Networks and Heterogeneous Media

http://www.aimspress.com/journal/nhm

NHM, 19(3): 923–939.
DOI: 10.3934/nhm.2024041
Received: 15 July 2024
Revised: 26 August 2024
Accepted: 10 September 2024
Published: 14 September 2024

Research article

A modified domain decomposition spectral collocation method for parabolic partial differential equations

Wei-Hua Luo [1,2,*], Liang Yin [3] and Jun Guo[4]

[1] School of Mathematics and Physics, Hunan University of Arts and Science, Changde, Hunan 415000, P.R. China

[2] Laboratory of Numerical Simulation of Sichuan Provincial Universities, Neijiang Normal University, Neijiang, Sichuan, 641000, P.R. China

[3] College of Mechanical Engineering, Hunan University of Arts and Science, Changde, Hunan, 415000, P.R. China

[4] College of Applied Mathematics, Chengdu University of Information Technology, Chengdu, Sichuan, 610225, P.R. China

* **Correspondence:** Email: huaweiluo2012@163.com.

Abstract: In this paper, utilizing Legendre polynomials as the basis functions in both space and time, we present a modified domain decomposition spectral method for 2-dimensional parabolic partial differential equations. For solving the obtained linear/nonlinear algebraic equations, a dimension expanding preconditioner is applied employing the obtained saddle construction of the coefficient matrix. Numerical examples are given to show the performance of the presented method and the efficiency of the preconditioner.

Keywords: domain decomposition; spectral collocation method; preconditioner; parabolic PDE

1. Introduction

The parabolic partial differential equation (PDE)

$$\frac{\partial u}{\partial t} = k_1^*\frac{\partial^2 u}{\partial x^2} + k_2^*\frac{\partial^2 u}{\partial y^2} + k_3^*\frac{\partial^2 u}{\partial x\partial y} + k_4^*\frac{\partial u}{\partial x} + k_5^*\frac{\partial u}{\partial y} + f^*(u,x,y,t),\ (x,y)\in\Omega,\ t\in(0,T], \tag{1.1}$$

equipped with the initial condition

$$u(x,y,0) = \varphi^*(x,y),\ (x,y)\in\Omega \tag{1.2}$$

and boundary condition

$$u(x,y,t) = \psi^*(x,y,t),\ (x,y) \in \partial\Omega,\ t \in (0,T] \tag{1.3}$$

is a class of important mathematical models, where $\Omega = (\overline{a}, \overline{b}) \times (\tilde{a}, \tilde{b})$, and k_i^*, $i = 1, 2, \cdots, 5$ are all smooth functions of the variables x, y, t. This class of models plays important roles in many engineering problems. For example, when $k_1 = k_2 = 1$, $k_3 = k_4 = k_5 = 0$, and $f^*(u,x,y,t) = g(x,y,t)$, (1.1) becomes the heat conduction equation [1]. Taking $k_1 = k_2 = 1$, $k_3 = k_4 = k_5 = 0$, and $f(u,x,y,t) = au(1-u^2) + g(x,y,t)$, (1.1) represents the Allen-Cahn equation [2]. When describing the Brownian motion of particles, denote by y the velocity of the particles, and let k_1 be a nonzero constant, $k_2 = k_3 = 0$, $k_4 = -y$, $k_5 = \beta y - \gamma x$, and $f(u,x,y,t) = \beta u$. Then, (1.1) corresponds to the Fokker-Planck equation [3]. Moreover, some reaction-diffusion models and convection-diffusion models can also be seen as special cases of (1.1) [4, 5].

Usually, it is difficult to exactly solve Eqs (1.1)–(1.3), and hence finding its numerical solutions is a more popular choice for many researchers. In recent decades, finite difference methods (FDM) and finite element methods (FEM) have been increasingly developed for solving various PDEs, including the parabolic case Eqs (1.1)–(1.3) [6–14]. The most remarkable feature of these two methods is that they can lead to sparse linear equations when some iterative methods (such as Netwon iterative method) are employed to deal with the nonlinear term. However, we all know that these methods are usually inferior to spectral methods in terms of approximating precision.

Spectral methods have attracted extensive attentions since they were first proposed. As an excellent competitor for numerically solving PDEs, they possess great approximation accuracy and are very easy to implement [15–18]. Especially, Lui et al. [19–21] have recently presented spectral discretization in both time and space, and this can greatly accelerate the solving of PDEs containing both space and time factors. However, the drawback of common spectral methods is that the resultant coefficient matrix is dense, which leads to difficulties when solving the corresponding linear equations. This fact is very obvious when spectral methods are used for high-dimensional PDEs or PDEs with large intervals in time and/or space. For remedying the above deficiency, some researchers have considered the combination of domain decomposition with spectral method, and presented some domain decomposition spectral methods [3, 22, 23].

In this paper, we study a novel domain decomposition spectral collocation method employing Legendre polynomials as basis functions. Comparing with the existing techniques, our proposed method enjoys an additional advantage, that is, the resultant linear system has a saddle structure, for which a specially designed preconditioning technique is suitable. In the Sections 3 and 4, we will see that implementing this preconditioner only requires solving the sub-problem $Ax = y$, with A a block diagonal matrix. This means that a lot of CPU time can be saved.

The rest of this paper is as follows. In Section 2, the modified domain decomposition spectral method (shortly, MDDSM) is introduced, and the difference between MDDSM and the conventional domain decomposition spectral method (shortly, CDDSM) is discussed. Later, as an important component, in Section 3, according to [24, 25], we apply a dimension expanding preconditioning technique to iteratively solve the linear system obtained in Section 2. In Section 4, numerical examples are given to show the efficiency of the proposed MDDSM and the preconditioner. Some conclusions are finally drawn in the last section.

2. Description of MDDSM

We first divide the whole interval $[0, T] \times \Omega$ into some big uniform sub-domains. For the fixed positive integers $\overline{p},\ \tilde{p},\ \hat{p}$, let $\overline{q} = (\overline{b} - \overline{a})/\overline{p},\ \tilde{q} = (\tilde{b} - \tilde{a})/\tilde{p},\ \hat{q} = T/\hat{p}$, and we give a uniform partition $[T_0, T_1],\ [T_1, T_2],\ \cdots,\ [T_{\hat{p}-1}, T_{\hat{p}}]$ in time and $\Omega_{i,j} = [\overline{a}_{i-1}, \overline{a}_i] \times [\tilde{a}_{j-1}, \tilde{a}_j],\ i = 1,\ 2,\ \cdots,\ \overline{p},\ j = 1,\ 2,\ \cdots,\ \tilde{p}$ in space, where $T_0 = 0,\ T_{\hat{p}} = T,\ T_k - T_{k-1} = \hat{q},\ k = 1,\ 2,\ \cdots,\ \hat{p}, \overline{a}_0 = \overline{a},\ \overline{a}_{\overline{p}} = \overline{b}$, $\tilde{a}_0 = \tilde{a},\ \tilde{a}_{\tilde{p}} = \tilde{b}$, and $\overline{a}_i - \overline{a}_{i-1} = \overline{q},\ i = 1,\ 2,\ \cdots,\ \overline{p},\ \tilde{a}_j - \tilde{a}_{j-1} = \tilde{q},\ j = 1,\ 2,\ \cdots,\ \tilde{p}$.

Next, at the k_{th} level $[T_{k-1}, T_k],\ k = 1,\ 2,\ \cdots,\ \hat{p}$, and in the big sub-domains $\Omega^k_{i,j} = \Omega_{i,j} \times [T_{k-1}, T_k]$, we do the transformation $x = \frac{\overline{q}}{2}x^* + (i - \frac{1}{2})\overline{q} + \overline{a},\ y = \frac{\tilde{q}}{2}y^* + (j - \frac{1}{2})\tilde{q} + \tilde{a},\ t = \frac{\hat{q}}{2}t^* + (k - \frac{1}{2})\hat{q}$, and denote

$$v(x^*, y^*, t^*) = u(\frac{\overline{q}}{2}x^* + (i - \frac{1}{2})\overline{q} + \overline{a}, \frac{\tilde{q}}{2}y^* + (j - \frac{1}{2})\tilde{q} + \tilde{a}, \frac{\hat{q}}{2}t^* + (k - \frac{1}{2})\hat{q}),$$

$$k_s(x^*, y^*, t^*) = \frac{2\hat{q}}{\overline{q}^2}k^*_s(\frac{\overline{q}}{2}x^* + (i - \frac{1}{2})\overline{q} + \overline{a}, \frac{\tilde{q}}{2}y^* + (j - \frac{1}{2})\tilde{q} + \tilde{a}, \frac{\hat{q}}{2}t^* + (k - \frac{1}{2})\hat{q}),\ s = 1,\ 2,\ 3,\ 4,\ 5,$$

$$f(v(x^*, y^*, t^*), x^*, y^*, t^*) = \frac{\hat{q}}{2}f^*(v(x^*, y^*, t^*), \frac{\overline{q}}{2}x^* + (i - \frac{1}{2})\overline{q} + \overline{a}, \frac{\tilde{q}}{2}y^* + (j - \frac{1}{2})\tilde{q} + \tilde{a}, \frac{\hat{q}}{2}t^* + (k - \frac{1}{2})\hat{q}).$$

Then, by some computations, Eq (1.1) is transformed into an equivalent system of the form

$$\begin{aligned} &\frac{\partial v}{\partial t^*} = k_1\frac{\partial^2 v}{\partial x^{*2}} + k_2\frac{\partial^2 v}{\partial y^{*2}} + k_3\frac{\partial^2 v}{\partial x^*\partial y^*} + k_4\frac{\partial v}{\partial x^*} + k_5\frac{\partial v}{\partial y^*} + f(v, x^*, y^*, t^*), \\ &(x^*, y^*, t^*) \in (-1, 1) \times (-1, 1) \times (-1, 1], \end{aligned} \tag{2.1}$$

where $k_i = k_i(x^*, y^*, t^*),\ i = 1,\ \cdots,\ 5$. For ease of description, we uniformly replace the variables $x^*,\ y^*,\ t^*$ with $x,\ y,\ t$, and rewrite the equivalent equation of (2.1) as

$$\begin{aligned} &\frac{\partial v}{\partial t} = k_1\frac{\partial^2 v}{\partial x^2} + k_2\frac{\partial^2 v}{\partial y^2} + k_3\frac{\partial^2 v}{\partial x\partial y} + k_4\frac{\partial v}{\partial x} + k_5\frac{\partial v}{\partial y} + f(v, x, y, t), \\ &(x, y, t) \in (-1, 1) \times (-1, 1) \times (-1, 1], \end{aligned} \tag{2.2}$$

with $k_i = k_i(x, y, t),\ i = 1,\ \cdots,\ 5$.

Similarly, the initial value (1.2) and boundary condition (1.3) can be equivalently transformed into

$$v(x, y, -1) = \varphi(x, y),\ (x, y) \in [-1, 1] \times [-1, 1] \tag{2.3}$$

and

$$v(x, y, t) = \psi(x, y, t),\ (x, y) \in \partial([-1, 1] \times [-1, 1]),\ t \in (-1, 1], \tag{2.4}$$

where $\varphi(x, y), \psi(x, y, t)$ are known functions.

Now we start to compute the approximating solution of $v(x, y, t)$ in Eqs (2.2)–(2.4) at the first level $[T_0, T_1]$. In each big sub-domain $\Omega^1_{i,j},\ i = 1,\ 2,\ \cdots,\ \overline{p},\ j = 1,\ 2,\ \cdots,\ \tilde{p}$, we choose the Legendre-Gauss-Lobatto type points $x^i_0,\ \cdots,\ x^i_{\overline{N}},\ y^j_0,\ \cdots,\ y^j_{\tilde{N}},\ t^1_0,\ \cdots,\ t^1_{\hat{N}}$ as the nonuniform grid nodes in $x,\ y,\ t$ directions, respectively; this can be seen in Page 20–21 in [15]. It is easy to understand that $x^i_{\overline{N}} = x^{i+1}_0,\ i = 1,\ 2,\ \cdots,\ \overline{p} - 1,\ y^j_{\tilde{N}} = y^{j+1}_0,\ j = 1,\ 2,\ \cdots,\ \tilde{p} - 1$.

For conveniently implementing the coming domain decomposition spectral collocation method, we denote by P_{all} the set of all grid nodes (x_l^i, y_m^j, t_n^1), $i = 1, \cdots, \overline{p}$, $j = 1, \cdots, \tilde{p}$, $l = 0, \cdots, \overline{N}$, $m = 0, \cdots, \tilde{N}$, $n = 0, \cdots, \hat{N}$, and we group this set into the following four subsets.

(g_1) P_{inner} : interior points: (x_l^i, y_m^j, t_n^1), $i = 1, \cdots, \overline{p}$, $j = 1, \cdots, \tilde{p}$, $l = 1, \cdots, \overline{N} - 1$, $m = 1, \cdots, \tilde{N} - 1$, $n = 1, \cdots, \hat{N}$.

(g_2) P_{init} : initial values points: (x_l^i, y_m^j, t_0^1), $i = 1, \cdots, \overline{p}$, $j = 1, \cdots, \tilde{p}$, $l = 0, \cdots, \overline{N}$, $m = 0, \cdots, \tilde{N}$.

(g_3) actual boundary points P_{act} (namely grid points that lie on $\partial\Omega$):

$$(x_0^1, y_m^j, t_n^1),\ j = 1, \cdots, \tilde{p},\ m = 0, \cdots, \tilde{N},\ n = 1, \cdots, \hat{N};$$

$$(x_{\overline{N}}^{\overline{p}}, y_m^j, t_n^1),\ j = 1, \cdots, \tilde{p},\ m = 0, \cdots, \tilde{N},\ n = 1, \cdots, \hat{N};$$

$$(x_l^i, y_0^1, t_n^1),\ i = 1, \cdots, \overline{p},\ l = 0, \cdots, \overline{N},\ n = 1, \cdots, \hat{N};$$

$$(x_l^i, y_{\tilde{N}}^{\tilde{p}}, t_n^1),\ i = 1, \cdots, \overline{p},\ l = 0, \cdots, \overline{N},\ n = 1, \cdots, \hat{N}.$$

(g_4) ghost boundary points P_{gho}: the interior grid points which lie on $\partial(\Omega_{i,j}^1)$, namely $P_{gho} = P_{all} - (P_{inner} \cup P_{init} \cup P_{act})$. Specifically, when $l = 0$ or $l = \overline{N}$, (x_l^i, y_m^j, t_n^1), they are called a x-type ghost boundary points (shortly, x-type gbp), alternatively, when $m = 0$ or $m = \tilde{N}$, (x_l^i, y_m^j, t_n^1), they are called a y-type ghost boundary points (shortly, y-type gbp). (Figure 1 shows an illustrative example of the ghost boundary points P_{gho}, where the green line represents an x-type gbp, and the red shows a y-type gbp.)

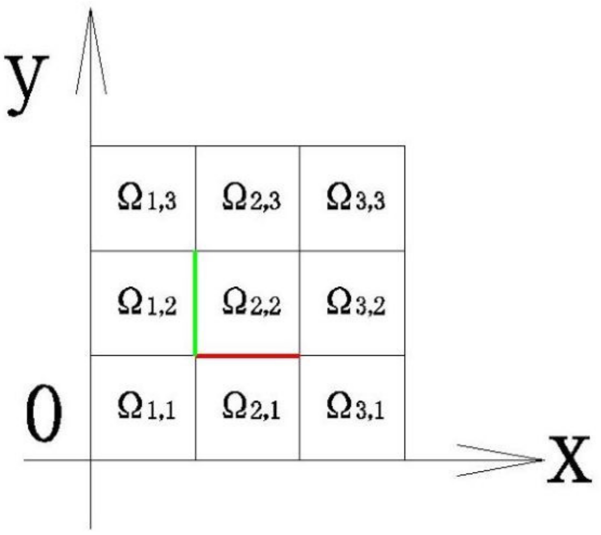

Figure 1. An illustration of the ghost boundary points, green: x-type gbp; red: y-type gbp.

Moreover, in order to ensure the number of unknowns is the same as that of equations, we should employ the fact that the left-half derivative is equivalent to the right-half derivative, and consequently, we impose the following ghost boundary conditions on each ghost boundary points:

(a) the x-type ghost boundary condition (for x-type gbp):

$$
\begin{aligned}
& v(x^i_{\overline{N}}, y^j_m, t^1_n) = v^{gx}_{i,(j-1)\widetilde{N}+m,n}, \quad v(x^{i+1}_0, y^j_m, t^1_n) = v^{gx}_{i,(j-1)\widetilde{N}+m,n}, \\
& \frac{dv}{dx^-}|_{(x^i_{\overline{N}}, y^j_m, t^1_n)} = \frac{dv}{dx^+}|_{(x^{i+1}_0, y^j_m, t^1_n)}, \\
& i = 1, \cdots, \overline{p}-1, \\
& j = 1, \cdots, \widetilde{p}-1, \ m = 1, \cdots, \widetilde{N}; \text{ or } \ j = \widetilde{p}, \ m = 1, \cdots, \widetilde{N}-1, \\
& n = 1, \cdots, \hat{N}.
\end{aligned} \tag{2.5}
$$

(b) the y-type ghost boundary condition (for y-type gbp):

$$
\begin{aligned}
& v(x^i_l, y^j_{\widetilde{N}}, t^1_n) = v^{gy}_{(i-1)(\overline{N}-1)+l,j,n}, \quad v(x^i_l, y^{j+1}_0, t^1_n) = v^{gy}_{(i-1)(\overline{N}-1)+l,j,n}, \\
& \frac{dv}{dy^-}|_{(x^i_l, y^j_{\widetilde{N}}, t^1_n)} = \frac{dv}{dy^+}|_{(x^i_l, y^{j+1}_0, t^1_n)}, \\
& i = 1, \cdots, \overline{p}, \ l = 1, \cdots, \overline{N}-1, \ j = 1, \cdots, \widetilde{p}-1, \ n = 1, \cdots, \hat{N},
\end{aligned} \tag{2.6}
$$

where $v^{gx}_{i,(j-1)\widetilde{N}+m,n}$, $v^{gy}_{(i-1)(\overline{N}-1)+l,j,n}$ are all undetermined unknowns.

Let

$$
v_h(x, y, t) = \sum_{l=0,m=0,n=0}^{\overline{N},\widetilde{N},\hat{N}} L_l(x)L_m(y)L_n(t)v^{i,j,1}_{l,m,n} \tag{2.7}
$$

be the approximation of the exact solution $v(x, y, t)$ of (2.2)-(2.4), where $L_l(x)$, $L_m(y)$ and $L_n(t)$ are all Legendre polynomials with degrees of l, m and n, respectively, and $v^{i,j,1}_{l,m,n}$ are the unknowns to be determined. Then, in each sub space $\Omega^1_{i,j}$, taking $x = x^i_l$, $y = y^j_m$, $t = t^1_n$ as the collocation points, and substituting (2.7) into Eqs (2.2)–(2.6), we can obtain the corresponding algebraic equations. Concretely, MDDSM can be listed as the following algorithm.

Algorithm 1. MDDSM algorithm.

```
•step 1. For all the interior points $P \in P_{inner}$, substitute (2.7) into the Eq (2.2);
•step 2. For all the initial points $P \in P_{init}$, substitute (2.7) into the Eq (2.3);
•step 3. For all the actual boundary points $P \in P_{act}$, substitute (2.7) into the Eqs (2.4);
•step 4. For all the ghost boundary points $P \in P_{gho}$, substitute (2.7) into the ghost boundary conditions (2.5) (for $x$-type gbp) and (2.6) (for $y$-type gbp).
```

For describing the resultant linear equations, we define all the unknowns as $\mathbf{v}_h = (\mathbf{v}^a_h, \mathbf{v}^g_h)$, with

$$
\begin{aligned}
\mathbf{v}^a_h &= (\mathbf{v}^a_{1,1}, \mathbf{v}^a_{1,2}, \cdots, \mathbf{v}^a_{1,\widetilde{p}}, \mathbf{v}^a_{2,1}, \mathbf{v}^a_{2,2}, \cdots, \mathbf{v}^a_{2,\widetilde{p}}, \cdots, \mathbf{v}^a_{\overline{p},1}, \mathbf{v}^a_{\overline{p},2}, \cdots, \mathbf{v}^a_{\overline{p},\widetilde{p}}), \\
\mathbf{v}^g_h &= (\mathbf{v}^g_{1,1}, \mathbf{v}^g_{1,2}, \cdots, \mathbf{v}^g_{1,\widetilde{p}}, \mathbf{v}^g_{2,1}, \mathbf{v}^g_{2,2}, \cdots, \mathbf{v}^g_{2,\widetilde{p}}, \cdots, \mathbf{v}^g_{\overline{p},1}, \mathbf{v}^g_{\overline{p},2}, \cdots, \mathbf{v}^g_{\overline{p},\widetilde{p}-1}),
\end{aligned} \tag{2.8}
$$

where $\mathbf{v}^a_{i,j}$ is the solution vector involved in $\Omega^1_{i,j}$, and $\mathbf{v}^g_{i,j}$ is the ghost value vector included in Eqs (2.5) and (2.6) on $\partial\Omega^1_{i,j}$. That is,

$$
\mathbf{v}^a_{i,j} = (v^{i,j,1}_{0,0,0}, v^{i,j,1}_{0,0,1}, \ldots, v^{i,j,1}_{0,0,\hat{N}}, v^{i,j,1}_{0,1,0}, v^{i,j,1}_{0,1,1}, \ldots, v^{i,j,1}_{0,1,\hat{N}} \ldots, \ldots, v^{i,j,1}_{0,\tilde{N},0}, v^{i,j,1}_{0,\tilde{N},1}, \ldots, v^{i,j,1}_{0,\tilde{N},\hat{N}},
$$

$$v^{i,j,1}_{1,0,0},\ v^{i,j,1}_{1,0,1},\ \ldots,\ v^{i,j,1}_{1,0,\hat{N}},\ v^{i,j,1}_{1,1,0},\ v^{i,j,1}_{1,1,1},\ \ldots,\ v^{i,j,1}_{1,1,\hat{N}}\ \ldots,\ \ldots,\ v^{i,j,1}_{1,\tilde{N},0},\ v^{i,j,1}_{1,\tilde{N},1},\ \ldots,\ v^{i,j,1}_{1,\tilde{N},\hat{N}},$$
$$\ldots,\ \ldots,\ \ldots,\ \ldots,\ \ldots,\ \ldots,$$
$$v^{i,j,1}_{\overline{N},0,0},\ v^{i,j,1}_{\overline{N},0,1},\ \ldots,\ v^{i,j,1}_{\overline{N},0,\hat{N}},\ v^{i,j,1}_{\overline{N},1,0},\ v^{i,j,1}_{\overline{N},1,1},\ \ldots,\ v^{i,j,1}_{\overline{N},1,\hat{N}}\ \ldots,\ \ldots,\ v^{i,j,1}_{\overline{N},\tilde{N},0},\ v^{i,j,1}_{\overline{N},\tilde{N},1},\ \ldots,\ v^{i,j,1}_{\overline{N},\tilde{N},\hat{N}}),$$
$$i = 1,\ \cdots,\ \overline{p},\ j = 1,\ \cdots,\ \tilde{p},$$

and

$$\mathbf{v}^g_{i,j} = (v^{gx}_{i,(j-1)\tilde{N}+1,1},\ v^{gx}_{i,(j-1)\tilde{N}+1,2},\ \ldots,\ v^{gx}_{i,(j-1)\tilde{N}+1,\hat{N}}, v^{gx}_{i,(j-1)\tilde{N}+2,1},\ v^{gx}_{i,(j-1)\tilde{N}+2,2},\ \ldots,\ v^{gx}_{i,(j-1)\tilde{N}+2,\hat{N}},$$
$$\ldots,\ \ldots,\ v^{gx}_{i,j\tilde{N},1},\ v^{gx}_{i,j\tilde{N},2},\ \ldots,\ v^{gx}_{i,j\tilde{N},\hat{N}},\ v^{gy}_{(i-1)(\overline{N}-1)+1,j,1},\ v^{gy}_{(i-1)(\overline{N}-1)+1,j,2},$$
$$\ldots,\ v^{gy}_{(i-1)(\overline{N}-1)+1,j,\hat{N}},\ v^{gy}_{(i-1)(\overline{N}-1)+2,j,1},\ v^{gy}_{(i-1)(\overline{N}-1)+2,j,2},\ \ldots,\ v^{gy}_{(i-1)(\overline{N}-1)+2,j,\hat{N}},$$
$$\ldots,\ \ldots,\ v^{gy}_{i(\overline{N}-1),j,1},\ v^{gy}_{i(\overline{N}-1),j,2},\ \ldots,\ v^{gy}_{i(\overline{N}-1),j,\hat{N}}),\ \text{when}\ \ i = 1,\ \cdots,\ \overline{p}-1,\ j = 1,\ \cdots,\ \tilde{p}-1,$$

$$\mathbf{v}^g_{i,j} = (v^{gx}_{i,(j-1)\tilde{N}+1,1},\ v^{gx}_{i,(j-1)\tilde{N}+1,2},\ \ldots,\ v^{gx}_{i,(j-1)\tilde{N}+1,\hat{N}},\ v^{gx}_{i,(j-1)\tilde{N}+2,1},\ v^{gx}_{i,(j-1)\tilde{N}+2,2},\ \ldots,\ v^{gx}_{i,(j-1)\tilde{N}+2,\hat{N}},$$
$$\ldots,\ \ldots,\ v^{gx}_{i,j\tilde{N}-1,1},\ v^{gx}_{i,j\tilde{N}-1,2},\ \ldots,\ v^{gx}_{i,j\tilde{N}-1,\hat{N}}),\ \text{when}\ \ i = 1,\ \cdots,\ \overline{p}-1,\ j = \tilde{p},$$

$$\mathbf{v}^g_{i,j} = (v^{gy}_{(i-1)(\overline{N}-1)+1,j,1},\ v^{gy}_{(i-1)(\overline{N}-1)+1,j,2},\ \ldots,\ v^{gy}_{(i-1)(\overline{N}-1)+1,j,\hat{N}},\ v^{gy}_{(i-1)(\overline{N}-1)+2,j,1},\ v^{gy}_{(i-1)(\overline{N}-1)+2,j,2},$$
$$\ldots, v^{gy}_{(i-1)(\overline{N}-1)+2,j,\hat{N}},\ \ldots,\ \ldots,\ v^{gy}_{i(\overline{N}-1),j,1},\ v^{gy}_{i(\overline{N}-1),j,2},\ \ldots, v^{gy}_{i(\overline{N}-1),j,\hat{N}}),\ \text{when}\ i = \overline{p},\ j = 1,\ \cdots,\ \tilde{p}-1.$$

Then, by deleting the error term caused by the approximating solution function $v_h(x, y, t)$ in (2.7), Algorithm 1 will result in $M = \overline{p}\tilde{p}(\overline{N}+1)(\tilde{N}+1)(\hat{N}+1) + (\overline{p}-1)(\tilde{p}-1)(\tilde{N}\hat{N} + (\overline{N}-1)\hat{N}) + (\overline{p}-1)(\tilde{N}-1)\hat{N} + (\tilde{p}-1)(\overline{N}-1)$ algebraic equations

$$g_i(\mathbf{v}_h) = 0,\ i = 1,\ 2,\ \cdots,\ M. \tag{2.9}$$

When $f(v, x, y, t)$ in Eq (2.2) is a linear term of v, it is easy to know that Eq (2.9) becomes the following linear system

$$H\mathbf{v}_h^T = b, \tag{2.10}$$

where the coefficient matrix H has a saddle structure of the form

$$H = \begin{pmatrix} A & B \\ C & \mathbf{0} \end{pmatrix}.$$

Concretely, $A = diag(A_{1,1},\ A_{2,2},\ \cdots,\ A_{N_1,N_1})$, $N_1 = \overline{p}\tilde{p}$, $size(A_i) = (\overline{N}+1)(\tilde{N}+1)(\hat{N}+1) \times (\overline{N}+1)(\tilde{N}+1)(\hat{N}+1)$, $i = 1,\ 2,\ \cdots,\ N_1$, where $size(A)$ expresses the numbers of rows and columns of A, and B is a full column rank matrix that has at most one entry per row equal to -1, at least two entries per column equal to -1, and at most four entries per column equal to -1. Specifically, when k_i, $i = 1,\ \cdots,\ 5$ in (2.2) are all constants, there is $A_{1,1} = A_{2,2} = \cdots = A_{N_1,N_1}$. The left one in Figure 2 shows an illustration of H taking $\overline{a} = \tilde{a} = 0, \overline{b} = \tilde{b} = T = 12, \overline{N} = \tilde{N} = \hat{N} = 8, \overline{q} = \tilde{q} = \hat{q} = 2$.

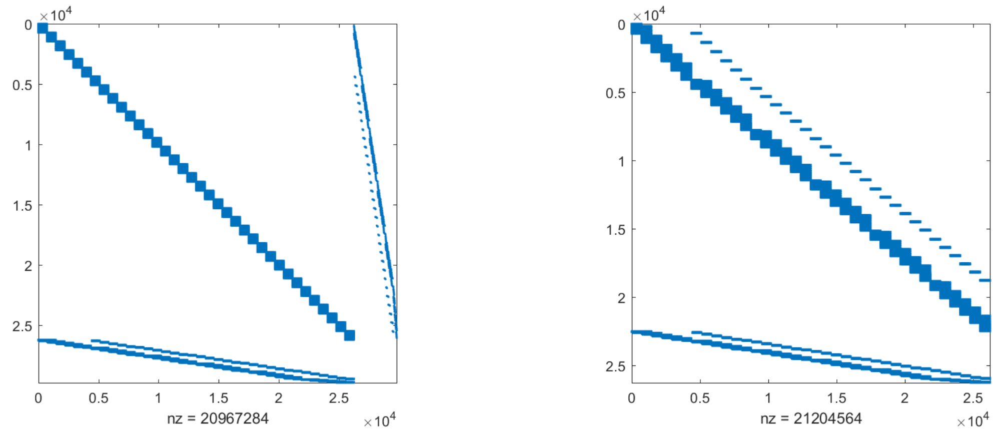

Figure 2. left: structure of H in (2.10); right: structure of $\hat{H}$ in (2.13). $\bar{a} = \tilde{a} = 0$, $\bar{b} = \tilde{b} = T = 12$, $\overline{N} = \tilde{N} = \hat{N} = 8$, $\bar{q} = \tilde{q} = \hat{q} = 2$.

When $f(v, x, y, t)$ in Eq (2.2) is a nonlinear term of v, then, after dealing with the nonlinear algebraic equations by employing some iterative methods (for instance, the classic Newton iterative methods), (2.9) will still result in the saddle linear system (2.10).

Considering we have computed the values of $v^{i,j,k-1}_{l,m,n}$, $l = 0, \cdots, \overline{N}$, $m = 0, \cdots, \tilde{N}$, $n = 0, \cdots, \hat{N}$, $i = 1, \cdots, \bar{p}$, $j = 1, \cdots, \tilde{p}$, we now substitute the points $(x^i_l, y^j_m, t^{k-1}_{\hat{N}})$, $i = 1, \cdots, \bar{p}$, $j = 1, \cdots, \tilde{p}$, $l = 0, \cdots, \overline{N}$, $m = 0, \cdots, \tilde{N}$ into

$$v_h(x, y, t) = \sum_{l=0,m=0,n=0}^{\overline{N},\tilde{N},\hat{N}} L_l(x)L_m(y)L_n(t)v^{i,j,k-1}_{l,m,n}$$

to get $v_h(x^i_l, y^j_m, t^{k-1}_{\hat{N}})$. These are also the values of $v_h(x^i_l, y^j_m, t^k_0)$ since $(x^i_l, y^j_m, t^{k-1}_{\hat{N}}) = (x^i_l, y^j_m, t^k_0)$, $i = 1, \cdots, \bar{p}$, $j = 1, \cdots, \tilde{p}$, $l = 0, \cdots, \overline{N}$, $m = 0, \cdots, \tilde{N}$. Consequently, in this way, we obtain the initial values at the k_{th} level $[T_{k-1}, T_k]$. By virtue of this group of initial values, we can solve

$$v_h(x, y, t) = \sum_{l=0,m=0,n=0}^{\overline{N},\tilde{N},\hat{N}} L_l(x)L_m(y)L_n(t)v^{i,j,k}_{l,m,n}$$

using the same idea of Algorithm 1.

For investigating the difference between MDDSM and CDDSM, here we additionally give the idea of CDDSM and the resultant algebraic equations. The actual efficiency of these two algorithms will be shown in the numerical examples in Section 4.

In the CDDSM, the ghost boundary conditions are replaced by

(c) the x-type ghost boundary condition (for x-type gbp):

$$\begin{aligned} &v(x^i_{\overline{N}}, y^j_m, t^1_n) = v(x^{i+1}_0, y^j_m, t^1_n), \ \frac{dv}{dx^-}\Big|_{(x^i_{\overline{N}}, y^j_m, t^1_n)} = \frac{dv}{dx^+}\Big|_{(x^{i+1}_0, y^j_m, t^1_n)}, \\ &i = 1, \cdots, \overline{p} - 1, \\ &j = 1, \cdots, \widetilde{p} - 1, \ m = 1, \cdots, \widetilde{N}; \ j = \widetilde{p}, \ m = 1, \cdots, \widetilde{N} - 1, \\ &n = 1, \cdots, \hat{N}. \end{aligned} \tag{2.11}$$

(d) the y-type ghost boundary condition (for y-type gbp):

$$\begin{gathered} v(x_l^i, y_{\widetilde{N}}^j, t_n^1) = v(x_l^i, y_0^{j+1}, t_n^1),\ \frac{dv}{dy^-}|_{(x_l^i, y_{\widetilde{N}}^j, t_n^1)} = \frac{dv}{dy^+}|_{(x_l^i, y_0^{j+1}, t_n^1)}, \\ i = 1,\ \cdots,\ \overline{p},\ l = 1,\ \cdots,\ \overline{N}-1,\ j = 1,\ \cdots,\ \widetilde{p}-1,\ n = 1,\ \cdots,\ \hat{N}. \end{gathered} \tag{2.12}$$

Then, CDDSM can be expressed as follows

Algorithm 2. CDDSM algorithm.

- •*step* 1. For all the interior points $P \in P_{inner}$, substitute (2.7) into the Eqs (2.2);
- •*step* 2. For all the initial points $P \in P_{init}$, substitute (2.7) into the Eqs (2.3);
- •*step* 3. For all the actual boundary points $P \in P_{act}$, substitute (2.7) into the Eqs (2.4);
- •*step* 4. For all the ghost boundary points $P \in P_{gho}$, substitute (2.7) into the ghost boundary conditions (2.11) (for x-type gbp) and (2.12) (for y-type gbp).

From these two algorithms, we can clearly find the unique difference between MDDSM and CDDSM is the ghost boundary conditions in step 4. In MDDSM, the participation of ghost unknowns $v^{gx}_{i,(j-1)\widetilde{N}+m,n}$, $v^{gy}_{(i-1)(\overline{N}-1)+l,j,n}$ leads to a saddle structure of the coefficient matrix H, for which we will see that a dimension expanding preconditioner is suitable in the next section. However, in CDDSM, the resulting linear system is

$$\hat{H}\mathbf{v}_h^a = \hat{b}, \tag{2.13}$$

where $\mathbf{v}_h^a$ is the unknown vector, and the coefficient matrix $\hat{H}$ is no longer a saddle matrix. The right figure in Figure 2 shows an illustration of $\hat{H}$ corresponding to $\overline{a} = \tilde{a} = 0$, $\overline{b} = \tilde{b} = T = 12$, $\overline{N} = \tilde{N} = \hat{N} = 8$, $\overline{q} = \tilde{q} = \hat{q} = 2$.

3. The application of dimension expanding preconditioning technique on the saddle problem of MDDSM

In this section, we introduce a preconditioning technique for quickly solving the saddle system (2.10). This kind of preconditioning technique was first presented by Luo et al. [24, 25]. When this technique is used combined with some Krylov subspace method such as the GRMES method [26], an advantage is that, in each iteration, one only needs to solve a sub problem $Ax = r$, where A is a block diagonal matrix.

For ease of description, we rewrite (2.10) as

$$\begin{pmatrix} A & B \\ C & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{r}_b^1 \\ \mathbf{r}_b^2 \end{pmatrix}, \tag{3.1}$$

and suppose the size of A, B, C are respectively $m \times m$, $m \times n$, and $n \times m$. Clearly, (3.1) can be equivalently augmented as

$$\tilde{H}\tilde{\mathbf{u}} = \tilde{\mathbf{r}}_b, \tag{3.2}$$

where

$$\tilde{H} = \begin{pmatrix} I & \mathbf{0} & I \\ B & A & \mathbf{0} \\ I & C & I \end{pmatrix}, \ \tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_1 \\ \mathbf{u}_3 \end{pmatrix}, \ \tilde{\mathbf{r}}_b = \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_b^1 \\ \mathbf{r}_b^2 \end{pmatrix},$$

with I an $n \times n$ identity matrix, and $\mathbf{u}_2 = -\mathbf{u}_3$.

Denoting by $\tilde{A} = A + \frac{\tilde{\alpha}}{\tilde{\alpha}-1} BC$, and letting $\tilde{\alpha}$ be a chosen parameter, we take the preconditioner as

$$P_{DE} = \begin{pmatrix} I & \mathbf{0} & \tilde{\alpha} I \\ B & A & \tilde{\alpha} B \\ \mathbf{0} & C & (1-\tilde{\alpha}) I \end{pmatrix}. \tag{3.3}$$

This preconditioner P_{DE} can be implemented by

$$P_{DE}^{-1} = Q_3^{-1} Q_2 Q_1, \tag{3.4}$$

where

$$Q_1 = \begin{pmatrix} I & & \\ & I & \frac{\tilde{\alpha}}{\tilde{\alpha}-1} B \\ & & I \end{pmatrix}, \ Q_2 = \begin{pmatrix} I & & \\ & I & \\ I & & I \end{pmatrix}, \ Q_3 = \begin{pmatrix} I & \mathbf{0} & \tilde{\alpha} I \\ B & \tilde{A} & \mathbf{0} \\ I & C & I \end{pmatrix},$$

and Q_3^{-1} can be further expressed as

$$\begin{aligned} Q_3^{-1} &= \begin{pmatrix} \mathbf{0} & \frac{\tilde{\alpha}}{1-\tilde{\alpha}} C & \frac{\tilde{\alpha}}{\tilde{\alpha}-1} I \\ \mathbf{0} & I & \mathbf{0} \\ I & \frac{1}{\tilde{\alpha}-1} C & \frac{1}{1-\tilde{\alpha}} I \end{pmatrix} \begin{pmatrix} I & & \\ & (\tilde{A} + \frac{\tilde{\alpha}}{1-\tilde{\alpha}} BC)^{-1} & \\ & & I \end{pmatrix} \begin{pmatrix} \frac{1}{\tilde{\alpha}} I & \mathbf{0} & \mathbf{0} \\ \frac{1}{\tilde{\alpha}-1} B & I & \frac{\tilde{\alpha}}{1-\tilde{\alpha}} B \\ \frac{-1}{\tilde{\alpha}} I & \mathbf{0} & I \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{0} & \frac{\tilde{\alpha}}{1-\tilde{\alpha}} C & \frac{\tilde{\alpha}}{\tilde{\alpha}-1} I \\ \mathbf{0} & I & \mathbf{0} \\ I & \frac{1}{\tilde{\alpha}-1} C & \frac{1}{1-\tilde{\alpha}} I \end{pmatrix} \begin{pmatrix} I & & \\ & A^{-1} & \\ & & I \end{pmatrix} \begin{pmatrix} \frac{1}{\tilde{\alpha}} I & \mathbf{0} & \mathbf{0} \\ \frac{1}{\tilde{\alpha}-1} B & I & \frac{\tilde{\alpha}}{1-\tilde{\alpha}} B \\ \frac{-1}{\tilde{\alpha}} I & \mathbf{0} & I \end{pmatrix}. \end{aligned} \tag{3.5}$$

Remark 1. From Eq (3.5) we see that when implementing the preconditioner P_{DE} in Eq (3.3), some parallel computations can be employed. In fact, when computing $P_{DE}^{-1} r$, one of important step is to solve $A^{-1} v$, and this can be independently completed by multiple processors, considering the fact that A is a block diagonal matrix.

4. Numerical results

In this section, we give some examples to investigate the actual efficiency of the presented MDDSM and the preconditioner P_{DE}. For reflecting the superiority, we compare MDDSM with CDDSM from the perspectives of approximating accuracy and the costed CPU time. When dealing with the nonlinear PDEs (Example 3), we transform the resultant nonlinear algebraic equations into linear equations by the classical Newton iterative method, always taking an initial value vector $x = ones(N)$. All the obtained linear algebraic equations are solved by the popular GMRES method [26]. In MDDSM, the GMRES method is equipped with the preconditioner P_{DE}, and in CDDSM, the common ILU preconditioner is used. All these examples are implemented using

MATLAB R2016(a) on a PC equipped with an Intel(R)Core(TM)i5-8265U processor (CPU@1.60GHz).

Table 1. Numerical results of Example 1: $c = 1,\ T = 2,\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N}$, $[\overline{a}, \overline{b}] = [0, 4]$.

$MDDSM + DE(\tilde{\alpha} = 2,\ tol = 1e-15)$				$MDDSM + ILU(tol = 1e-6)$	
$\overline{N}$	size	iter	error	iter	error
8	3148	151	6.2496e-7	*	*
10	5694	177	1.9817e-9	*	*
12	9328	239	4.1875e-12	*	*
14	14242	260	8.1180e-13	*	*
16	20628	298	2.0273e-13	*	*

The following Table 1 is given only to investigate the resultant accuracy of the presented MDDSM in both time and space, hence, the convergence tolerance for GMRES (in the left column) is taken as $1e-15$, and the maximum number of iterations is set equal to 1000. Furthermore, in this table, we additionally investigate the efficiency of the common ILU preconditioner (setup.type = 'crout'; setup.milu = 'row'; setup.droptol = 0.1) for MDDSM with convergence tolerance $1e-6$, and the results are reported in the right column.

Table 2. Numerical results of Example 1: $c = 1,\ T = 2,\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N} = 8$.

	$MDDSM + DE(\tilde{\alpha} = 2)$					CDDSM+ILU				
$[\overline{a}, \overline{b}]$	size	iter	time1(s)	time2(s)	error	size	iter	time1(s)	time2(s)	error
$[0, 6]$	7265	89	0.4287	1.0642	2.1575e-6	6561	*	0.4643	*	*
$[0, 12]$	29804	100	3.0688	4.8494	1.6649e-6	26244	*	2.9094	*	*
$[0, 18]$	67625	101	13.1635	10.3642	1.5477e-6	59049	*	13.4892	*	*
$[0, 24]$	120728	114	39.9983	22.8669	1.6715e-6	104976	*	40.0834	*	*
$[0, 30]$	189113	117	99.6043	34.7329	1.3065e-6	164025	*	100.1375	*	*

In the Tables 2–8, 'MDDSM+DE' means that the PDE is solved by the MDDSM method, and the obtained linear system is solved by the GMRES method equipped with DE preconditioner, 'CDDSM+ILU' means that the PDE is solved by the CDDSM method, and the obtained linear system is solved by the GMRES method equipped with ILU preconditioner (setup.type = 'crout'; setup.milu = 'row'; setup.droptol = 0.1), and 'CDDSM+direct' means that the PDE is solved by the CDDSM method, and the obtained linear system is solved by the direct method $x = H\backslash b$. This direct method is listed only to check the accuracy of CDDSM. In all the runs in Tables 2–8, the convergence tolerance for GMRES is taken as $1e-6$, and the maximum number of iterations is still set equal to 1000. 'size' shows the number of rows of the coefficient matrix H, 'iter' denotes the number of iterations in the GMRES method, 'time1(s)' and 'time2(s)' respectively denote the time for discretizing the PDE (2.2)–(2.4) using MDDSM/ CDDSM and the time for solving the resultant linear system (2.10) (using P_{DE}), (2.13) (using ILU preconditioner), or (2.13) (using the direct method). '*' indicates that the GMRES method fails. 'error' is the error between the exact solution and the approximating solution at all the interior mesh points, initial value mesh points, and actual boundary mesh points, and this is computed using the infinite norm of a vector.

Table 3. Numerical results of Example 1: $c = 0.001,\ T = 2,\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N} = 8$.

	$MDDSM + DE(\tilde{\alpha} = 1.52)$					CDDSM+ILU				
$[\overline{a}, \overline{b}]$	size	iter	time1(s)	time2(s)	error	size	iter	time1(s)	time2(s)	error
$[0, 6]$	7265	14	0.4258	0.1927	1.7299e-6	6561	*	0.4319	*	*
$[0, 12]$	29804	15	3.0224	0.8419	1.8874e-6	26244	*	3.2308	*	*
$[0, 18]$	67625	15	12.9018	1.8050	1.8976e-6	59049	*	13.3429	*	*
$[0, 24]$	120728	15	40.2507	3.2387	1.9454e-6	104976	*	40.8738	*	*
$[0, 30]$	189113	15	100.3354	5.4993	1.9519e-6	164025	*	101.0340	*	*

Table 4. Numerical results of Example 1: $c = 1,\ T = 50,\ [\overline{a}, \overline{b}] = [0, 8],\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N} = 8$.

	$MDDSM + DE(\tilde{\alpha} = 2)$				$CDDSM + direct$		
T	iter	time1	time2	error	time1	time2	error
$[0, 2]$	97	0.8103	2.4002	1.6911e-6	1.0984	4.4612	6.3931e-7
$[2, 4]$	97	0.3053	2.1951	1.7005e-6	0.3831	4.5718	7.4927e-7
$[4, 6]$	97	0.2836	2.2318	2.2183e-6	0.3862	4.5986	8.2930e-7
$[6, 8]$	97	0.2923	2.1227	2.1574e-6	0.3332	4.5491	7.2504e-7
$[8, 10]$	97	0.2746	2.1236	1.7716e-6	0.3264	4.5788	6.8281e-7
$[10, 12]$	97	0.2747	2.1191	2.2123e-6	0.3123	4.5817	7.7594e-7
$[12, 14]$	97	0.2804	2.1186	2.1558e-6	0.3929	4.5264	7.1809e-7
$[14, 16]$	97	0.2712	2.1177	1.9980e-6	0.3880	4.5511	7.2365e-7
$[16, 18]$	97	0.2851	2.1917	2.0841e-6	0.3611	4.5636	7.5500e-7
$[18, 20]$	97	0.2968	2.3040	2.2315e-6	0.3194	4.5535	7.3403e-7
$[20, 22]$	97	0.2939	2.1958	2.1634e-6	0.3002	4.6308	8.0671e-7
$[22, 24]$	97	0.2979	2.3162	1.9259e-6	0.3416	4.6268	7.2688e-7
$[24, 26]$	97	0.3066	2.2550	2.3538e-6	0.3195	4.4624	7.3930e-7
$[26, 28]$	97	0.3100	2.2219	2.2830e-6	0.3712	4.4442	8.2556e-7
$[28, 30]$	97	0.3155	2.1853	1.8687e-6	0.3398	4.5492	7.2441e-7
$[30, 32]$	97	0.2975	2.1825	2.4293e-6	0.3175	4.5575	6.8583e-7
$[32, 34]$	97	0.3091	2.1738	2.3607e-6	0.3128	4.4386	7.7852e-7
$[34, 36]$	97	0.3007	2.1840	1.8962e-6	0.3278	4.4459	7.1699e-7
$[36, 38]$	97	0.2948	2.1739	2.4550e-6	0.3256	4.4715	7.2006e-7
$[38, 40]$	97	0.2976	2.2187	2.3881e-6	0.3252	4.5395	7.5564e-7
$[40, 42]$	97	0.2894	2.1785	1.9015e-6	0.3217	4.5934	7.3293e-7
$[42, 44]$	97	0.2967	2.1920	2.4288e-6	0.3271	4.5325	8.0508e-7
$[44, 46]$	97	0.3018	2.3206	2.3647e-6	0.3350	4.4296	7.2659e-7
$[46, 48]$	97	0.3103	2.2826	1.8804e-6	0.3272	4.5711	7.4001e-7
$[48, 50]$	97	0.2958	2.1941	2.3547e-6	0.3235	4.5819	8.2593e-7

Moreover, in Tables 7 and 8, ‘iter(Newton)’ is the number of iterations in the Newton iterative method, and 'iter(gmres)' is the sum of all the iterations of GMRES involved in the whole Newton iterative method.

For convenience, by taking $\overline{a} = \tilde{a}, \overline{b} = \tilde{b}$ in the space domain Ω, we consider the following examples.

Table 5. Numerical results of Example 2: $T = 36,\ [\overline{a}, \overline{b}] = [0, 6],\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N} = 7$.

	$MDDSM + DE(\tilde{\alpha} = 1.52)$			
T	iter	time1(s)	time2(s)	error
$[0,2]$	15	4.6895	0.8031	1.5455e-6
$[2,4]$	59	4.9846	1.8762	6.3147e-6
$[4,6]$	70	4.8806	2.1244	8.6783e-6
$[6,8]$	62	4.7776	1.8990	4.0616e-6
$[8,10]$	60	4.9584	1.7632	5.8417e-6
$[10,12]$	70	4.8203	2.1371	8.0284e-6
$[12,14]$	54	4.7827	1.6499	3.2658e-6
$[14,16]$	66	4.7457	1.9148	7.0381e-6
$[16,18]$	70	4.7627	2.0677	9.9410e-6
$[18,20]$	59	4.8260	1.7937	9.2254e-6
$[20,22]$	67	5.0983	2.0591	1.3391e-5
$[22,24]$	57	5.1899	1.8296	4.9781e-6
$[24,26]$	70	5.0206	2.1269	6.8350e-6
$[26,28]$	70	4.9481	2.1788	8.7829e-6
$[28,30]$	59	5.0516	1.9088	3.0236e-6
$[30,32]$	70	4.8576	2.1034	7.4539e-6
$[32,34]$	60	5.0004	1.8663	8.2583e-6
$[34,36]$	57	5.0687	1.8578	4.3468e-6

Example 1. (the heat conduction equation):

$k_1^* = k_2^* = c,\ k_3^* = k_4^* = k_5^* = 0,\ f(u, x, y, t) = cos(x+y+t)+2csin(x+y+t),\ u(x, y, t) = sin(x+y+t)+2.$

Example 2. (with variable coefficients):

$k_1^* = 0.5exp(x + 2y + 5t),\ k_2^* = cos(x - y + 2t)/exp(x + y + 2t),\ k_3^* = 0.1,\ k_4^* = 0.2,\ k_5^* = -0.1,\ f(u, x, y, t) = 20u + g(x, y, t),\ u(x, y, t) = sin(x + y + t) + 2.$

Example 3. (the Allen-Cahn equation):

$k_1^* = k_2^* = 1,\ k_3^* = k_4^* = k_5^* = 0,\ f(u, x, y, t) = \frac{1}{\varepsilon}u(1 - u^2) + g(x, y, t),\ g(x, y, t) = cos(x + y + t) + 2sin(x + y + t) - \frac{1}{\varepsilon}(sin(x + y + t) + 2 - (sin(x + y + t) + 2)^3),\ u(x, y, t) = sin(x + y + t) + 2.$

Table 6. Numerical results of Example 2: $T = 2,\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N} = 7$.

	$MDDSM + DE(\tilde{\alpha} = 1.8)$			
$[\overline{a}, \overline{b}]$	iter	time1(s)	time2(s)	error
$[0, 4]$	13	1.1013	0.4274	1.6036e-6
$[0, 6]$	14	4.6144	0.9936	1.9553e-6
$[0, 8]$	15	14.3932	1.9605	2.7416e-6
$[0, 10]$	15	38.3138	3.6263	3.0260e-6
$[0, 12]$	16	100.8327	6.0296	2.9328e-6

Table 7. Numerical results of Example 3: $\varepsilon = 0.1,\ T = 2,\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N} = 7$.

	$MDDSM + DE(\tilde{\alpha} = 1.52)$				
$[\overline{a}, \overline{b}]$	iter(Newton)	iter(gmres)	time1(s)	time2(s)	error
$[0, 4]$	14	290	0.2821	13.7961	2.0411e-6
$[0, 6]$	14	323	0.4920	36.8127	2.0268e-6
$[0, 8]$	15	421	0.9341	107.7529	2.0266e-6
$[0, 10]$	15	443	1.5115	278.8113	2.0266e-6
$[0, 12]$	15	445	2.3945	719.9104	2.0266e-6

Table 8. Numerical results of Example 3: $\varepsilon = 0.1,\ T = 30,\ [\overline{a}, \overline{b}] = [0, 4],\ \overline{q} = \tilde{q} = \hat{q} = 2,\ \overline{N} = \tilde{N} = \hat{N} = 7$.

	$MDDSM + DE(\tilde{\alpha} = 1.52)$				
T	iter(Newton)	iter(gmres)	time1(s)	time2(s)	error
$[0, 2]$	14	290	0.2843	13.9283	2.0411e-6
$[2, 4]$	14	245	0.2936	12.6400	1.0533e-6
$[4, 6]$	14	258	0.2786	13.7926	1.0279e-6
$[6, 8]$	14	302	0.2829	14.1546	1.9625e-6
$[8, 10]$	14	261	0.3015	13.6505	1.0869e-6
$[10, 12]$	14	250	0.2782	13.4784	1.0365e-6
$[12, 14]$	14	298	0.2898	14.0763	1.8943e-6
$[14, 16]$	14	263	0.2911	13.4321	1.3047e-6
$[16, 18]$	14	240	0.2950	13.7441	9.9710e-7
$[18, 20]$	14	297	0.2817	14.5544	2.0532e-6
$[20, 22]$	14	273	0.2769	13.8349	1.7510e-6
$[22, 24]$	14	230	0.2809	13.1796	8.9217e-7
$[24, 26]$	14	297	0.2914	14.4809	2.0169e-6
$[26, 28]$	14	286	0.3102	13.2578	2.0168e-6
$[28, 30]$	14	225	0.3025	12.7572	8.2530e-7

From the numerical results in Tables 1–8, we can get the following observations:

1. From the left column in Table 1, we see that for a fixed interval $[0, T] \times \Omega$, as the degree of the Legendre polynomial (also the number of collocation points) increases, the resulting error decreases accordingly. Moreover, the results in the right column of Table 1 reflect that the common ILU preconditioner is ineffective for MDDSM.

2. In all the tested intervals in time and space, the expected accuracy $1e-6$ can be obtained by only taking $\overline{N} = \tilde{N} = \hat{N} = 7$ or 8 in MDDSM. This means that MDDSM does well in large time-space interval $\Omega \times [0, T]$ with little cost.

3. The results in Tables 4, 5, and 8 reflect that MDDSM is stable in time level, namely, supposing we have obtained the values in $[T_0, T_1]$, $[T_1, T_2]$, $\cdots$, $[T_{k-2}, T_{k-1}]$, then, the initial value of the k_{th} level $[T_{k-1}, T_k]$ can be chosen from the value of T_{k-1} computed in $[T_{k-2}, T_{k-1}]$, and the error in $[T_{k-1}, T_k]$ will be of the same order of magnitude of $[T_{k-2}, T_{k-1}]$.

4. Tables 2–8 show that the DE preconditioner is efficient in convergence, and that it is stable, namely, it appears to be independent of the size of interval in time/space.

5. From Tables 7 and 8, we can see that the classical Newton iterative method is suitable for the nonlinear algebraic equations which are obtained by using MDDSM to solve the tested model.

6. In Tables 2 and 3, we find that "MDDSM+DE" is obviously superior to "CDDSM+ILU"; in fact, in the heat equation, ILU has failed in CDDSM.

7. In all these results, we find that, for a fixed N, the number of iterations of the GMRES method is basically independent of the time interval $[0, T]$ and space region Ω, as reflected in Tables 2–8. Consequently, for a given numerical experiment, we can choose the parameter $\tilde{\alpha}$ by testing it in a small region $[0, T] \times \Omega$.

5. Conclusions

In this paper, we presented a modified domain decomposition spectral collocation method, and explored a dimension expanding preconditioner for the obtained linear system. Numerical examples show that this MDDSM is efficient for model (1.1) with proper boundary values and initial values in a large time-space interval $\Omega \times [0, T]$, and that the classical Newton iterative method and DE preconditioner are suitable for the proposed MDDSM.

How to choose an optimal parameter $\tilde{\alpha}$ in (3.3) is still a concern of our future work. Also, we will focus on improving this kind of dimension expanding preconditioner for saddle problem (2.10).

Author contributions

Wei Hua Luo conceived the idea for this paper and derived the algebraic equations; Liang Yin conducted all the numerical experiments; and Jun Guo was responsible for editing the paper.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research is supported by the Scientific Research Fund of Hunan Provincial Science and Technology Department (2022JJ30416), the Scientific Research Fund of Hunan Provincial Education Department (22A0483). The work of Jun Guo is supported by the Sichuan National Applied Mathematics co-construction project (2022ZX004), CUIT (KYTD202243) and the Scientific Research Foundation (KYTZ202184).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. W. L. Wood, R. W. Lewis, A comparison of time marching schemes for the transient heat conduction equation, *Int. J. Numer. Methods Eng*, **9** (2010), 679–689. https://doi.org/10.1002/nme.1620090314

2. P. Tsatsoulis, H. Weber, Exponential loss of memory for the 2-dimensional Allen-Cahn equation with small noise, *Probab. Theory Relat. Fields* , **177**(2019), 257–322. https://doi.org/10.1007/s00440-019-00945-x

3. B. Y. Guo, T. J. Wang, Composite generalized Laguerre-Legendre spectral method with domain decomposition and its application to Fokker-Planck equation in an infinite channel, *Math. Comp.*, **78** (2009), 129–151. https://doi.org/10.1090/S0025-5718-08-02152-2

4. R. Codina, A discontinuity-capturing crosswind-dissipation for the finite element solution of the convection-diffusion equation, *Comput. Methods Appl. Mech. Eng.*, **110** (1993), 325–342. https://doi.org/10.1016/0045-7825(93)90213-H

5. N. Li, J. Steiner, S. Tang, Convergence and stability analysis of an explicit finite difference method for 2-dimensional reaction-diffusion equations, *The ANZIAM Journal* , **36** (1994), 234–241. https://doi.org/10.1017/S0334270000010377

6. F. J. Domínguez-Mota, S. M. Armenta, G. Tinoco-Guerrero, J. G. Tinoco-Ruiz, Finite difference schemes satisfying an optimality condition for the unsteady heat equation, *Math Comput Simulat*, **106** (2014), 76–83. https://doi.org/10.1016/j.matcom.2014.02.007

7. N. Riane, C. David, The finite difference method for the heat equation on Sierpinski simplices, *Int J Comput Math*, **96** (2019), 1477–1501. https://doi.org/10.1080/00207160.2018.1517209

8. X. He, K. Wang, Uniformly convergent novel finite difference methods for singularly perturbed reaction-diffusion equations, *Numer. Methods Partial Differ. Equ.*, **35** (2019), 2120–2148. https://doi.org/10.1002/num.22405

9. C. C. Ji, R. Du, Z. Z. Sun, Stability and convergence of difference schemes for multi-dimensional parabolic equations with variable coefficients and mixed derivatives, *Int J Comput Math*, **95** (2018), 255–277. https://doi.org/10.1080/00207160.2017.1381336

10. J. Kim, D. Jeong, S. D. Yang, Y. Choi, A finite difference method for a conservative Allen-Cahn equation on non-flat surfaces, *J. Comput. Phys.*, (2017), 170–181. https://doi.org/10.1016/j.jcp.2016.12.060

11. X. Feng, H. J. Wu, A posteriori error estimates and an adaptive finite element method for the Allen-Cahn equation and the mean curvature flow, *J. Sci. Comput.*, **24** (2005), 121–146. https://doi.org/10.1007/s10915-004-4610-1

12. C. M. Elliott, D. A. French, A nonconforming finite-element method for the two-dimensional Cahn-Hilliard equation, *SIAM J. Numer. Anal.*, **26** (1989), 884–903. https://doi.org/10.1137/0726049

13. A. Georgios, B. Li, Error estimates for fully discrete BDF finite element approximations of the Allen-Cahn equation, *IMA J. Numer. Anal.*, **42** (2020), 363–391. https://doi.org/10.1093/imanum/draa065

14. A. Al-Taweel, S. Hussain, X. Wang, A stabilizer free weak Galerkin finite element method for parabolic equation, *J. Comput. Appl. Math.*, **392** (2021), 113373. https://doi.org/10.1016/j.cam.2020.113373

15. J. Shen, T. Tang, Spectral and High-Order Methods with Applications, Science Press, Beijing, 2006.

16. Y. Gu, J. Shen, Accurate and efficient spectral methods for elliptic PDEs in complex domains, *J Sci Comput*, **83** (2020), 42. https://doi.org/10.1007/s10915-020-01226-9

17. E. Pindza, M. K. Owolabi, K. C. Patidar, Barycentric Jacobi spectral method for numerical solutions of the generalized Burgers-Huxley equation, *Int. J. Nonlinear Sci. Numer. Simul.*, **18** (2017), 67–81. https://doi.org/10.1515/ijnsns-2016-0032

18. W. H. Luo, T. Z. Huang, X. M. Gu, Y. Liu, Barycentric rational collocation methods for a class of nonlinear parabolic partial differential equations, *Appl Math Lett*, **68** (2017), 13–19. https://doi.org/10.1016/j.aml.2016.12.011

19. S. H.Lui, S. Nataj, Spectral collocation in space and time for linear PDEs, *J. Comput. Phys.*, **424** (2021), 109843. https://doi.org/10.1016/j.jcp.2020.109843

20. S. H.Lui, S. Nataj, Chebyshev spectral collocation in space and time for the heat equation, *Electron Tran Numer Ana*, **52** (2020), 295–319. http://doi.org/10.1553/etna_vol52s295

21. S. H.Lui, Legendre spectral collocation in space and time for PDEs, *Numer. Math.* , **136** (2017), 75–99. https://doi.org/10.1007/s00211-016-0834-x

22. E. Pindza, F. Youbi, E. Mare, M. Davison, Barycentric spectral domain decomposition methods for valuing a class of infinite activity Lévy models, *Discrete Cont Dyn-s*, **12** (2018), 625–643. http://hdl.handle.net/2263/70151

23. J. L. Vay, I. Haber, B. B. Godfrey, A domain decomposition method for pseudo-spectral electromagnetic simulations of plasmas, *J. Comput. Phys.*, **243** (2013), 260–268. https://doi.org/10.1016/j.jcp.2013.03.010

24. W. H. Luo, X. M.Gu, B. Carpentieri, A dimension expanded preconditioning technique for saddle point problems, *Bit Numer Math*, **62** (2022), 1983–2004. https://doi.org/10.1007/s10543-022-00938-8

25. W. H. Luo, B. Carpentieri, J. Guo, A dimension expanded preconditioning technique for block two-by-two linear equations, *Demonstr. Math.*, **56** (2023), 20230260. https://doi.org/10.1515/dema-2023-0260

26. Y. Saad, Iterative Methods for Sparse Linear Systems (2nd ed.), Society for Industrial and Applied Mathematics, Philadelphia, 2003.