A preconditioning technique for an all-at-once system from Volterra subdiffusion equations with graded time steps

Yong-Liang Zhao[a], Xian-Ming Gu[b,*], Alexander Ostermann[c,*]

[a]School of Mathematical Sciences,
University of Electronic Science and Technology of China,
Chengdu, Sichuan 611731, P.R. China
[b]School of Economic Mathematics/Institute of Mathematics,
Southwestern University of Finance and Economics,
Chengdu, Sichuan 611130, P.R. China
[c]Department of Mathematics, University of Innsbruck,
Technikerstraße 13, Innsbruck 6020, Austria

Abstract

Volterra subdiffusion problems with weakly singular kernel describe the dynamics of subdiffusion processes well. The graded $L1$ scheme is often chosen to discretize such problems since it can handle the singularity of the solution near $t = 0$. In this paper, we propose a modification. We first split the time interval $[0, T]$ into $[0, T_0]$ and $[T_0, T]$, where T_0 $(0 < T_0 < T)$ is reasonably small. Then, the graded $L1$ scheme is applied in $[0, T_0]$, while the uniform one is used in $[T_0, T]$. Our all-at-once system is derived based on this strategy. In order to solve the arising system efficiently, we split it into two subproblems and design two preconditioners. Some properties of these two preconditioners are also investigated. Moreover, we extend our method to solve semilinear subdiffusion problems. Numerical results are reported to show the efficiency of our method.

Keywords: Variable time steps, All-at-once discretization, Parallel-in-time preconditioning, Krylov subspace methods, Semilinear subdiffusion equations

1. Introduction

Anomalous diffusion phenomena are common in various complex systems such as amorphous semiconductors [1], filled polymers [2], porous systems [3] and turbulent plasma [4]. Generally, they do not have the Markovian stochastic property and cannot be simulated by the classical diffusion equations arising from Fick's law. One of the widely used approaches for modelling anomalous diffusion are fractional diffusion equations [5–8]. Numerous studies about fractional partial differential equations can be found in [9–17] and the references therein.

---

*Corresponding authors
Email addresses: `ylzhaofde@sina.com` (Yong-Liang Zhao), `guxianming@live.cn` (Xian-Ming Gu), `alexander.ostermann@uibk.ac.at` (Alexander Ostermann)

In this work, we are interested in solving the subdiffusion equation:

$$
\begin{cases}
\int_0^t \xi_{1-\beta}(t-s)\partial_s u(x,y,s)ds = \kappa\Delta u(x,y,t) + f(x,y,t), & (x,y,t) \in \Omega \times (0,T], \\
u(x,y,t) = 0, & (x,y) \in \partial\Omega,\ 0 < t \leq T, \\
u(x,y,0) = u_0(x,y), & (x,y) \in \Omega,
\end{cases} \tag{1.1}
$$

where $0 < \beta < 1$, the diffusion coefficient $\kappa > 0$, $\Omega = [x_L, x_R] \times [y_L, y_R] \subset \mathbb{R}^2$ and the weakly singular kernel $\xi_\gamma(t) = t^{\gamma-1}/\Gamma(\gamma)$ $(t > 0)$.

Eq. (1.1) describes the dynamics of subdiffusion processes well, in which the mean square variance grows at a rate slower than in a Gaussian process [18]. For the analytical solution of Volterra problems with weakly singular kernel (so-called time-fractional problems), the Fourier transform method, the Laplace transform method and the Mellin transform method can be used [9]. However, in real applications, these methods are inappropriate for most time-fractional problems because of the nonlocality and complexity of the fractional derivatives. Thus, the development of efficient and reliable numerical techniques for solving Eq. (1.1) attracts many researchers.

Until now, numerous articles have been published for solving time-fractional problems numerically in an efficient way [19–26]. In these studies, the $L1$ or $L1$-type approximation is the most considered method to approach the Volterra operator in Eq. (1.1). The convergence rate of this approximation on a uniform mesh is $2 - \beta$ under the assumption that u is smooth on the closure of the domain [19]. This smoothness assumption is unrealistic and ignores the weak singularity near the initial time $t = 0$. This singularity has a great influence on the convergence rate. In order to compensate for the singular behaviour of the exact solution at $t = 0$, Mustapha et al. [27, 28] applied the $L1$ formula with non-uniform time step to solve a class of subdiffusion equations. Zhang et al. [22] investigated the non-uniform $L1$ approximation and applied it to solve fractional diffusion equations. Stynes et al. [29] proposed a finite difference scheme on a graded mesh in time to numerically solve time-fractional diffusion equations. Liao et al. [30] analyzed the non-uniform $L1$ approximation for solving reaction-subdiffusion equations. Other techniques for improving the poor accuracy of numerical approaches for the Volterra operator in Eq. (1.1) can be found in [24, 25, 31–35].

The numerical schemes mentioned above are time-stepping schemes. This means that the numerical solutions are obtained step-by-step. Another class of methods consists in computing the numerical solutions in a parallel-in-time (PinT) pattern. This includes the Laplacian inversion technique [36, 37] and the parareal algorithm [38–40]. Recently, all-at-once systems arising from fractional partial differential equations have been studied by many researchers [15, 16, 41–46]. Ke et al. [41] developed a fast solver based on the divide-and-conquer (DC) method for the all-at-once system arising from time-fractional partial differential equations. Lu et al. [43] proposed an approximate inversion (AI) method to solve the block lower triangular Toeplitz linear system with tridiagonal blocks from fractional sub-diffusion equations. Later, according to the

short-memory principle [47], Bertaccini and Durastante [45] proposed a limited memory block preconditioner for fast solving their linear systems from space-fractional equations. In [46], the authors studied an all-at-once system arising from high-dimensional space-fractional equations. In order to solve their system efficiently, they rewrote it into a tensor form and designed a tensor structured preconditioner. However, in these articles, fast algorithms are designed based on uniform meshes. In this paper, we try to get the numerical solution of Eq. (1.1) globally in time by solving the corresponding all-at-once system with variable time steps. In [27–30], the authors used a time graded mesh when approximating the Caputo fractional derivative. However, if the temporal regularity of the solution is small, the grid points of the time graded mesh become very dense near $t = 0$ and very sparse near $t = T$. This will reduce the numerical resolution of the solution. It is also a bad choice to derive an all-at-once system based on such a non-uniform mesh, because the storage requirement would be terrible and cannot be reduced. The authors in [30, Remark 8] suggested that if u is smooth away from $t = 0$, the time interval $[0, T]$ can be split into $[0, T_0]$ and $[T_0, T]$, where T_0 $(0 < T_0 < T)$ is reasonably small. Then, the graded mesh is applied to $[0, T_0]$ and the uniform mesh is used in $[T_0, T]$. According to their suggestion, we derive an all-at-once system in this paper based on such a strategy. Further, a fast algorithm is designed to solve this system efficiently.

The rest of this paper is organized as follows. Section 2 derives our all-at-once system of Eq. (1.1). In Section 3, the system is split into two subproblems, and two preconditioners are designed to efficiently solve these subproblems. Moreover, several properties of these preconditioners are investigated. In Section 4, we extend our algorithm to solve the semilinear problem of Eq. (1.1). In Section 5, numerical results are reported. Concluding remarks are given in Section 6.

2. The all-at-once system

In this section, we first derive the time-stepping scheme for approximating Eq. (1.1) by using the finite difference method. Then, the all-at-once system is obtained based on this scheme.

2.1. The time-stepping scheme

For a given reasonably small T_0 $(0 < T_0 < T)$, we split $[0, T]$ into two parts $[0, T_0]$ and $[T_0, T]$. In the first part $[0, T_0]$, we use the graded mesh $t_k = T_0 \left(\frac{k}{M_0} \right)^r$ for $k = 0, 1, \cdots, M_0$, where M_0 is a positive integer and $r \geq 1$ is the grading parameter. In the second part $[T_0, T]$, the uniform mesh $t_{M_0+k} = T_0 + k\tilde{\tau}$ $(k = 1, 2, \cdots, M - M_0)$ is used with $\tilde{\tau} = \frac{T - T_0}{M - M_0}$, where $M > M_0$ is a positive integer. Then, we get the mesh points $\{t_k\}_{k=0}^{M}$ and denote the time steps $\tau_k = t_k - t_{k-1}$ $(k = 1, 2, \cdots, M)$. Notice that for $M_0 + 1 \leq k \leq M$, we have $\tau_k = \tilde{\tau}$. Let $h_x = \frac{x_R - x_L}{N_x}$ and $h_y = \frac{y_R - y_L}{N_y}$ be the grid spacing in x and y directions for given positive integers N_x and N_y. Hence the space domain is discretized by $\bar{\omega}_h = \{(x_i, y_j) = (x_L + ih_x, y_L + jh_y) \mid 0 \leq i \leq N_x, \ 0 \leq j \leq N_y\}$. According to [30], the approximation on the

3

non-uniform mesh of the Volterra operator given in Eq. (1.1) is

$$
\int_0^t \xi_{1-\beta}(t-s)\partial_s u(x,y,s)ds \mid_{t=t_k} \approx a_0^{(k,\beta)} u(x,y,t_k) + \sum_{\ell=1}^{k-1} \left(a_{k-\ell}^{(k,\beta)} - a_{k-\ell-1}^{(k,\beta)} \right) u(x,y,t_\ell)
$$
$$
- a_{k-1}^{(k,\beta)} u(x,y,t_0)
$$
$$
\triangleq \delta_t^\beta u(x,y,t_k),
$$
(2.1)

where

$$
a_{k-\ell}^{(k,\beta)} = \int_{t_{\ell-1}}^{t_\ell} \frac{\xi_{1-\beta}(t_k-s)}{\tau_\ell}ds = \frac{\xi_{2-\beta}(t_k-t_{\ell-1}) - \xi_{2-\beta}(t_k-t_\ell)}{\tau_\ell}, \quad \ell=1,2,\cdots,k.
$$

Let u_{ij}^k be the approximation of $u(x_i,y_j,t_k)$ and $f_{ij}^k = f(x_i,y_j,t_k)$. Then, the time-stepping scheme of Eq. (1.1) is

$$
\delta_t^\beta u_{ij}^k = \kappa \delta_{xy}^2 u_{ij}^k + f_{ij}^k \quad \text{for} \quad 1 \le i \le N_x - 1, \ 1 \le j \le N_y - 1, \ 1 \le k \le M,
$$
(2.2)

where

$$
\delta_{xy}^2 u_{ij}^k = \frac{u_{i-1,j}^k - 2u_{ij}^k + u_{i+1,j}^k}{h_x^2} + \frac{u_{i,j-1}^k - 2u_{ij}^k + u_{i,j+1}^k}{h_y^2}.
$$

Let

$$
B_x = 1/h_x^2 \ \text{tridiag}(1,-2,1) \in \mathbb{R}^{(N_x-1)\times(N_x-1)}, \quad B_y = 1/h_y^2 \ \text{tridiag}(1,-2,1) \in \mathbb{R}^{(N_y-1)\times(N_y-1)},
$$
$$
\boldsymbol{u}^k = \left[u_{11}^k, \cdots, u_{N_x-1,1}^k, u_{12}^k, \cdots, u_{N_x-1,2}^k, \cdots, u_{1,N_y-1}^k, \cdots, u_{N_x-1,N_y-1}^k \right]^T
$$

and

$$
\boldsymbol{f}^k = \left[f_{11}^k, \cdots, f_{N_x-1,1}^k, f_{12}^k, \cdots, f_{N_x-1,2}^k, \cdots, f_{1,N_y-1}^k, \cdots, f_{N_x-1,N_y-1}^k \right]^T.
$$

Then, the matrix form of Eq. (2.2) is given by

$$
\delta_t^\beta \boldsymbol{u}^k = B\boldsymbol{u}^k + \boldsymbol{f}^k,
$$
(2.3)

where $B = \kappa \left(I_y \otimes B_x + B_y \otimes I_x \right)$ ("$\otimes$" denotes the Kronecker product). Here I_x and I_y are two identity matrices with sizes $N_x - 1$ and $N_y - 1$, respectively. Furthermore, the stability and convergence of the time-stepping scheme (2.2) can be proved simply based on the work [29, 30].

2.2. *The all-at-once system*

Before deriving our all-at-once system, several auxiliary symbols are introduced: I_t and I_s represent identity matrices with sizes M and $(N_x - 1)(N_y - 1)$, respectively. Denote

$$
\boldsymbol{u} = \left[\left(\boldsymbol{u}^1\right)^T, \left(\boldsymbol{u}^2\right)^T, \cdots, \left(\boldsymbol{u}^M\right)^T \right]^T \quad \text{and} \quad \boldsymbol{f} = \left[\left(\boldsymbol{f}^1\right)^T, \left(\boldsymbol{f}^2\right)^T, \cdots, \left(\boldsymbol{f}^M\right)^T \right]^T.
$$

With the help of Eq. (2.3), the all-at-once system can be written as:

$$\mathcal{M}\boldsymbol{u} = \boldsymbol{\eta} + \boldsymbol{f}, \tag{2.4}$$

where $\mathcal{M} = A \otimes I_s - I_t \otimes B$ with

$$A = \begin{bmatrix} a_0^{(1,\beta)} & 0 & 0 & \cdots & 0 & 0 \\ a_1^{(2,\beta)} - a_0^{(2,\beta)} & a_0^{(2,\beta)} & 0 & \cdots & \cdots & 0 \\ \vdots & a_1^{(3,\beta)} - a_0^{(3,\beta)} & a_0^{(3,\beta)} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ a_{M-2}^{(M-1,\beta)} - a_{M-3}^{(M-1,\beta)} & \ddots & \ddots & \ddots & a_0^{(M-1,\beta)} & 0 \\ a_{M-1}^{(M,\beta)} - a_{M-2}^{(M,\beta)} & a_{M-2}^{(M,\beta)} - a_{M-3}^{(M,\beta)} & \cdots & \cdots & a_1^{(M,\beta)} - a_0^{(M,\beta)} & a_0^{(M,\beta)} \end{bmatrix}$$

and $\boldsymbol{\eta} = \left[a_0^{(1,\beta)} \left(\boldsymbol{u}^0 \right)^T, a_1^{(2,\beta)} \left(\boldsymbol{u}^0 \right)^T, \cdots, a_{M-1}^{(M,\beta)} \left(\boldsymbol{u}^0 \right)^T \right]^T$.

If Gaussian elimination is applied in the block forward substitution (BFS) method [41, 42] to solve Eq. (2.4), the matrix $\mathcal{M}$ must be stored. Thus, the computational complexity and storage requirement of such a method are $\mathcal{O}(MN_x^3 N_y^3 + M^2 N_x N_y)$ and $\mathcal{O}(MN_x^2 N_y^2)$, respectively. In order to reduce the computational cost, we prefer to use Krylov subspace methods such as the biconjugate gradient stabilized (BiCGSTAB) method [48]. In the next section, an efficient algorithm is designed for fast solving Eq. (2.4).

3. Two preconditioners and their spectral analysis

In this section, two preconditioners are designed for solving Eq. (2.4). Several properties of these preconditioners are also investigated. It is easy to find that one part of A has Toeplitz structure due to the uniform mesh is used in $[T_0, T]$. Thus, the matrix A can be rewritten as the following 2×2 block matrix:

$$A = \begin{bmatrix} A_{11} & \mathbf{0} \\ A_{21} & A_{22} \end{bmatrix},$$

5

where $\mathbf{0}$ is a zero matrix with suitable size,

$$
A_{11} = \begin{bmatrix}
a_0^{(1,\beta)} & 0 & 0 & \cdots & 0 & 0 \\
a_1^{(2,\beta)} - a_0^{(2,\beta)} & a_0^{(2,\beta)} & 0 & \cdots & \cdots & 0 \\
\vdots & a_1^{(3,\beta)} - a_0^{(3,\beta)} & a_0^{(3,\beta)} & \ddots & \ddots & \vdots \\
\vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\
a_{M_0-2}^{(M_0-1,\beta)} - a_{M_0-3}^{(M_0-1,\beta)} & & \ddots & & \ddots & \ddots & a_0^{(M_0-1,\beta)} & 0 \\
a_{M_0-1}^{(M_0,\beta)} - a_{M_0-2}^{(M_0,\beta)} & a_{M_0-2}^{(M_0,\beta)} - a_{M_0-3}^{(M_0,\beta)} & \cdots & \cdots & a_1^{(M_0,\beta)} - a_0^{(M_0,\beta)} & a_0^{(M_0,\beta)}
\end{bmatrix},
$$

$$
A_{21} = \begin{bmatrix}
a_{M_0}^{(M_0+1,\beta)} - a_{M_0-1}^{(M_0+1,\beta)} & a_{M_0-1}^{(M_0+1,\beta)} - a_{M_0-2}^{(M_0+1,\beta)} & \cdots & a_1^{(M_0+1,\beta)} - b_0^{(\beta)} \\
a_{M_0+1}^{(M_0+2,\beta)} - a_{M_0}^{(M_0+2,\beta)} & a_{M_0}^{(M_0+2,\beta)} - a_{M_0-1}^{(M_0+2,\beta)} & \cdots & a_2^{(M_0+2,\beta)} - b_1^{(\beta)} \\
\vdots & \vdots & & \vdots \\
a_{M-2}^{(M-1,\beta)} - a_{M-3}^{(M-1,\beta)} & a_{M-3}^{(M-1,\beta)} - a_{M-4}^{(M-1,\beta)} & \cdots & a_{M-M_0-1}^{(M-1,\beta)} - b_{M-M_0-2}^{(\beta)} \\
a_{M-1}^{(M,\beta)} - a_{M-2}^{(M,\beta)} & a_{M-2}^{(M,\beta)} - a_{M-3}^{(M,\beta)} & \cdots & a_{M-M_0}^{(M,\beta)} - b_{M-M_0-1}^{(\beta)}
\end{bmatrix}
$$

and

$$
A_{22} = \begin{bmatrix}
\omega_0^{(\beta)} & 0 & 0 & \cdots & 0 & 0 \\
\omega_1^{(\beta)} & \omega_0^{(\beta)} & 0 & \cdots & \cdots & 0 \\
\vdots & \omega_1^{(\beta)} & \omega_0^{(\beta)} & \ddots & \ddots & \vdots \\
\vdots & & \ddots & \ddots & \ddots & 0 \\
\omega_{M-M_0-2}^{(\beta)} & \ddots & & \ddots & \omega_0^{(\beta)} & 0 \\
\omega_{M-M_0-1}^{(\beta)} & \omega_{M-M_0-2}^{(\beta)} & \cdots & \cdots & \omega_1^{(\beta)} & \omega_0^{(\beta)}
\end{bmatrix}
$$

with

$$
\omega_k^{(\beta)} = \begin{cases} b_0^{(\beta)}, & k = 0, \\ b_k^{(\beta)} - b_{k-1}^{(\beta)}, & k = 1, 2, \cdots, M - M_0 - 1. \end{cases} \tag{3.1}
$$

Here, $b_\ell^{(\beta)} = \frac{\tilde{\tau}^{-\beta}}{\Gamma(2-\beta)} \left[(\ell+1)^{1-\beta} - \ell^{1-\beta} \right]$ for $\ell \geq 0$. Let I_{t1} and I_{t2} be two identity matrices with sizes M_0 and $M - M_0$, respectively. Then, the solution of Eq. (2.4) can be obtained by solving the following equivalent two subproblems:

$$
\mathcal{M}_{11} \tilde{\boldsymbol{u}}_1 = \boldsymbol{\eta}_1 + \tilde{\boldsymbol{f}}_1, \tag{3.2a}
$$

$$
\mathcal{M}_{22} \tilde{\boldsymbol{u}}_2 = \boldsymbol{\eta}_2 + \tilde{\boldsymbol{f}}_2 - \mathcal{M}_{21} \tilde{\boldsymbol{u}}_1, \tag{3.2b}
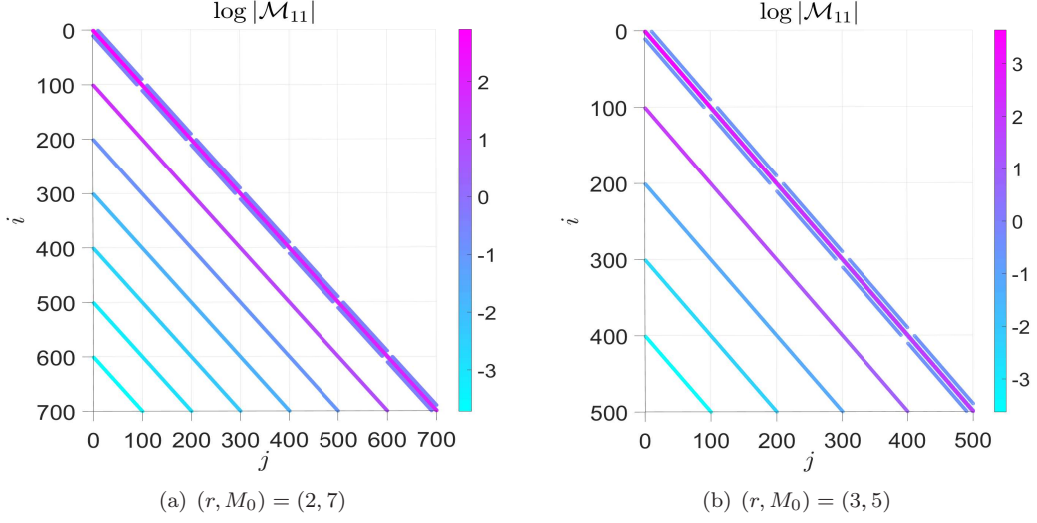$$

Fig. 1: The decay of the elements of matrix $\mathcal{M}_{11}$, where $\beta = 0.5$ and $N_x = N_y = 11$.

where

$$\mathcal{M}_{11} = A_{11} \otimes I_s - I_{t1} \otimes B, \quad \mathcal{M}_{21} = A_{21} \otimes I_s, \quad \mathcal{M}_{22} = A_{22} \otimes I_s - I_{t2} \otimes B,$$

$$\tilde{\boldsymbol{u}}_1 = \left[\left(\boldsymbol{u}^1 \right)^T, \left(\boldsymbol{u}^2 \right)^T, \cdots, \left(\boldsymbol{u}^{M_0} \right)^T \right]^T, \quad \tilde{\boldsymbol{u}}_2 = \left[\left(\boldsymbol{u}^{M_0+1} \right)^T, \left(\boldsymbol{u}^{M_0+2} \right)^T, \cdots, \left(\boldsymbol{u}^M \right)^T \right]^T,$$

$$\boldsymbol{\eta}_1 = \left[a_0^{(1,\beta)} \left(\boldsymbol{u}^0 \right)^T, a_1^{(2,\beta)} \left(\boldsymbol{u}^0 \right)^T, \cdots, a_{M_0-1}^{(M_0,\beta)} \left(\boldsymbol{u}^0 \right)^T \right]^T,$$

$$\boldsymbol{\eta}_2 = \left[a_{M_0}^{(M_0+1,\beta)} \left(\boldsymbol{u}^0 \right)^T, a_{M_0+1}^{(M_0+2,\beta)} \left(\boldsymbol{u}^0 \right)^T, \cdots, a_{M-1}^{(M,\beta)} \left(\boldsymbol{u}^0 \right)^T \right]^T,$$

$$\tilde{\boldsymbol{f}}_1 = \left[\left(\boldsymbol{f}^1 \right)^T, \left(\boldsymbol{f}^2 \right)^T, \cdots, \left(\boldsymbol{f}^{M_0} \right)^T \right]^T, \quad \tilde{\boldsymbol{f}}_2 = \left[\left(\boldsymbol{f}^{M_0+1} \right)^T, \left(\boldsymbol{f}^{M_0+2} \right)^T, \cdots, \left(\boldsymbol{f}^M \right)^T \right]^T.$$

These two subproblems are only coupled by $\tilde{\boldsymbol{u}}_1$. Thus, different methods can be used to solve them, such as the DC method [41, 42] and the AI method [43, 44]. For example, Krylov subspace methods [49] are used to solve Eq. (3.2a), while the DC method [41, 42] is employed to solve Eq. (3.2b). In this work, we choose the preconditioned BiCGSTAB (PBiCGSTAB) method [48] to solve both of them. It is worth mentioning that the general minimum residual (GMRES) method [49] is not used, since it requires large amounts of storage due to the orthogonalization process.

3.1. A block lower tridiagonal preconditioner for Eq. (3.2a)

It can be seen from Fig. 1 that the diagonal entries of $\mathcal{M}_{11}$ decay quickly. Inspired by this observation, a block lower tridiagonal preconditioner is designed for solving the subproblem (3.2a):

$$P_1 = \text{tri}(A_{11}) \otimes I_s - I_{t1} \otimes B,$$

7

where $\mathrm{tri}(A_{11})$ is a matrix which only preserves the first three diagonals of A_{11} and the others are set 0, i.e.,

$$
\mathrm{tri}(A_{11}) =
\begin{bmatrix}
a_0^{(1,\beta)} & 0 & 0 & \cdots & 0 & 0 \\
a_1^{(2,\beta)} - a_0^{(2,\beta)} & a_0^{(2,\beta)} & 0 & \cdots & \cdots & 0 \\
a_2^{(3,\beta)} - a_1^{(3,\beta)} & a_1^{(3,\beta)} - a_0^{(3,\beta)} & a_0^{(3,\beta)} & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & a_0^{(M_0-1,\beta)} & 0 \\
0 & 0 & \cdots & \cdots & a_1^{(M_0,\beta)} - a_0^{(M_0,\beta)} & a_0^{(M_0,\beta)}
\end{bmatrix}.
$$

The nonsingularity of P_1 is easy to check since all main diagonal blocks of it are nonsingular.

Theorem 3.1. *The degree of the minimal polynomial p [50] of $P_1^{-1}\mathcal{M}_{11}$ satisfies that*

$$
\deg p(P_1^{-1}\mathcal{M}_{11}) \leq \lceil M_0/3 \rceil.
$$

Thus, the dimension of the Krylov subspace $\mathcal{K}\left(P_1^{-1}\mathcal{M}_{11}; \boldsymbol{b}\right)$ is at most $\lceil M_0/3 \rceil$.

Proof. After some simple calculations, we have

$$
P_1^{-1}\mathcal{M}_{11} =
\begin{bmatrix}
I_s & \mathbf{0} & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \mathbf{0} \\
\mathbf{0} & I_s & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \vdots \\
\mathbf{0} & \mathbf{0} & I_s & \ddots & & & & \vdots \\
J_4^4 & \mathbf{0} & \ddots & \ddots & \ddots & & & \vdots \\
J_5^5 & J_5^4 & \ddots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
J_{M_0-1}^{M_0-1} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\
J_{M_0}^{M_0} & J_{M_0}^{M_0-1} & \cdots & \cdots & J_{M_0}^4 & \mathbf{0} & \mathbf{0} & I_s
\end{bmatrix},
$$

where

$$
J_k^4 = \left(a_3^{(k,\beta)} - a_2^{(k,\beta)}\right)\left(a_0^{(k,\beta)} I_s - B\right)^{-1}, \quad k \geq 4,
$$
$$
J_k^5 = \left(a_0^{(k,\beta)} I_s - B\right)^{-1}\left[\left(a_4^{(k,\beta)} - a_3^{(k,\beta)}\right) I_s - \left(a_1^{(k,\beta)} - a_0^{(k,\beta)}\right) J_{k-1}^4\right], \quad k \geq 5,
$$

and, for $6 \leq m \leq M_0$,

$$
J_k^m = \left(a_0^{(k,\beta)} I_s - B\right)^{-1}\left[\left(a_{k-1}^{(k,\beta)} - a_{k-2}^{(k,\beta)}\right) I_s - \left(a_2^{(k,\beta)} - a_1^{(k,\beta)}\right) J_{k-2}^{m-2} - \left(a_1^{(k,\beta)} - a_0^{(k,\beta)}\right) J_{k-1}^{m-1}\right], \quad m \leq k \leq M_0.
$$

This implies that all eigenvalues of $P_1^{-1}\mathcal{M}_{11}$ are equal to 1. On the other hand, $P_1^{-1}\mathcal{M}_{11} - I_{ts1}$ is a strictly block lower triangular matrix, where $I_{ts1} = I_{t1} \otimes I_s$. A simple computation shows that $\left(P_1^{-1}\mathcal{M}_{11} - I_{ts1}\right)^{\lceil M_0/3 \rceil} = \mathbf{0}$. However, it is even possible that $\left(P_1^{-1}\mathcal{M}_{11} - I_{ts1}\right)^m = \mathbf{0}$ for some $m \leq \lceil M_0/3 \rceil$. This means that the minimal polynomial $p(P_1^{-1}\mathcal{M}_{11})$ has a maximum degree of $\lceil M_0/3 \rceil$. From Saad's work [49], we know that the dimension of the Krylov subspace $\mathcal{K}\left(P_1^{-1}\mathcal{M}_{11}; \boldsymbol{b}\right)$ is then also at most $\lceil M_0/3 \rceil$. Thus, the proof is completed. $\qquad\square$

According to Theorem 3.1, it can be seen that if a Krylov subspace method with an optimal or Galerkin property is employed to solve the preconditioned form of (3.2a) with the coefficient matrix $P_1^{-1}\mathcal{M}_{11}$ in exact arithmetic, it will converge to the exact solution of (3.2a) in at most $\lceil M_0/3 \rceil$ iterations.

In practice, preconditioned Krylov subspace methods need to calculate $P_1^{-1}\boldsymbol{v}$, where $\boldsymbol{v}$ is a vector. Let

$$Q_x = \left(\sqrt{2/N_x}\sin\left(\frac{ij\pi}{N_x}\right)\right)_{1 \leq i,j \leq N_x - 1}, \qquad Q_y = \left(\sqrt{2/N_y}\sin\left(\frac{ij\pi}{N_y}\right)\right)_{1 \leq i,j \leq N_y - 1},$$

$$D_{Bx} = \mathrm{diag}\left(\lambda_1^{Bx}, \lambda_2^{Bx}, \cdots, \lambda_{N_x-1}^{Bx}\right), \qquad D_{By} = \mathrm{diag}\left(\lambda_1^{By}, \lambda_2^{By}, \cdots, \lambda_{N_y-1}^{By}\right)$$

with

$$\lambda_i^{Bx} = -\frac{4\kappa}{h_x^2}\sin^2\left(\frac{i\pi}{2N_x}\right) < 0 \quad \text{for} \quad 1 \leq i \leq N_x - 1$$

and

$$\lambda_j^{By} = -\frac{4\kappa}{h_y^2}\sin^2\left(\frac{j\pi}{2N_y}\right) < 0 \quad \text{for} \quad 1 \leq j \leq N_y - 1.$$

According to [51, Sec. 4.3], we know that $B = QD_BQ^T$, where

$$Q = Q_y \otimes Q_x \quad \text{and} \quad D_B = I_y \otimes D_{Bx} + D_{By} \otimes I_x = \mathrm{diag}\left(\lambda_1^B, \lambda_2^B, \cdots, \lambda_{(N_x-1)(N_y-1)}^B\right).$$

Hence, $\boldsymbol{z} = P_1^{-1}\boldsymbol{v}$ can be computed in the following way:

$$\begin{cases} \tilde{\boldsymbol{z}}_1 = (I_{t1} \otimes Q)\,\boldsymbol{v}, & \text{Step-(a1)}, \\ [\mathrm{tri}(A_{11}) \otimes I_s - I_{t1} \otimes D_B]\,\tilde{\boldsymbol{z}}_2 = \tilde{\boldsymbol{z}}_1, & \text{Step-(b1)}, \\ \boldsymbol{z} = \left(I_{t1} \otimes Q^T\right)\tilde{\boldsymbol{z}}_2, & \text{Step-(c1)}. \end{cases} \tag{3.3}$$

In Eq. (3.3), the first and the third step can be done by discrete sine transform. Each step itself can be carried out in parallel on M_0 processors. We notice that the matrix in Step-(b1) is a block tridiagonal matrix with diagonal blocks. Thus, $\tilde{\boldsymbol{z}}_2$ can be obtained by the BFS method with $\mathcal{O}(M_0 N_x N_y)$ operations and $\mathcal{O}(M_0 N_x N_y)$ memory.

3.2. A PinT preconditioner for Eq. (3.2b)

In this subsection, we concentrate on solving Eq. (3.2b) efficiently. Note that this subproblem is a block triangular Toeplitz matrix with a tridiagonal block system. The DC method [41, 42] or the AI method [43, 44] can be used to solve it. In this paper, the system is solved by a Krylov subspace method with a preconditioner. This preconditioner can be efficiently implemented in the PinT framework. Our PinT preconditioner (or structuring-circulant preconditioner, or semi-circulant preconditioner) is

$$P_\alpha = A_{22}^\alpha \otimes I_s - I_{t2} \otimes B, \tag{3.4}$$

where $A_{22}^\alpha = A_{22} + \alpha \tilde{A}$ is a α-circulant matrix with the parameter $\alpha \in (0, 1]$ and

$$\tilde{A} = \begin{bmatrix} 0 & \omega_{M-M_0-1}^{(\beta)} & \cdots & \omega_2^{(\beta)} & \omega_1^{(\beta)} \\ 0 & 0 & \omega_{M-M_0-1}^{(\beta)} & \cdots & \omega_2^{(\beta)} \\ 0 & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \omega_{M-M_0-1}^{(\beta)} \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix}.$$

Denote $\Theta_\alpha = \text{diag}\left(1, \alpha^{-\frac{1}{M-M_0}}, \cdots, \alpha^{-\frac{M-M_0-1}{M-M_0}}\right)$. Let $\mathbb{F}$ represent the discrete Fourier matrix, let "*" denote the conjugate transpose of a matrix. We know that the α-circulant matrix A_{22}^α has the following diagonalization:

$$A_{22}^\alpha = V_\alpha \Lambda_\alpha V_\alpha^{-1}$$

with $V_\alpha = \Theta_\alpha \mathbb{F}^*$ and

$$\Lambda_\alpha = \text{diag}\left(\mathbb{F}\Theta_\alpha^{-1} A_{22}^\alpha(:, 1)\right) = \text{diag}\left(\lambda_1^{(\alpha)}, \lambda_2^{(\alpha)}, \cdots, \lambda_{M-M_0}^{(\alpha)}\right) \tag{3.5}$$

contains all eigenvalues of A_{22}^α, where $A_{22}^\alpha(:, 1)$ is the first column of A_{22}^α.

The diagonalization of A_{22}^α immediately implies that $\boldsymbol{z} = P_\alpha^{-1}\boldsymbol{v}$ can be computed via the following three steps:

$$\begin{cases} \boldsymbol{z}_1 = \left(V_\alpha^{-1} \otimes I_s\right)\boldsymbol{v} = (\mathbb{F} \otimes I_s)\left[\left(\Theta_\alpha^{-1} \otimes I_s\right)\boldsymbol{v}\right], & \text{Step-(a)}, \\ \left(\lambda_n^{(\alpha)} I_s - B\right)\boldsymbol{z}_{2,n} = \boldsymbol{z}_{1,n}, \ 1 \leq n \leq M - M_0, & \text{Step-(b)}, \\ \boldsymbol{z} = (V_\alpha \otimes I_s)\boldsymbol{z}_2 = (\Theta_\alpha \otimes I_s)\left[(\mathbb{F}^* \otimes I_s)\boldsymbol{z}_2\right], & \text{Step-(c)}, \end{cases} \tag{3.6}$$

where $\boldsymbol{z}_j = \left[\boldsymbol{z}_{j,1}^T, \boldsymbol{z}_{j,2}^T, \cdots, \boldsymbol{z}_{j,M-M_0}^T\right]^T$ with $j = 1, 2$. In the first and third step, the matrix-vector multiplications can be done by fast Fourier transforms (FFTs) [1] in parallel on $M - M_0$ processors [15]. Thus,

---

[1] A parallel version of FFT is available at `http://www.fftw.org/parallel/parallel-fftw.html`.

the major computation cost of $P_\alpha^{-1} v$ comes from the second step. In this step, $M - M_0$ complex linear equations need to be solved, but they can be solved simultaneously (i.e., parallel computing).

Remark 1. *According to the work [52], we only solve the first $\left\lceil \frac{M-M_0+1}{2} \right\rceil$ shifted complex-valued linear systems in Step-(b). On the other hand, the matrix B can be diagonalized by a discrete sine transform. Thus, the storage requirement and the computational cost in Step-(b) are $\mathcal{O}\left(\left\lceil \frac{M-M_0+1}{2} \right\rceil N_x N_y\right)$ and $\mathcal{O}\left(\left\lceil \frac{M-M_0+1}{2} \right\rceil N_x N_y \log N_x N_y\right)$, respectively.*

Firstly, we investigate the nonsingularity of P_α. For this, the following result is needed.

Lemma 3.1. *For any $\beta \in (0,1)$ and $\omega_k^{(\beta)}$ defined in (3.1), it holds*

$$\omega_0^{(\beta)} > \sum_{\ell=1}^{M-M_0-1} \left| \omega_\ell^{(\beta)} \right| \quad \text{and} \quad \omega_\ell^{(\beta)} < 0 \quad \text{for} \quad \ell \geq 1. \tag{3.7}$$

Proof. It is direct to check that

$$\frac{\tilde{\tau}^{-\beta}}{\Gamma(2-\beta)} = \omega_0^{(\beta)} = b_0^{(\beta)} > b_1^{(\beta)} > \cdots > b_\ell^{(\beta)} > 0, \quad b_\ell^{(\beta)} \to 0 \quad \text{as} \quad \ell \to +\infty.$$

Immediately, we have $\omega_\ell^{(\beta)} < 0$ for $\ell \geq 1$. Then,

$$\sum_{\ell=1}^{M-M_0-1} \left| \omega_\ell^{(\beta)} \right| = \sum_{\ell=1}^{M-M_0-1} \left(b_{\ell-1}^{(\beta)} - b_\ell^{(\beta)} \right) = \omega_0^{(\beta)} - b_{M-M_0-1}^{(\beta)} < \omega_0^{(\beta)},$$

which completes the proof. $\qquad \square$

Based on Eq. (3.6) and the diagonalization of B, in order to prove the nonsingularity of P_α, we only need to show that the real part of $\lambda_j^{(\alpha)}$ is positive, i.e., $\operatorname{Re}\left(\lambda_j^{(\alpha)}\right) > 0$ for $1 \leq j \leq M - M_0$.

Theorem 3.2. *For any $\beta \in (0,1)$ and $\alpha \in (0,1]$, it holds that*

$$Re\left(\lambda_j^{(\alpha)}\right) > 0 \quad \text{for} \quad 1 \leq j \leq M - M_0.$$

Proof. By Eq. (3.5),

$$\lambda_j^{(\alpha)} = \sum_{n=0}^{M-M_0-1} \theta^{(j-1)n} \alpha^{n/(M-M_0)} \omega_n^{(\beta)}, \quad 1 \leq j \leq M - M_0$$

with $\theta = \exp(\frac{2\pi\iota}{M-M_0})$ and $\iota = \sqrt{-1}$. For some given suitable angle $\phi_{j,n}$, we rewrite $\theta^{(j-1)n} = \cos(\phi_{j,n}) + \iota \sin(\phi_{j,n})$. Then, using Lemma 3.1, we get

$$
\begin{aligned}
\mathrm{Re}\,(\lambda_j^{(\alpha)}) &= \omega_0^{(\beta)} + \sum_{n=1}^{M-M_0-1} \cos(\phi_{j,n}) \alpha^{n/(M-M_0)} \omega_n^{(\beta)} \\
&\geq \omega_0^{(\beta)} - \sum_{n=1}^{M-M_0-1} \left| \cos(\phi_{j,n}) \alpha^{n/(M-M_0)} \omega_n^{(\beta)} \right| \\
&\geq \omega_0^{(\beta)} - \sum_{\ell=1}^{M-M_0-1} \left| \omega_\ell^{(\beta)} \right| > 0,
\end{aligned}
$$

and the proof is completed. $\qquad\qquad\square$

To estimate $I_{ts2} - P_\alpha^{-1} \mathcal{M}_{22}$ (where $I_{ts2} = I_{t2} \otimes I_s$), the following lemma is useful.

Lemma 3.2. *([15]) For a strictly diagonally dominant (SDD) matrix $W \in \mathbb{C}^{M \times M}$, it holds*

$$
\left\| W^{-1} \right\|_\infty \leq \frac{\max\limits_{1 \leq n \leq M} \tilde{z}_n(W) / |W(n,n)|}{\min\limits_{1 \leq n \leq M} \left(1 - \tilde{h}_n(W) / |W(n,n)| \right)},
$$

where $\{\tilde{z}_n(W)\}_{n=1}^M$ and $\left\{ \tilde{h}_n(W) \right\}_{n=1}^M$ are given by the following two recursions

$$
\tilde{z}_1(W) = 1, \quad \tilde{z}_n(W) = 1 + \sum_{k=1}^{n-1} \frac{|W(n,k)|}{|W(k,k)|} \tilde{z}_k(W)
$$

and

$$
\tilde{h}_1(W) = \sum_{k=2}^{M} |W(1,k)|, \quad \tilde{h}_n(W) = \sum_{k=1}^{n-1} \frac{|W(n,k)|}{|W(k,k)|} \tilde{h}_k(W) + \sum_{k=n+1}^{M} |W(n,k)|,
$$

respectively.

For $\tilde{W} \in \mathbb{C}^{(M-M_0)(N_x-1)(N_y-1) \times (M-M_0)(N_x-1)(N_y-1)}$, we define the norm:

$$
\left\| \tilde{W} \right\|_{Q,\infty} = \left\| \left(I_{t2} \otimes Q^T \right) \tilde{W} \left(I_{t2} \otimes Q \right) \right\|_\infty.
$$

In order to get a sharp estimate of $\left\|I_{ts2} - P_\alpha^{-1}\mathcal{M}_{22}\right\|_{Q,\infty}$, we also need an auxiliary matrix defined as:

$$
L_\epsilon = \begin{bmatrix}
1 & & & & \\
\epsilon & 1 & & & \\
\epsilon^2 & \epsilon & 1 & & \\
\vdots & \ddots & \ddots & \ddots & \\
\epsilon^{M-M_0} & \cdots & \epsilon^2 & \epsilon & 1
\end{bmatrix},
$$

where ϵ is a free parameter. With this at hand, we can prove that $P_\alpha^{-1}\mathcal{M}_{22}$ is close to the identity.

Theorem 3.3. *Let $\epsilon_{max} = -\omega_1^{(\beta)}/\omega_0^{(\beta)}$ and $R_{\epsilon_{max}} = L_{\epsilon_{max}}A_{22}^\alpha$. Then, for any $\beta \in (0,1)$ and $\alpha \in (0,1]$, the following inequality holds*

$$
\left\|I_{ts2} - P_\alpha^{-1}\mathcal{M}_{22}\right\|_{Q,\infty} \le C\alpha,
$$

where

$$
C = \frac{\max\limits_{1 \le n \le M-M_0} \tilde{z}_n(R_{\epsilon_{max}})/R_{\epsilon_{max}}(n,n)}{\min\limits_{1 \le n \le M-M_0} \left(1 - \tilde{h}_n(R_{\epsilon_{max}})/R_{\epsilon_{max}}(n,n)\right)} \sum_{n=1}^{M-M_0-1} \epsilon_{max}^n \sum_{k=M-M_0-n}^{M-M_0-1} \left|\omega_k^{(\beta)}\right|
$$

is independent of the eigenvalues of B.

Proof. From the definition of P_α (3.4), we have

$$
\left\|I_{ts2} - P_\alpha^{-1}\mathcal{M}_{22}\right\|_{Q,\infty} = \alpha \left\|\left(A_{22}^\alpha \otimes I_s - I_{t2} \otimes D_B\right)^{-1}\left(\tilde{A} \otimes I_s\right)\right\|_\infty = \alpha \max_{\mu \in \sigma(B)} \left\|\left(A_{22}^\alpha - \mu I_{t2}\right)^{-1}\tilde{A}\right\|_\infty,
$$

where $\sigma(B) = \left\{\lambda_k^B\right\}_{k=1}^{(N_x-1)(N_y-1)}$. Thus, it turns to estimate

$$
\left\|\left(A_{22}^\alpha - \mu I_{t2}\right)^{-1}\tilde{A}\right\|_\infty = \left\|\left[L_\epsilon\left(A_{22}^\alpha - \mu I_{t2}\right)\right]^{-1}L_\epsilon\tilde{A}\right\|_\infty \le \left\|\left[L_\epsilon\left(A_{22}^\alpha - \mu I_{t2}\right)\right]^{-1}\right\|_\infty \left\|L_\epsilon\tilde{A}\right\|_\infty.
$$

Denote $\tilde{R}_\epsilon = L_\epsilon\left(A_{22}^\alpha - \mu I_{t2}\right)$, $\tilde{A}_\epsilon = L_\epsilon\tilde{A}$ and let $\epsilon = -\frac{\omega_1^{(\beta)}}{\omega_0^{(\beta)} - \mu}$. Then, according to Lemma 2.2 in [15], we know that $\tilde{R}_\epsilon$ is a SDD matrix. Using Lemma 3.2, we obtain

$$
\left\|\tilde{R}_\epsilon^{-1}\right\|_\infty \le \frac{\max\limits_{1 \le n \le M-M_0} \tilde{z}_n(\tilde{R}_\epsilon)/\left|\tilde{R}_\epsilon(n,n)\right|}{\min\limits_{1 \le n \le M-M_0} \left(1 - \tilde{h}_n(\tilde{R}_\epsilon)/\left|\tilde{R}_\epsilon(n,n)\right|\right)} \le \frac{\max\limits_{1 \le n \le M-M_0} \tilde{z}_n(R_{\epsilon_{max}})/R_{\epsilon_{max}}(n,n)}{\min\limits_{1 \le n \le M-M_0} \left(1 - \tilde{h}_n(R_{\epsilon_{max}})/R_{\epsilon_{max}}(n,n)\right)},
$$

where the relation $\left|\tilde{R}_\epsilon(n,n)\right| \ge R_{\epsilon_{max}}(n,n)$ is used in the second inequality.

On the other hand, after a routine calculation, we get

$$
\left\|\tilde{A}_\epsilon\right\|_\infty = \sum_{n=1}^{M-M_0-1} \epsilon^n \sum_{k=M-M_0-n}^{M-M_0-1} \left|\omega_k^{(\beta)}\right| \le \sum_{n=1}^{M-M_0-1} \epsilon_{max}^n \sum_{k=M-M_0-n}^{M-M_0-1} \left|\omega_k^{(\beta)}\right|,
$$

13

where $|\epsilon| \leq \epsilon_{max}$ is used. Thus, the proof is completed. $\qquad\square$

The result in Theorem 3.3 indicates that, with the new norm $\|\cdot\|_{Q,\infty}$, the preconditioned matrix $P_\alpha^{-1}\mathcal{M}_{22}$ is close to I_{ts2} as $\alpha \to 0$. This also means that the preconditioner P_α can indeed accelerate the convergence of an iterative method.

Remark 2. *Actually, Theorem 3.3 needs two essential properties:*

(1) The quadrature weights $\left\{\omega_k^{(\beta)}\right\}_{k=1}^{M-M_0}$ satisfy relation (3.7);

(2) The spatial discretization matrix B can be diagonalized.

If B cannot be diagonalized, but we know that $-B$ is a nonsingular M-matrix [43, Definition 1], then, according to [43, Corollary 10], P_α is invertible for $\alpha \in (0,1)$ and

$$\frac{\left\|P_\alpha^{-1} - \mathcal{M}_{22}^{-1}\right\|_\infty}{\left\|\mathcal{M}_{22}^{-1}\right\|_\infty} = \mathcal{O}(\alpha).$$

4. Extension to the semilinear form of Eq. (1.1)

In this section, we extend our method to solve the semilinear problem [53] of Eq. (1.1), i.e.,

$$\begin{cases} \int_0^t \xi_{1-\beta}(t-s)\partial_s u(x,y,s)ds = \kappa \Delta u(x,y,t) + g(u), & (x,y,t) \in \Omega \times (0,T], \\ u(x,y,t) = 0, & (x,y) \in \partial\Omega, \ 0 < t \leq T, \\ u(x,y,0) = u_0(x,y), & (x,y) \in \Omega, \end{cases} \quad (4.1)$$

where g is a nonlinear function and nonstiff. After discretization, we have the following nonlinear implicit scheme

$$\delta_t^\beta \boldsymbol{u}^k = B\boldsymbol{u}^k + g(\boldsymbol{u}^k), \quad \text{for} \quad 1 \leq k \leq M.$$

Then, the all-at-once system reads

$$\mathcal{M}\boldsymbol{u} - \boldsymbol{G}(\boldsymbol{u}) = \boldsymbol{\eta}, \quad (4.2)$$

where $\boldsymbol{G}(\boldsymbol{u}) = \left[g(\boldsymbol{u}^1)^T, \cdots, g(\boldsymbol{u}^M)^T\right]^T$. Similar to the linear case, this system is split into two subproblems:

$$\boldsymbol{G}_1(\tilde{\boldsymbol{u}}_1) = \mathcal{M}_{11}\tilde{\boldsymbol{u}}_1 - \boldsymbol{G}(\tilde{\boldsymbol{u}}_1) - \boldsymbol{\eta}_1 = \boldsymbol{0}, \quad (4.3a)$$

$$\boldsymbol{G}_2(\tilde{\boldsymbol{u}}_2) = \mathcal{M}_{22}\tilde{\boldsymbol{u}}_2 - \boldsymbol{G}(\tilde{\boldsymbol{u}}_2) - \boldsymbol{\eta}_2 - \mathcal{M}_{21}\tilde{\boldsymbol{u}}_1 = \boldsymbol{0}. \quad (4.3b)$$

In this paper, both of them are solved by a modified Newton method. Again, to accelerate the speed of a Krylov subspace method for solving the linearized equations, the two preconditioners P_1 and P_α are used.

Now, we derive our modified Newton method for solving (4.3). For the subproblem (4.3a), its solution can be obtained from the following iteration process with an initial value $\tilde{\boldsymbol{u}}_1^{(0)}$

$$\mathcal{M}_{11}\boldsymbol{U}_1^\ell = \boldsymbol{G}_1(\tilde{\boldsymbol{u}}_1^{(\ell)}), \qquad \tilde{\boldsymbol{u}}_1^{(\ell+1)} = \tilde{\boldsymbol{u}}_1^{(\ell)} - \boldsymbol{U}_1^\ell.$$

Then, the preconditioner P_1 can be directly used to accelerate solving the above equation.

Similarly, the solution of (4.3b) can be obtained from

$$\mathcal{M}_{22}\boldsymbol{U}_2^\ell = \boldsymbol{G}_2(\tilde{\boldsymbol{u}}_2^{(\ell)}), \qquad \tilde{\boldsymbol{u}}_2^{(\ell+1)} = \tilde{\boldsymbol{u}}_2^{(\ell)} - \boldsymbol{U}_2^\ell$$

with an initial value $\tilde{\boldsymbol{u}}_2^{(0)}$. The preconditioner P_α can be used to solve this equation efficiently.

In the modified Newton method, the choice of the initial values $\tilde{\boldsymbol{u}}_1^{(0)}$ and $\tilde{\boldsymbol{u}}_2^{(0)}$ can affect its convergence rate. Thus, they should be chosen with care. In this work, following the idea of [16], the initial values $\tilde{\boldsymbol{u}}_1^{(0)}$ and $\tilde{\boldsymbol{u}}_2^{(0)}$ are obtained by interpolating the numerical solution of the following linearized scheme on the coarser mesh:

$$\delta_t^\beta \boldsymbol{u}^k = B\boldsymbol{u}^k + g(\boldsymbol{u}^{k-1}), \quad \text{for} \quad 1 \le k \le M.$$

Moreover, the modified Newton method terminates if $\frac{\|\boldsymbol{U}_k^\ell\|_2}{\|\tilde{\boldsymbol{u}}_k^{(0)}\|_2} \le 10^{-10}$ $(k = 1, 2)$ or the iteration number is more than 200.

Table 1: Summary of used abbreviations.

Symbol	Explanation
BFSM	The BFS method is used to solve (2.4) or (4.3).
$\mathcal{I}$	The BiCGSTAB method is used when solving (2.4) or (4.3).
$\mathcal{P}$	The PBiCGSTAB method is used when solving (2.4) or (4.3).
Iter	Iter $=$ (Iter(1), Iter(2)), where Iter(1) (Iter(2)) is the number of iterations required for solving Eq. (3.2a) (Eq. (3.2b)).
Iter1	Iter1 $= \frac{1}{M}\sum_{k=1}^{M}$ Iter1(k), where Iter1(k) is the number of iterations of the modified Newton method in the kth step of BFSM for solving (4.2).
Iter$_O$	Iter$_O$ $=$ (Iter$_O$(1), Iter$_O$(2)), where Iter$_O$(1) (Iter$_O$(2)) is the number of iterations of the modified Newton method for solving Eq. (4.3a) (Eq. (4.3b)).
Iter$_I$	Iter$_I$ $=$ (Iter$_I$(1), Iter$_I$(2)), where Iter$_I$(1) $= \frac{1}{\text{Iter}_O(1)}\sum_{k=1}^{\text{Iter}_O(1)}$ Iter$_{I1}$(k) and $\quad$ Iter$_I$(2) $= \frac{1}{\text{Iter}_O(2)}\sum_{k=1}^{\text{Iter}_O(2)}$ Iter$_{I2}$(k).
Iter$_{I1}$(k)	The number of iterations required by the (P)BiCGSTAB method in the kth step of the modified Newton method for solving Eq. (4.3a).
Iter$_{I2}$(k)	The number of iterations required by the (P)BiCGSTAB method in the kth step of the modified Newton method for solving Eq. (4.3b).
Time	Total CPU time in seconds.
−	The data is not obtained in 8 hours.
†	The BiCGSTAB method needs more than 1000 iterations to reach the desired tolerance.

5. Numerical experiments

In this section, three examples are reported to show the performance of our strategy proposed in Sections 3 and 4. The (P)BiCGSTAB method for solving (3.2) terminates if the relative residual error satisfies $\frac{\|\boldsymbol{r}^{(k)}\|_2}{\|\boldsymbol{r}^{(0)}\|_2} \leq 10^{-9}$ (for the nonlinear case we chose 10^{-6}) or the iteration number is more than 1000, where $\boldsymbol{r}^{(k)}$ denotes residual vector in the kth iteration. The initial guess is chosen as the zero vector. In our experiments, we set $N = N_x = N_y$, $T_0 = 2^{-r}$, $M_0 = \left\lceil \frac{r}{2^r-1+r} M \right\rceil$ and $\alpha = \min\{10^{-4}, 0.5\tilde{\tau}\}$. All of the symbols shown in Table 1 will appear later.

All experiments are carried out via MATLAB 2018b on a Windows 10 (64 bit) PC with the configuration: Intel(R) Core(TM) i7-8700k CPU 3.20 GHz and 16 GB RAM.

Table 2: Results of various methods for $M = N$ for Example 1.

(β, r)	N	BFSM Time	$\mathcal{I}$ Iter	$\mathcal{I}$ Time	$\mathcal{P}$ Iter	$\mathcal{P}$ Time
(0.1, 2)	32	0.037	(37.0, 41.0)	0.251	(3.0, 1.0)	0.032
	64	0.328	(67.0, 71.0)	1.645	(4.0, 1.0)	0.182
	128	4.190	(136.0, 145.0)	36.962	(4.0, 1.0)	1.399
	256	44.256	(265.0, 303.0)	906.017	(5.0, 1.0)	18.111
	512	1108.741	–	–	(5.0, 1.0)	224.922
(0.5, 2)	32	0.039	(27.0, 31.0)	0.190	(3.0, 1.0)	0.033
	64	0.326	(55.0, 77.0)	1.706	(5.0, 2.0)	0.263
	128	3.937	(128.0, 184.0)	44.789	(7.0, 2.0)	2.425
	256	44.382	(308.0, 401.0)	1178.911	(10.0, 2.0)	34.433
	512	1115.207	–	–	(14.0, 2.0)	551.918
(0.9, 2)	32	0.050	(32.0, 24.0)	0.154	(3.0, 2.0)	0.042
	64	0.344	(64.0, 71.0)	1.632	(4.0, 2.0)	0.230
	128	3.890	(177.0, 213.0)	53.116	(5.0, 2.0)	2.050
	256	44.434	(579.0, 638.0)	1915.291	(6.0, 2.0)	25.939
	512	1111.531	–	–	(9.0, 2.0)	434.259
(0.1, 3)	32	0.038	(36.0, 44.0)	0.338	(3.0, 1.0)	0.032
	64	0.323	(63.0, 73.0)	2.617	(3.0, 1.0)	0.162
	128	3.894	(139.0, 148.0)	51.725	(4.0, 1.0)	1.392
	256	44.510	(291.0, 312.0)	1156.569	(4.0, 1.0)	15.548
	512	1115.510	–	–	(5.0, 1.0)	217.845
(0.5, 3)	32	0.036	(33.0, 36.0)	0.277	(3.0, 1.0)	0.032
	64	0.320	(56.0, 86.0)	3.013	(4.0, 2.0)	0.255
	128	3.911	(127.0, 197.0)	66.950	(5.0, 2.0)	2.194
	256	44.728	(267.0, 422.0)	1510.010	(7.0, 2.0)	27.578
	512	1108.584	–	–	(10.0, 2.0)	442.688
(0.9, 3)	32	0.035	(48.0, 28.0)	0.224	(2.0, 2.0)	0.038
	64	0.323	(121.0, 85.0)	3.102	(3.0, 2.0)	0.225
	128	3.882	(386.0, 256.0)	93.875	(4.0, 2.0)	2.042
	256	45.184	†	†	(5.0, 2.0)	24.569
	512	1111.146	–	–	(6.0, 2.0)	373.949

Example 1. In this example, the subdiffusion problem (1.1) is considered on $\Omega = [-4, 10] \times [-4, 10]$ with

$T = 1$ and the source term

$$
f(x, y, t) = \frac{\xi_{1+\sigma-\beta}(t)}{\sqrt{2\pi}} \left[\exp\left(-\frac{x^2 + y^2}{2}\right) + \exp\left(-\frac{(x-3)^2 + (y-3)^2}{2}\right) \right] - \kappa \frac{1 + \xi_{1+\sigma}(t)}{\sqrt{2\pi}} \times
$$

$$
\left\{ (x^2 + y^2 - 2) \exp\left(-\frac{x^2 + y^2}{2}\right) + \left[(x-3)^2 + (y-3)^2 - 2\right] \exp\left(-\frac{(x-3)^2 + (y-3)^2}{2}\right) \right\},
$$

where $\kappa = 1$ and $\sigma = 2.2 - \beta$. For the above choice, the exact solution is

$$
u(x, y, t) = \frac{1 + \xi_{1+\sigma}(t)}{\sqrt{2\pi}} \left[\exp\left(-\frac{x^2 + y^2}{2}\right) + \exp\left(-\frac{(x-3)^2 + (y-3)^2}{2}\right) \right].
$$

(a) Eigenvalues of $\mathcal{M}_{11}$

(b) Eigenvalues of $P_1^{-1}\mathcal{M}_{11}$

(c) Eigenvalues of $\mathcal{M}_{22}$

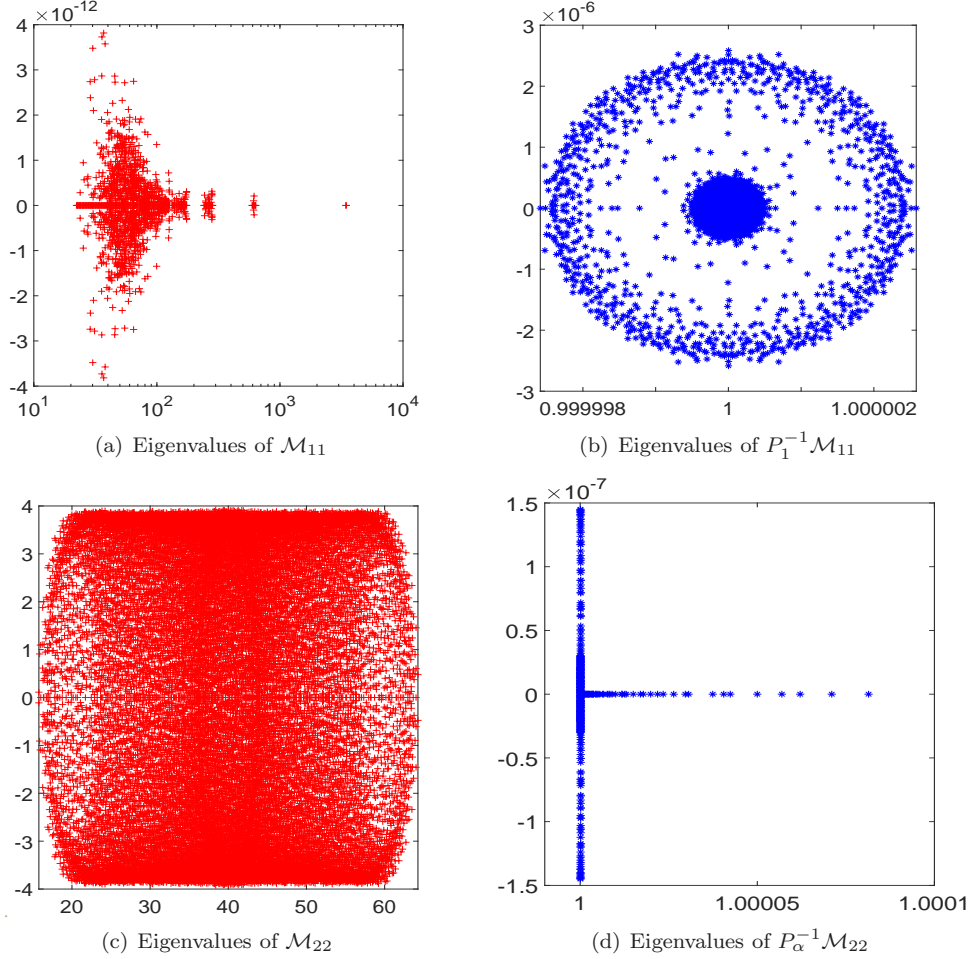(d) Eigenvalues of $P_\alpha^{-1}\mathcal{M}_{22}$

Fig. 2: Spectra of $\mathcal{M}_{11}$, $P_1^{-1}\mathcal{M}_{11}$, $\mathcal{M}_{22}$ and $P_\alpha^{-1}\mathcal{M}_{22}$, for $(\beta, r) = (0.9, 3)$ and $M = N = 32$ in Example 1.

In Table 2, the CPU time of method $\mathcal{P}$ is the smallest one among the three tested methods. Comparing the number Iter of methods $\mathcal{I}$ and $\mathcal{P}$, it can be found that our preconditioners P_1 and P_α are very efficient. We

also notice that the number Iter of our method (i.e., $\mathcal{P}$) is not strongly influenced by the mesh size. Moreover, Fig. 2 shows the spectra of $\mathcal{M}_{11}$, $P_1^{-1}\mathcal{M}_{11}$, $\mathcal{M}_{22}$ and $P_\alpha^{-1}\mathcal{M}_{22}$ with $(\beta, r) = (0.9, 3)$ and $M = N = 32$. It should be mentioned that the effect of the clustered eigenvalues on the convergence of the (P)BiCGSTAB method can be not so crucial [54].

Table 3: Results of various methods for $M = N$ for Example 2.

		BFSM		$\mathcal{I}$			$\mathcal{P}$		
(β, r)	N	Iter1	Time	Iter$_O$	Iter$_I$	Time	Iter$_O$	Iter$_I$	Time
(0.1, 2)	32	9.8	0.263	(10.0, 10.0)	(66.9, 55.7)	3.409	(10.0, 10.0)	(2.0, 1.0)	0.320
	64	9.0	2.193	(10.0, 10.0)	(135.9, 100.9)	24.137	(10.0, 10.0)	(2.1, 1.0)	1.283
	128	9.0	25.604	(10.0, 10.0)	(271.2, 219.3)	579.351	(10.0, 10.0)	(2.8, 1.0)	10.991
	256	8.1	238.001	(10.0, 10.0)	(607.2, 457.3)	14827.553	(10.0, 10.0)	(2.9, 1.0)	138.210
	512	8.0	3115.653	–	–	–	(10.0, 10.0)	(3.0, 1.0)	1789.639
(0.5, 2)	32	7.7	0.207	(8.0, 10.0)	(61.0, 65.2)	3.869	(8.0, 10.0)	(2.4, 1.0)	0.303
	64	7.1	1.772	(8.0, 10.0)	(139.9, 135.4)	30.318	(8.0, 10.0)	(3.5, 1.0)	1.468
	128	6.7	19.401	(8.0, 10.0)	(318.4, 295.3)	742.383	(8.0, 10.0)	(4.5, 1.0)	12.320
	256	6.0	181.785	(8.0, 10.0)	(701.5, 642.7)	19232.301	(8.0, 10.0)	(6.1, 1.0)	181.685
	512	5.7	2469.990	–	–	–	(8.0, 10.0)	(8.0, 1.0)	2628.717
(0.9, 2)	32	6.4	0.175	(6.0, 9.0)	(42.5, 57.7)	2.993	(6.0, 9.0)	(2.0, 1.0)	0.262
	64	5.6	1.427	(6.0, 9.0)	(104.2, 152.3)	28.987	(6.0, 9.0)	(2.7, 1.0)	1.107
	128	4.9	14.642	(6.0, 9.0)	(279.5, 418.4)	873.289	(6.0, 9.0)	(3.2, 1.0)	9.044
	256	4.7	146.312	(6.0, 9.0)	(841.2, 856.4)	22173.364	(6.0, 9.0)	(4.0, 1.0)	123.540
	512	4.0	1978.387	–	–	–	(6.0, 9.0)	(6.2, 1.0)	1842.001
(0.1, 3)	32	9.8	0.269	(9.0, 10.0)	(66.9, 59.8)	4.424	(9.0, 10.0)	(2.0, 1.0)	0.253
	64	9.0	2.249	(9.0, 10.0)	(135.4, 110.1)	39.107	(9.0, 10.0)	(2.0, 1.0)	1.388
	128	8.9	25.403	(9.0, 10.0)	(276.1, 231.6)	817.984	(9.0, 10.0)	(2.0, 1.0)	11.298
	256	8.2	237.566	(9.0, 10.0)	(595.4, 488.2)	18366.031	(9.0, 10.0)	(2.6, 1.0)	133.166
	512	8.0	3031.510	–	–	–	(9.0, 10.0)	(2.9, 1.0)	1795.297
(0.5, 3)	32	7.6	0.215	(7.0, 10.0)	(51.6, 66.5)	4.882	(7.0, 10.0)	(2.0, 1.0)	0.228
	64	6.9	1.759	(7.0, 10.0)	(116.4, 140.8)	47.844	(7.0, 10.0)	(2.9, 1.0)	1.463
	128	6.7	19.198	(7.0, 10.0)	(258.3, 296.9)	984.842	(7.0, 10.0)	(3.6, 1.0)	12.189
	256	5.9	174.579	(7.0, 10.0)	(601.3, 648.1)	23378.635	(7.0, 10.0)	(5.1, 1.0)	152.465
	512	5.7	2410.630	–	–	–	(7.0, 10.0)	(6.4, 1.0)	2161.753
(0.9, 3)	32	6.3	0.180	(6.0, 9.0)	(58.5, 60.8)	3.999	(6.0, 9.0)	(2.0, 1.0)	0.199
	64	5.7	1.450	(6.0, 9.0)	(144.8, 153.8)	46.973	(6.0, 9.0)	(2.0, 1.0)	1.171
	128	4.8	14.497	(6.0, 9.0)	(431.0, 430.8)	1294.543	(6.0, 9.0)	(2.3, 1.0)	9.938
	256	4.7	143.793	†	†	†	(6.0, 9.0)	(3.3, 1.0)	119.759
	512	3.9	1913.440	–	–	–	(6.0, 9.0)	(3.8, 1.0)	1626.390

Example 2. We consider the two-dimensional time fractional Fisher equation. More precisely, in Eq. (4.1), we choose $\Omega = [0, \pi] \times [0, \pi]$, $T = 1$, the diffusion coefficient $\kappa = 1$, the nonlinear term $g(u) = u(1 - u)$ and the initial value $u_0(x, y) = \sin x \sin y$.

In Table 3, the CPU time and the numbers of iterations of the methods BFSM, $\mathcal{I}$ and $\mathcal{P}$ for solving the nonlinear problem are reported. Compared with the BFSM method, our method indeed reduces the CPU time except for some cases. For these unsatisfied cases, although the CPU times required by our method $\mathcal{P}$ are larger than the BFSM method, our method still has a potential advantage in terms of parallel computing. Fig. 3 shows the spectra of $\mathcal{M}_{11}$, $P_1^{-1}\mathcal{M}_{11}$, $\mathcal{M}_{22}$ and $P_\alpha^{-1}\mathcal{M}_{22}$ with $(\beta, r) = (0.5, 2)$ and $M = N = 32$. It
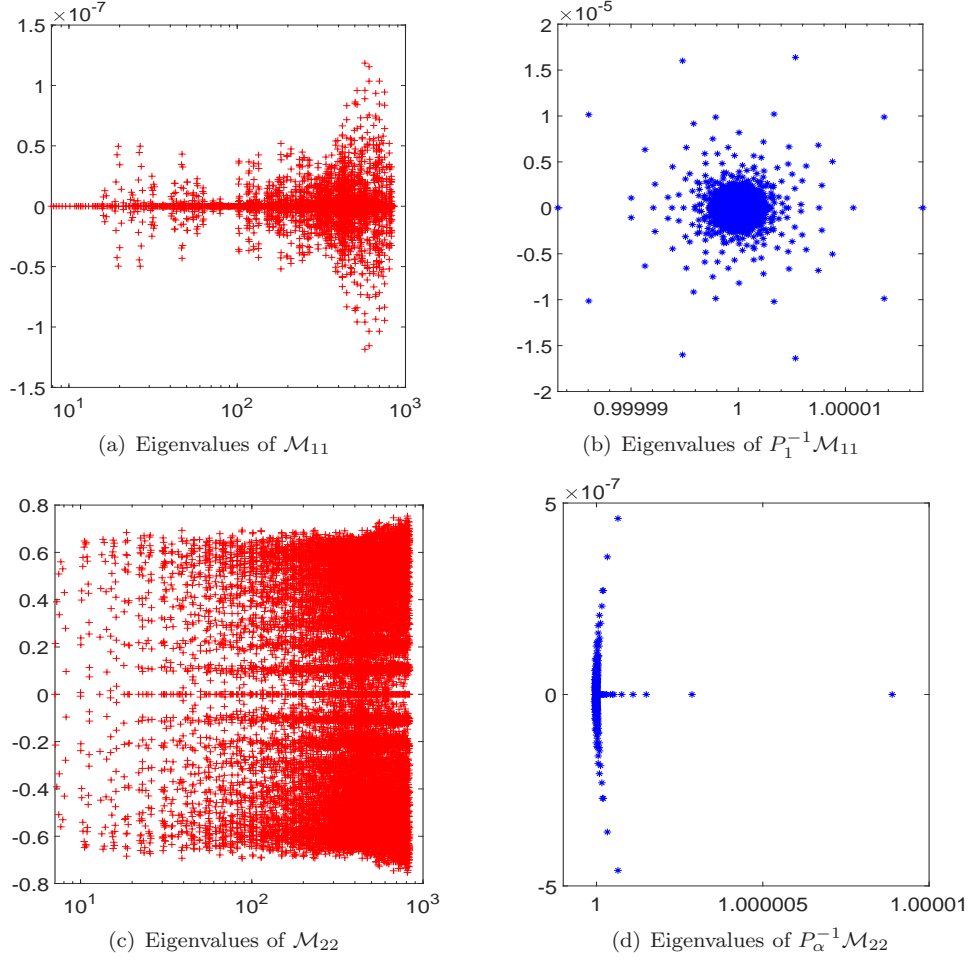
(a) Eigenvalues of $\mathcal{M}_{11}$

(b) Eigenvalues of $P_1^{-1}\mathcal{M}_{11}$

(c) Eigenvalues of $\mathcal{M}_{22}$

(d) Eigenvalues of $P_\alpha^{-1}\mathcal{M}_{22}$

Fig. 3: Spectra of $\mathcal{M}_{11}$, $P_1^{-1}\mathcal{M}_{11}$, $\mathcal{M}_{22}$ and $P_\alpha^{-1}\mathcal{M}_{22}$, for $(\beta, r) = (0.5, 2)$ and $M = N = 32$ in Example 2.

is clearly seen from Fig. 3 that all eigenvalues of the preconditioned matrices $P_1^{-1}\mathcal{M}_{11}$ and $P_\alpha^{-1}\mathcal{M}_{22}$ are clustered around 1.

6. Concluding remarks

A parallel preconditioning technique is proposed to solve the all-at-once system (2.4) with variable time steps arising from subdiffusion equations (1.1). Firstly, we split the time interval $[0, T]$ into two parts $[0, T_0]$ and $[T_0, T]$. Then, we use the graded $L1$ scheme to approximate (1.1) in $[0, T_0]$, while the uniform one is applied in $[T_0, T]$. Secondly, our all-at-once system (2.4) is derived based on this decomposition. Thanks to the local Toeplitz structure of the time discretization matrix A, the solution of Eq. (2.4) can be obtained by solving (3.2). Two preconditioners P_1 and P_α are proposed to accelerate obtaining the solution of Eq. (3.2). Some properties of these two preconditioners are also analyzed. In Section 4, we extend our technique the nonlinear subdiffusion problem (4.1). Finally, numerical experiments are reported that show the performance

of our preconditioning technique. It is worth mentioning that the CPU time required by the method $\mathcal{P}$ can be further reduced since it is suitable for parallel computing.

In this work, we consider the nonlinear function g to be nonstiff. If g is stiff, we suggest using Newton's method to solve (4.3). For this case, our preconditioners need some modifications as proposed in [15, Section 3] to make them more efficient. Another benefit of such modifications is that the new preconditioners are still suitable for parallel computing. In our future work, we will study the all-at-once system with a space discretization matrix B being indefinite.

Acknowledgments

Declarations

Conflict of interest The authors declare that they have no competing interests.

References

[1] I. M. Sokolov, J. Klafter, A. Blumen, Fractional kinetics, Phys. Today 55 (2002) 48–54.

[2] R. Metzler, W. Schick, H.-G. Kilian, T. F. Nonnenmacher, Relaxation in filled polymers: A fractional calculus approach, J. Chem. Phys. 103 (1995) 7180–7186.

[3] J.-H. He, Approximate analytical solution for seepage flow with fractional derivatives in porous media, Comput. Meth. Appl. Mech. Eng. 167 (1998) 57–68.

[4] D. del-Castillo-Negrete, B. Carreras, V. Lynch, Fractional diffusion in plasma turbulence, Phys. Plasmas 11 (2004) 3854–3864.

[5] R. Metzler, J. Klafter, Boundary value problems for fractional diffusion equations, Physica A 278 (2000) 107–125.

[6] R. Gorenflo, F. Mainardi, D. Moretti, P. Paradisi, Time fractional diffusion: a discrete random walk approach, Nonlinear Dyn. 29 (2002) 129–143.

[7] I. Podlubny, A. Chechkin, T. Skovranek, Y. Chen, B. M. V. Jara, Matrix approach to discrete fractional calculus II: Partial fractional differential equations, J. Comput. Phys. 228 (2009) 3137–3153.

[8] G. Pagnini, P. Paradisi, A stochastic solution with Gaussian stationary increments of the symmetric space-time fractional diffusion equation, Fract. Calc. Appl. Anal. 19 (2016) 408–440.

[9] I. Podlubny, Fractional Differential Equations, Vol. 198, Academic Press, San Diego, CA, 1998.

[10] X.-L. Lin, M. K. Ng, H.-W. Sun, Crank–Nicolson alternative direction implicit method for space-fractional diffusion equations with nonseparable coefficients, SIAM J. Numer. Anal. 57 (2019) 997–1019.

[11] S.-L. Lei, W. Wang, X. Chen, D. Ding, A fast preconditioned penalty method for American options pricing under regime-switching tempered fractional diffusion models, J. Sci. Comput. 75 (2018) 1633–1655.

[12] J. Shen, C. Li, Z.-Z. Sun, An H2N2 interpolation for Caputo derivative with order in (1,2) and its application to time-fractional wave equations in more than one space dimension, J. Sci. Comput. 83 (2020) 38. `doi:10.1007/s10915-020-01219-8`.

[13] H.-L. Liao, W. McLean, J. Zhang, A discrete Grönwall inequality with applications to numerical schemes for subdiffusion problems, SIAM J. Numer. Anal. 57 (2019) 218–237.

[14] J. Cao, G. Song, J. Wang, Q. Shi, S. Sun, Blow-up and global solutions for a class of time fractional nonlinear reaction-diffusion equation with weakly spatial source., Appl. Math. Lett. 91 (2019) 201–206.

[15] X.-M. Gu, S.-L. Wu, A parallel-in-time iterative algorithm for Volterra partial integral-differential problems with weakly singular kernel, J. Comput. Phys. 417 (2020) 109576. `doi:10.1016/j.jcp.2020.109576`.

[16] Y.-L. Zhao, P.-Y. Zhu, X.-M. Gu, X.-L. Zhao, H.-Y. Jian, A preconditioning technique for all-at-once system from the nonlinear tempered fractional diffusion equation, J. Sci. Comput. 83 (2020) 10. `doi:10.1007/s10915-020-01193-1`.

[17] M. Li, Y.-L. Zhao, A fast energy conserving finite element method for the nonlinear fractional Schrödinger equation with wave operator, Appl. Math. Comput. 338 (2018) 758–773.

[18] J.-P. Bouchaud, A. Georges, Anomalous diffusion in disordered media: statistical mechanisms, models and physical applications, Phys. Rep. 195 (1990) 127–293.

[19] Y. Lin, C. Xu, Finite difference/spectral approximations for the time-fractional diffusion equation, J. Comput. Phys. 225 (2007) 1533–1552.

[20] X. Li, C. Xu, A space-time spectral method for the time fractional diffusion equation, SIAM J. Numer. Anal. 47 (2009) 2108–2131.

[21] G.-H. Gao, Z.-Z. Sun, H.-W. Zhang, A new fractional numerical differentiation formula to approximate the Caputo fractional derivative and its applications, J. Comput. Phys. 259 (2014) 33–50.

[22] Y.-N. Zhang, Z.-Z. Sun, H.-L. Liao, Finite difference methods for the time fractional diffusion equation on non-uniform meshes, J. Comput. Phys. 265 (2014) 195–210.

[23] A. A. Alikhanov, A new difference scheme for the time fractional diffusion equation, J. Comput. Phy. 280 (2015) 424–438.

[24] B. Jin, R. Lazarov, Z. Zhou, An analysis of the L1 scheme for the subdiffusion equation with nonsmooth data, IMA J. Numer. Anal. 36 (2016) 197–221.

[25] F. Zeng, C. Li, F. Liu, I. Turner, Numerical algorithms for time-fractional subdiffusion equation with second-order accuracy, SIAM J. Sci. Comput. 37 (2015) A55–A78.

[26] X. Hu, C. Rodrigo, F. J. Gaspar, Using hierarchical matrices in the solution of the time-fractional heat equation by multigrid waveform relaxation, J. Comput. Phys. (2020) 109540.

[27] K. Mustapha, J. AlMutawa, A finite difference method for an anomalous sub-diffusion equation, theory and applications, Numer. Algorithms 61 (2012) 525–543.

[28] K. Mustapha, An implicit finite-difference time-stepping method for a sub-diffusion equation, with spatial discretization by finite elements, IMA J. Numer. Anal. 31 (2011) 719–739.

[29] M. Stynes, E. O'Riordan, J. L. Gracia, Error analysis of a finite difference method on graded meshes for a time-fractional diffusion equation, SIAM J. Numer. Anal. 55 (2017) 1057–1079.

[30] H.-L. Liao, D. Li, J. Zhang, Sharp error estimate of the nonuniform L1 formula for linear reaction-subdiffusion equations, SIAM J. Numer. Anal. 56 (2018) 1112–1133.

[31] C. Lubich, I. Sloan, V. Thomée, Nonsmooth data error estimates for approximations of an evolution equation with a positive-type memory term, Math. Comput. 65 (1996) 1–17.

[32] F. Zeng, Z. Zhang, G. E. Karniadakis, Second-order numerical methods for multi-term fractional differential equations: smooth and non-smooth solutions, Comput. Meth. Appl. Mech. Eng. 327 (2017) 478–502.

[33] Y. Yan, M. Khan, N. J. Ford, An analysis of the modified L1 scheme for time-fractional partial differential equations with

nonsmooth data, SIAM J. Numer. Anal. 56 (2018) 210–227.

[34] B. Jin, R. Lazarov, Z. Zhou, Numerical methods for time-fractional evolution equations with nonsmooth data: A concise overview, Comput. Meth. Appl. Mech. Eng. 346 (2019) 332–358.

[35] Y. Wang, Y. Yan, Y. Yan, A. K. Pani, Higher order time stepping methods for subdiffusion problems based on weighted and shifted Grünwald–Letnikov formulae with nonsmooth data, J. Sci. Comput. 83 (2020) 40. `doi:10.1007/s10915-020-01223-y`.

[36] K. Kwon, D. Sheen, A parallel method for the numerical solution of integro-differential equation with positive memory, Comput. Meth. Appl. Mech. Eng. 192 (2003) 4641–4658.

[37] W. McLean, V. Thomée, Maximum-norm error analysis of a numerical solution via Laplace transformation and quadrature of a fractional-order evolution equation, IMA J. Numer. Anal. 30 (2010) 208–230.

[38] X. Li, T. Tang, C. Xu, Parallel in time algorithm with spectral-subdomain enhancement for Volterra integral equations, SIAM J. Numer. Anal. 51 (2013) 1735–1756.

[39] S.-L. Wu, T. Zhou, Parareal algorithms with local time-integrators for time fractional differential equations, J. Comput.. Phys. 358 (2018) 135–149.

[40] H. Fu, H. Wang, A preconditioned fast parareal finite difference method for space-time fractional partial differential equation, J. Sci. Comput. 78 (2019) 1724–1743.

[41] R. Ke, M. K. Ng, H.-W. Sun, A fast direct method for block triangular Toeplitz-like with tri-diagonal block systems from time-fractional partial differential equations, J. Comput. Phys. 303 (2015) 203–211.

[42] Y.-C. Huang, S.-L. Lei, A fast numerical method for block lower triangular Toeplitz with dense Toeplitz blocks system with applications to time-space fractional diffusion equations, Numer. Algorithms 76 (2017) 605–616.

[43] X. Lu, H.-K. Pang, H.-W. Sun, Fast approximate inversion of a block triangular Toeplitz matrix with applications to fractional sub-diffusion equations, Numer. Linear Algebr. Appl. 22 (2015) 866–882.

[44] X. Lu, H.-K. Pang, H.-W. Sun, S.-W. Vong, Approximate inversion method for time-fractional subdiffusion equations, Numer. Linear Algebr. Appl. 25 (2018) e2132. `doi:10.1002/nla.2132`.

[45] D. Bertaccini, F. Durastante, Limited memory block preconditioners for fast solution of fractional partial differential equations, J. Sci. Comput. 77 (2018) 950–970.

[46] D. Bertaccini, F. Durastante, Block structured preconditioners in tensor form for the all-at-once solution of a finite volume fractional diffusion equation, Appl. Math. Lett. 95 (2019) 92–97.

[47] D. Bertaccini, F. Durastante, Solving mixed classical and fractional partial differential equations using the short-memory principle and approximate inverses, Numer. Algorithms 74 (2017) 1061–1082.

[48] H. A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 13 (1992) 631–644.

[49] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, PA, 2003.

[50] M. F. Murphy, G. H. Golub, A. J. Wathen, A note on preconditioning for indefinite linear systems, SIAM J. Sci. Comput. 21 (2000) 1969–1972.

[51] T. Moroney, Q. Yang, Efficient solution of two-sided nonlinear space-fractional diffusion equations using fast Poisson preconditioners, J. Comput. Phy. 246 (2013) 304–317.

[52] X.-M. Gu, Y.-L. Zhao, X.-L. Zhao, B. Carpentieri, Y.-Y. Huang, A note on parallel preconditioning for the all-at-once solution of Riesz fractional diffusion equations, Numer. Math. Theor. Meth. Appl. to appear (2021) 19 pages.

[53] H.-L. Liao, Y. Yan, J. Zhang, Unconditional convergence of a fast two-level linearized algorithm for semilinear subdiffusion equations, J. Sci. Comput. 80 (2019) 1–25.

[54] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, PA, 1997.