DCS}	B_3		P_3	
N				$l=20$	$l=24$	$l=20$	$l=24$
2^8	IT	57	8	6	6	4	**4**
	CPU	0.02	0.008	0.008	0.008	0.008	**0.008**
2^9	IT	66	8	6	6	4	**4**
	CPU	0.06	0.02	0.03	0.03	0.02	**0.02**
2^{10}	IT	74	8	6	6	4	**4**
	CPU	0.11	0.04	0.04	0.04	0.03	**0.03**
2^{11}	IT	78	8	6	6	4	**3**
	CPU	0.33	0.06	0.08	0.08	0.06	**0.06**
2^{12}	IT	82	7	5	5	3	**3**
	CPU	18.49	0.31	0.31	0.31	0.22	**0.22**
2^{13}	IT	84	7	5	5	3	**3**
	CPU	43.81	0.61	0.62	0.62	0.61	**0.61**

Numerical results

Table: Numerical results for preconditioned GMRES iteration methods for Example 1 ($\kappa = 0.1, \beta = 1.8$).

	Pre.:	I	P_{DCS}	B_3		P_3	
N				$l=20$	$l=24$	$l=20$	$l=24$
2^8	IT	133	8	7	7	5	**4**
	CPU	0.05	0.008	0.008	0.008	0.008	**0.008**
2^9	IT	199	9	7	7	4	**4**
	CPU	0.18	0.02	0.03	0.03	0.02	**0.02**
2^{10}	IT	300	8	6	6	4	**4**
	CPU	0.45	0.03	0.03	0.04	0.03	**0.03**
2^{11}	IT	431	8	6	6	4	**4**
	CPU	1.76	0.07	0.07	0.08	0.06	**0.06**
2^{12}	IT	594	7	6	6	4	**3**
	CPU	126.22	0.31	0.32	0.32	0.23	**0.23**
2^{13}	IT	797	7	5	5	3	**3**
	CPU	387.91	0.56	0.65	0.65	0.61	**0.61**

Example 2

Example

Consider the 2D fractional diffusion equation

$$\begin{cases} d(x,y,t)\dfrac{\partial u(x,y,t)}{\partial t} - \dfrac{\partial^{\beta_1} u(x,y,t)}{\partial_+ x^{\beta_1}} - \dfrac{\partial^{\beta_2} u(x,y,t)}{\partial_+ y^{\beta_2}} = f(x,y,t), \quad (x,y,t) \in \Omega \times (0,T], \\ u(x,y,t) = 0, \quad (x,y,t) \in \partial\Omega \times (0,T], \\ u(x,y,0) = u_0(x,y), \quad (x,y) \in \bar{\Omega}. \end{cases}$$

with the coefficient function being

$$d(x,y,t) = \frac{e^{-(\sin(40x)+s)(\sin(40y)+s)}}{(2+t+e^{\sin(t)})}$$

Example 2

Example

and the source term being

$$\begin{aligned} f(x,y,t) =& 2tx^4(2-x)^4y^4(2-y)^4d(x,y,t) \\ & -\sigma_{\beta_1}t^2y^4(2-y)^4\sum_{i=5}^{9}\frac{q_i\Gamma(i)(x^{i-1-\beta_1}+(2-x)^{i-1-\beta_1})}{\Gamma(i-\beta_1)} \\ & -\sigma_{\beta_2}t^2y^4(2-y)^4\sum_{i=5}^{9}\frac{q_i\Gamma(i)(x^{i-1-\beta_2}+(2-x)^{i-1-\beta_2})}{\Gamma(i-\beta_2)}, \end{aligned}$$

where $q_5 = 16$, $q_6 = -32$, $q_7 = 24$, $q_8 = -8$, $q_9 = 1$. *The exact solution of this equation is* $u(x,y,t) = t^2x^4(2-x)^4y^4(2-y)^4$ *and* $\Omega = [x_L, x_R]\times[y_D, y_U] = [0,2]\times[0,2]$ *and* $T = 1$. $\sigma_{\beta_1} = -\frac{1}{2\cos(\frac{\pi\beta_1}{2})}$ *and* $\sigma_{\beta_2} = -\frac{1}{2\cos(\frac{\pi\beta_2}{2})}$.

Settings

We report the iteration counts and computing times of the proposed methods with respect to the following choices of β_1 and β_2:
Case I: (β_1, β_2)=(1.2, 1.2),
Case II: (β_1, β_2)=(1.2, 1.5),
Case III: (β_1, β_2)=(1.2, 1.8),
Case IV: (β_1, β_2)=(1.5, 1.5),
Case V: (β_1, β_2)=(1.5, 1.8), and
Case VI: (β_1, β_2)=(1.8, 1.8).
Besides, the parameters s in coefficients are chosen as $s = 5$ and $s = 1$.

Numerical results

Table: Numerical results.

			$s=5$				$s=1$			
Case	N	Pre.:	I	P_{DCS}	B_3	P_3	I	P_{DCS}	B_3	P_3
I	2^5	IT	58	14	14	**5**	58	15	16	**7**
		CPU	0.17	0.12	0.12	**0.05**	0.17	0.12	0.13	**0.05**
	2^6	IT	93	15	15	**5**	63	18	18	**8**
		CPU	2.35	1.05	1.05	**0.29**	2.35	1.36	1.36	**0.41**
	2^7	IT	145	18	18	**6**	79	18	20	**10**
		CPU	19.68	10.36	10.36	**3.63**	19.68	10.36	11.23	**5.39**
II	2^5	IT	70	17	17	**5**	70	17	17	**5**
		CPU	0.22	0.13	0.13	**0.05**	0.22	0.13	0.13	**0.05**
	2^6	IT	117	20	20	**5**	93	20	21	**8**
		CPU	3.18	1.52	1.53	**0.29**	3.18	1.52	1.63	**0.41**
	2^7	IT	196	23	23	**6**	129	23	25	**9**
		CPU	22.77	12.73	12.73	**3.63**	22.77	12.73	14.21	**4.98**
III	2^5	IT	96	21	21	**4**	96	21	20	**5**
		CPU	0.32	0.17	0.17	**0.04**	0.32	0.17	0.17	**0.04**
	2^6	IT	178	26	26	**5**	156	26	26	**7**
		CPU	5.59	1.96	1.96	**0.29**	5.59	1.96	1.96	**0.34**
	2^7	IT	332	31	31	**5**	253	32	34	**8**
		CPU	41.59	19.64	19.64	**2.93**	41.59	20.73	21.46	**4.47**
IV	2^5	IT	59	14	14	**5**	59	15	15	**5**
		CPU	0.23	0.12	0.12	**0.05**	0.23	0.13	0.13	**0.05**
	2^6	IT	101	16	15	**5**	90	18	18	**7**
		CPU	2.83	1.15	1.05	**0.29**	2.83	1.36	1.36	**0.33**
	2^7	IT	171	19	19	**5**	137	23	22	**8**
		CPU	20.73	10.93	10.93	**2.93**	20.73	12.73	12.51	**4.47**
V	2^5	IT	77	18	18	**5**	77	18	17	**5**
		CPU	0.29	0.14	0.14	**0.05**	0.29	0.14	0.13	**0.05**
	2^6	IT	145	21	21	**5**	136	22	21	**6**
		CPU	4.67	1.63	1.63	**0.29**	4.67	1.71	1.63	**0.31**
	2^7	IT	267	24	24	**5**	227	27	27	**7**
		CPU	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Example 3

Example

We consider the 2D spatial fractional diffusion equation (1.1) defined on the domain $[0,2]\times[0,2]\times(0,1)$*, in which*

$$d(x_1,x_2,t) = |\sin(2\pi x_1)|\cosh(ax_2+b\pi)$$

and the source term is

$$f(x,t) = 2tx_1x_2(1-x_1)(1-x_2)d(x_1,x_2,t) - (t^2+0.01)[x_2(1-x_2)g(x_1,\beta_1) + x_1(1-x_1)g(x_2,\beta$$

with

$$g(\varsigma,\beta) = \frac{\varsigma^{1-\beta}+(1-\varsigma)^{1-\beta}}{\Gamma(2-\beta)} - \frac{2*\varsigma^{2-\beta}+2*(1-\varsigma)^{2-\beta}}{\Gamma(3-\beta)},$$

where $\Gamma(\cdot)$ *is the Gamma function.*

Example 3

Example

In addition, the initial condition is chosen as

$$u(x_1, x_2, 0) = 0.01x_1x_2(1 - x_1)(1 - x_2).$$

The true solution to the corresponding FDE is given by $u(x_1, x_2, t) = (t^2 + 0.01)x_1x_2(1 - x_1)(1 - x_2)$.

Numerical results

Table: Results

Case		Pre.:	$a=8$ P_{DCS}	B_3	P_3	$a=1$ P_{DCS}	B_3	P_3	$a=0.1$ P_{DCS}	B_3	P_3
I	2^5	IT	12	12	**8**	10	12	**8**	8	10	**7**
		CPU	0.09	0.09	**0.05**	0.08	0.09	**0.05**	0.05	0.08	**0.05**
	2^6	IT	14	14	**9**	11	13	**8**	10	11	**7**
		CPU	0.94	0.94	**0.48**	0.61	0.83	**0.41**	0.49	0.51	**0.34**
	2^7	IT	17	19	**12**	12	16	**9**	11	14	**9**
		CPU	9.82	10.93	**6.31**	6.33	8.93	**4.98**	5.86	8.03	**4.98**
II	2^5	IT	13	12	**8**	11	12	**7**	10	11	**7**
		CPU	0.09	0.09	**0.05**	0.09	0.09	**0.05**	0.08	0.09	**0.05**
	2^6	IT	16	13	**8**	15	14	**7**	12	12	**7**
		CPU	1.15	0.83	**0.41**	1.05	0.94	**0.35**	0.66	0.66	**0.34**
	2^7	IT	19	16	**10**	17	16	**8**	15	14	**9**
		CPU	10.93	8.93	**5.59**	9.28	8.93	**4.46**	8.59	8.03	**4.98**
III	2^5	IT	17	14	**8**	15	16	**6**	14	14	**6**
		CPU	0.13	0.12	**0.05**	0.12	0.13	**0.04**	0.12	0.12	**0.04**
	2^6	IT	21	16	**8**	17	19	**7**	17	17	**6**
		CPU	1.63	1.15	**0.41**	1.28	1.49	**0.34**	1.16	1.17	**0.31**
	2^7	IT	27	20	**9**	22	23	**8**	22	20	**8**
		CPU	15.23	11.23	**4.98**	12.34	12.73	**4.46**	12.34	11.23	**4.46**
IV	2^5	IT	14	13	**8**	12	13	**8**	10	12	**7**
		CPU	0.12	0.1	**0.05**	0.09	0.1	**0.05**	0.08	0.09	**0.05**
	2^6	IT	17	16	**8**	15	15	**7**	11	14	**7**
		CPU	1.28	1.15	**0.41**	1.05	1.05	**0.34**	0.61	0.94	**0.34**
	2^7	IT	22	18	**9**	16	19	**7**	13	16	**7**
		CPU	12.49	10.36	**4.96**	8.93	10.92	**3.94**	7.63	8.93	**3.94**
V	2^5	IT	15	14	**7**	13	15	**7**	12	13	**7**
		CPU	0.13	0.12	**0.05**	0.11	0.13	**0.05**	0.09	0.11	**0.05**
	2^6	IT	19	16	**8**	15	17	**7**	14	15	**7**
		CPU	1.49	1.15	**0.41**	1.05	1.28	**0.35**	0.94	1.05	**0.35**
	2^7	IT	22	18	**8**	18	21	**7**	15	18	**7**

Eigenvalues Distribution

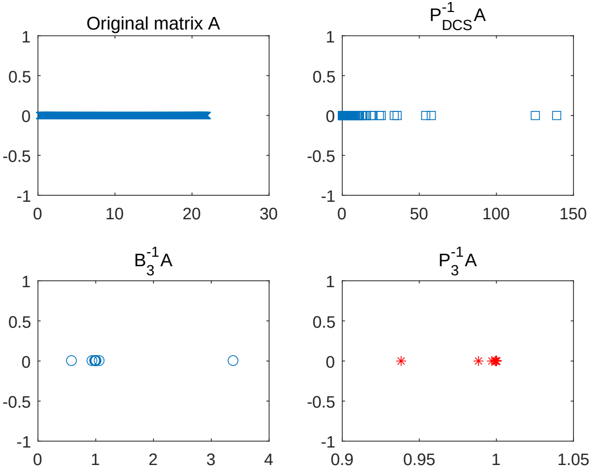

Figure: $\kappa = 0.1$, $\beta = 1.2$

Eigenvalues Distribution

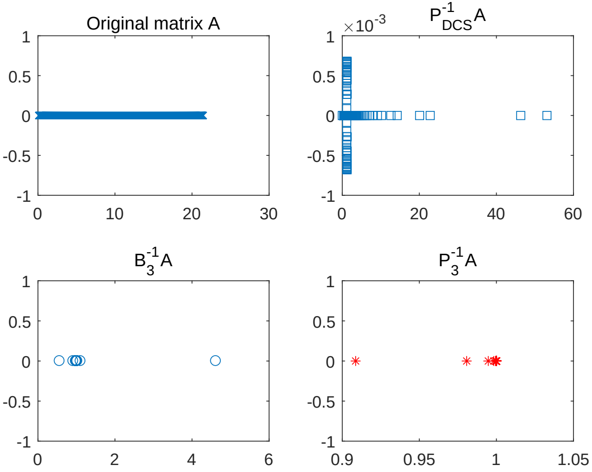

Figure: $\kappa = 1$, $\beta = 1.2$

Eigenvalues Distribution

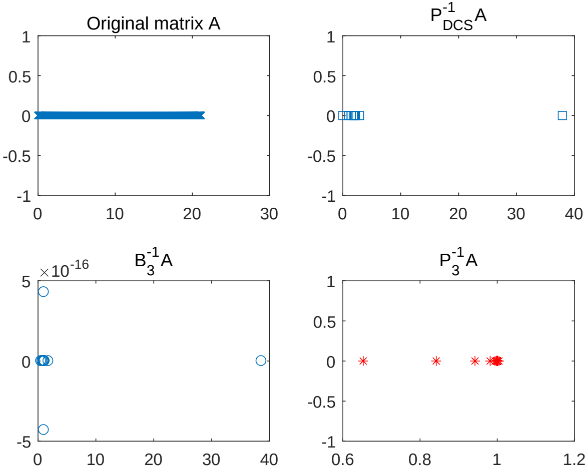

Figure: $\kappa = 12$, $\beta = 1.2$

Eigenvalues Distribution

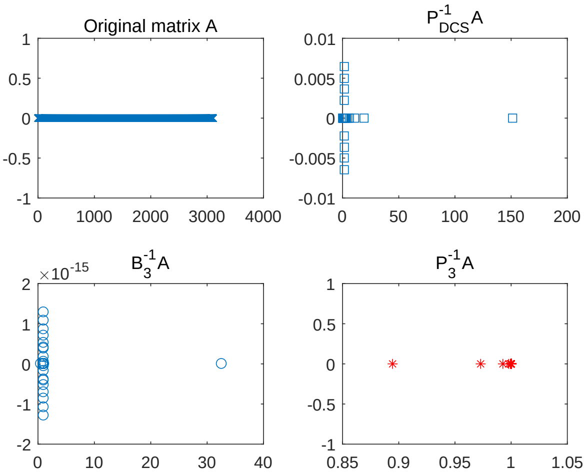

Figure: $\kappa = 0.1$, $\beta = 1.8$

Eigenvalues Distribution

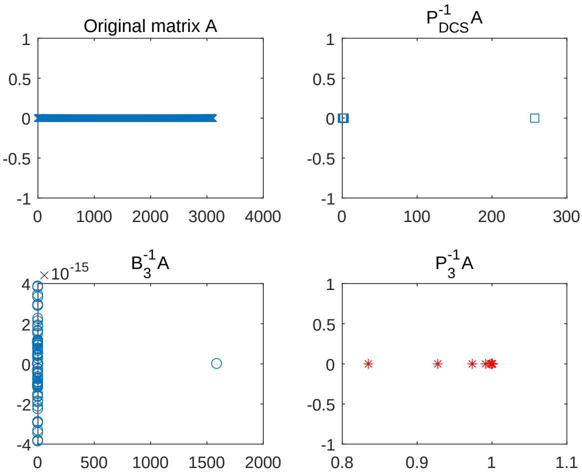

Figure: $\kappa = 12$, $\beta = 1.8$

Eigenvalues Distribution

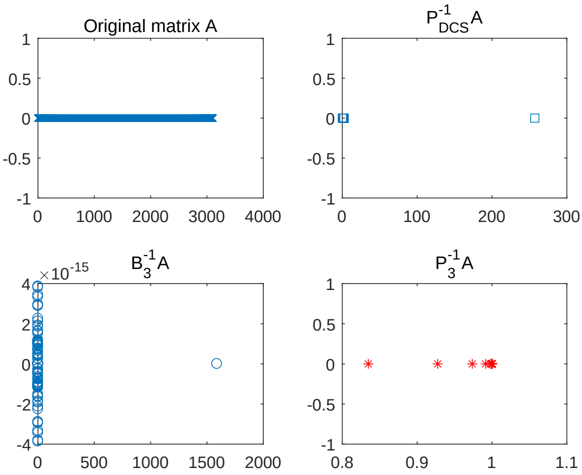

Figure: $\kappa = 12$, $\beta = 1.8$

Concluding remarks

- We propose a TAI preconditioning technique for solving the discretized linear systems with diagonal-plus-Toeplitz coefficient matrix arising from the fractional diffusion equations.
- By approximating the inverses of symmetric positive and definite Toeplitz matrix with a τ matrix and combining them row-by-row, we obtain a new preconditioning technique.
- Theoretical analysis shows that when the new preconditioners are used, the eigenvalues of the preconditioned matrices are tightly clustered around unity.
- We implement the numerical examples to show the efficiency of the proposed preconditioning technique.

Thank you for your attention!

感谢聆听!

欢迎大家批评与指正!

Thanks!