ELECTRONIC RESEARCH ARCHIVE
Volume **29**, Number **5**, November **2021**

doi:10.3934/era.2021031
pp. **3141–3170**

A MULTIGRID BASED FINITE DIFFERENCE METHOD FOR SOLVING PARABOLIC INTERFACE PROBLEM

Hongsong Feng and Shan Zhao*

Department of Mathematics, University of Alabama
Tuscaloosa, AL 35487, USA

Abstract. In this paper, a new Cartesian grid finite difference method is introduced to solve two-dimensional parabolic interface problems with second order accuracy achieved in both temporal and spatial discretization. Corrected central difference and the Matched Interface and Boundary (MIB) method are adopted to restore second order spatial accuracy across the interface, while the standard Crank-Nicolson scheme is employed for the implicit time stepping. In the proposed augmented MIB (AMIB) method, an augmented system is formulated with auxiliary variables introduced so that the central difference discretization of the Laplacian could be disassociated with the interface corrections. A simple geometric multigrid method is constructed to efficiently invert the discrete Laplacian in the Schur complement solution of the augmented system. This leads a significant improvement in computational efficiency in comparing with the original MIB method. Being free of a stability constraint, the implicit AMIB method could be asymptotically faster than explicit schemes. Extensive numerical results are carried out to validate the accuracy, efficiency, and stability of the proposed AMIB method.

1. **Introduction.** The paper is focused on the following two dimensional (2D) parabolic interface problem

$$\frac{\partial u}{\partial t} = \nabla \cdot (\beta \nabla u) + f, \quad \text{in } \Omega \subset \mathbb{R}^2 \tag{1}$$

with Dirichlet boundary conditions imposed along the boundary $\partial\Omega$ of a given rectangular domain. The whole domain is separated by an embedded interface into two subdomains such that $\Gamma = \Omega^+ \cap \Omega^-$, and $\Omega = \Omega^+ \cup \Omega^-$. In our study, we specify Ω^+ as outer domain, and Ω^- as inner domain. The diffusion coefficient β is defined to be a piecewise constant in different subdomains. Across the interface Γ, the function values of u from the two subdomains are associated by the following jump conditions

$$[\![u]\!] := u^+ - u^- = \phi(x, y, t), \tag{2}$$

$$[\![\beta u_n]\!] := \beta^+ \nabla u^+ \cdot \vec{n} - \beta^- \nabla u^- \cdot \vec{n} = \psi(x, y, t), \tag{3}$$

where superscripts $+$ and $-$ on u and β signify the corresponding subdomains, $\vec{n}$ indicates the normal direction pointing from Ω^- to Ω^+, and the jump values ϕ and

2020 *Mathematics Subject Classification.* 65M06, 65M55, 65M12.

Key words and phrases. Parabolic interface problem, matched interface and boundary (MIB), multigrid method, Schur complement, gradient recovery.

The authors are supported by NSF grant DMS-1812930.

* Corresponding author: Shan Zhao.

ψ are allowed to be temporally and spatially dependent to account for the most general scenario.

The parabolic interface problem modeled by (1) - (3) plays important roles in multiple physical and engineering applications. The conductive heat transfer process over composite media is widely investigated through this model when the interface conditions are homogeneous, i.e., $[\![u]\!] = 0$, and $[\![\beta u_n]\!] = 0$. Due to the discontinuity of diffusion coefficient across the interface, the analytical solution of the parabolic interface problem is not readily available even in the case of homogeneous interface jump conditions. Hence numerical approaches are in need for such problems. However, standard numerical procedures cannot deliver accurate solutions due to the assumption that solution is smooth over the whole domain.

Various specially designed numerical schemes with the jump conditions being incorporated into discretization have been introduced in the literature for solving parabolic interface problems. These include finite element methods and finite volume methods based on body-fitted interface treatments [3, 8, 32, 33, 34], finite difference methods based on Cartesian grids [2, 5, 6, 18, 19, 22, 17, 30, 40, 24, 36, 20], and immersed finite element methods based on Cartesian meshes [25, 39]. For instance, the immersed interface method (IIM) is one of the most successful finite difference methods in solving parabolic interface problems [5, 6, 18, 19, 20]. By avoiding mesh generation, generalized Taylor expansions are adopted to locally modify the finite difference weights so that the accuracy loss across the interface can be recovered.

Time integration is also crucial to the numerical solution of parabolic interface problems. Explicit time stepping would require a severe stability restriction with the time step size $\Delta t < Ch^2/\beta_{max}$, where h is the spatial grid size and $\beta_{max} = \max_{x\in\Omega}\beta(x)$. The implicit time integration is thus preferred in many applications involving long-time integration for the steady state solution. However, in implicit time stepping, a linear system, which involves all spatial degree of freedom N and is obtained from discretizing a Helmholtz type equation, has to be solved in each time step. With a generic iterative solver, the computational complexity could be $O(N^2)$ in each time step. This leads to tremendous amount of computing resources, especially when the parabolic interface problem is generalized to three dimensions. This calls for the development of fast algorithms for solving such a linear system with the complexity being $O(N)$ or $O(N \log N)$. In scientific computing, some possible candidates with such efficiency could be the fast Fourier transform (FFT), multigrid, or alternating directional implicit (ADI) methods. Nevertheless, the FFT is not readily applicable to such parabolic interface problem due to the non-constant coefficient of the reaction term in the Helmholtz equation. Therefore, ADI and multigrid are two popular techniques for designing fast algorithms for parabolic partial differential equations (PDEs).

The core philosophy of ADI methods [12, 11, 29] is to reduce a multidimensional system from discretizing parabolic equation to a sequence of independent one-dimensional (1D) sub-systems of tridiagonal structure, which can be solved efficiently with the Thomas algorithm [31]. Compared with generic iterative solvers, ADI methods can be regarded as an exact solver due to the fact that computation will be completed within a fixed number of steps. Recently, ADI schemes have been developed to cope with parabolic interface problems with desired ADI efficiency maintained. Li and Mayo [22] introduced the first ADI algorithm with help of the IIM to correct local differences near interface when facing a special interface jump with $[\![\beta]\!] = 0$ across the interface. The further studies of such IIM scheme can be

found in [26, 27]. Zhao [40] proposed the first ADI algorithm to tackle the general jump conditions with $[\![\beta]\!] \neq 0$, by adopting matched interface and boundary (MIB) method [41, 42] for handling interfaces. A tensor product approach is applied to decompose the jump conditions into Cartesian and tangential directions. The Cartesian jump quantities can be used to correct 1D finite difference discretization, while the tangential jump is approximated by using function values in the previous time step. A modified Thomas algorithm is introduced to accomplish the classic computational complexity $O(N)$ of the ADI scheme. The same MIB-based philosophy is further generalized to three dimensional parabolic interface problem with a non-orthogonal local coordinate for a simpler implementation [36]. These MIB based ADI schemes [40, 36] achieve second order accuracy in space, but attain only a first order accuracy in time, due to tangential approximations. More recently, a multiscale ADI algorithm [20] has been proposed with aid of the IIM for general interface conditions. Two types of auxiliary variables on the interface are introduced for the augmented system in order to incorporate the ADI framework. Second order accuracy is fulfilled for both spatial and temporal discretization with a time step size constraint $\Delta t \leq Ch$. Most recently, the ghost-fluid method has also been combined with the ADI to solve parabolic interface problem with symmetric and tridiagonal finite difference matrix obtained [23]. The metric of the algorithm lies in its superior stability despite first order accuracy in space.

The multigrid method is another popular fast solver used in solving PDEs with discontinuous coefficients. Based on structured or unstructured meshes, approaches for constructing interpolation, smoothing operators and coarse gird points selection would require special designs [7]. Algebraic multigrid (AMG) [28] selects coarse grid points in a pure algebraic sense to define interpolation without interface geometry and dimensionality needed in the construction. Yet it exploits the discontinuous coefficients and geometry in the matrix-dependent interpolation. A great amount of storage is usually needed for its coarse grid operator matrices generated by Galerkin process. Recently, the AMG scheme has been successfully applied in the immersed finite element method for solving both stationary and moving interface problems [13]. Geometric multigrid method is the other class of multigrid methods, which includes interface geometry in the process of discretization to the PDE with related boundary and interface jump conditions. In [2], Adams and Li proposed a 2D IIM multigrid method, which preserves maximum principle. Black box multigrid interpolation is adopted away from the interface, while interpolation weights are derived with help of Taylor expansion for the grid points adjacent to the interface. Later on, an improved IIM based multigrid method [1] has been developed with interpolation and restriction operators being modified such that coarse-grid matrices are M-matrices. The number of V-cycles does not change with mesh refinement, and increases slightly with the ratio of discontinuous coefficients growing. In [35], a multigrid method is proposed, which can maintain a uniform convergence with respect to mesh size and the high interface jump ratio. It can be extended to three dimensions, and can solve multiple interface problem efficiently. Besides, it can avoid the issue of large storage of coarse-grid matrix. Extending the multigrid approach for solving continuous coefficient boundary value problem in [9], Coco further designed a multigrid method for solving discontinuous coefficient interface problem [10], which can even be further generalized to matrix coefficient problems. Its multigrid efficiency is guaranteed with the convergence factor not depending on the coefficient ratio. In spite of the great success achieved by the aforementioned

multigrid methods, a new multigrid method has to be designed again when a new interface algorithm is under consideration. Because each interface algorithm has a different procedure to capture boundary or interface conditions, sophisticated treatments are demanded to define interpolation with knowledge of geometry from the interface and to define restriction and coarse grid operators.

In this paper, we propose a new geometric multigrid method based on the augmented matched interface and boundary (AMIB) method for solving parabolic interface problems. The AMIB method was originally designed to solve elliptic interface problems [14, 15, 16]. With a FFT acceleration, the AMIB method can deliver fourth order accuracy in treating both interfaces [16] and boundaries [15]. Nevertheless, the AMIB method has never been coupled with a multigrid method before. In this study, due to the augmented approach adopted, the finite difference interface treatment and the multigrid solution procedure can be decoupled in the proposed AMIB method. This considerably simplifies the interpolation and restriction process for the coarse grid operators in the multigrid method. Hence standard multigrid components can be applied in such augmented framework with the advantages including that storage for coarse grid matrix is avoided and the implementation is fairly simple. Second order corrected central differences are applied for spatial accuracy to compensate accuracy loss across the interface, and the Crank-Nicolson scheme is used for temporal discretization. This leads to second order accuracy in both space and time, while the multigrid approach considerably accelerates the computational speed.

The rest of the paper is organized as follows. Section 2 focuses on discussion of the several key aspects of the proposed efficient algorithm. In section 3, numerical experiments are implemented to justify the effectiveness of our method in solving parabolic interface problems. Comparisons to other methods are carried out to show the efficiency of our method. A summary and acknowledgment are included in the end of the paper.

2. Theory and algorithm.

In this work, we focus on the piecewise constant coefficient parabolic interface problem. We first rewrite the original problem (1) into the following form

$$\frac{1}{\beta}\frac{\partial u}{\partial t} = \Delta u + \frac{f}{\beta}, \qquad \text{in } \Omega^+ \cup \Omega^- \setminus \Gamma. \tag{4}$$

Subject to the same initial, boundary, and interface conditions, the solution to (4) is equivalent to that of (1). We partition the given rectangular domain $\Omega = [a,b] \times [c,d]$ into to n_x and n_y equally spaced intervals in $x-$ and $y-$ directions respectively. The mesh size in each direction can then be defined as $h_x = (b-a)/n_x$ and $h_y = (d-c)/n_y$. For simplicity, we assume that $h_x = h_y = h$. The grid nodes are hence defined as

$$x_i = a + ih,\ y_j = c + jh,\ i = 0, \cdots, n_x,\ j = 0, \cdots, n_y.$$

Assume that the interface Γ is represented by a level set function

$$\Gamma = \{(x,y), \varphi(x,y) = 0\},$$

with $\varphi(x,y) > 0$ in Ω^+ and $\varphi(x,y) < 0$ in Ω^-. Two functions are defined at each grid point

$$\varphi_{ij}^{min} = \min\{\varphi_{i-1,j}, \varphi_{i,j}, \varphi_{i+1,j}, \varphi_{i,j-1}, \varphi_{i,j+1}\},$$
$$\varphi_{ij}^{max} = \max\{\varphi_{i-1,j}, \varphi_{i,j}, \varphi_{i+1,j}, \varphi_{i,j-1}, \varphi_{i,j+1}\}.$$

The subscript (i, j) indicates the function value of φ at (x_i, y_j). It is similarly defined for other variables. If $\varphi_{i,j}^{min}\varphi_{i,j}^{max} > 0$, then the grid point (x_i, y_j) is called a regular point, otherwise an irregular point.

Moreover, a uniform time increment $\triangle t$ is adopted for temporal discretization. Let $t_n = n \cdot \triangle t$ represent the nth time step in the temporal discretization. For ease of exposition, we use notation $u_{i,j}^n$ to denote the function value on grid (x_i, y_j) at time t_n.

2.1. **Temporal discretization.** For temporal discretization, the Crank-Nicolson scheme is adopted

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\beta_{i,j}\triangle t} = \frac{1}{2}\Delta u_{i,j}^n + \frac{1}{2}\Delta u_{i,j}^{n+1} + \frac{1}{2\beta_{i,j}}(f_{i,j}^n + f_{i,j}^{n+1}), \tag{5}$$

which has the equivalent form below after reorganizing the terms in (5)

$$\frac{2}{\beta_{i,j}\triangle t}u_{i,j}^{n+1} - \Delta u_{i,j}^{n+1} = \frac{2}{\beta_{i,j}\triangle t}u_{i,j}^n + \Delta u_{i,j}^n + \frac{1}{\beta_{i,j}}(f_{i,j}^n + f_{i,j}^{n+1}). \tag{6}$$

The above Crank-Nicolson scheme admits a second order temporal truncation error. Note that the Laplacian terms at time step $t = t^{n+1}$ and $t = t^n$ need spatial discretization such that the time stepping process (6) can proceed for the discrete solutions at certain time instant. The spatial discretization details are fully demonstrated in following sections.

2.2. **Spatial discretization.** Ignoring the time dependence, we concern ourselves with second order spatial discretization to the Laplacian operator

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

We may focus the discussion on $\frac{\partial^2 u}{\partial x^2}$ due to the tensor-product decomposition to be employed for discretizing the Laplacian operator. The standard second order central difference can be utilized to approximate the partial derivative u_{xx} with a truncation error $O(h^2)$:

$$u_{xx}(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - \frac{1}{12}h^2 u_{xxxx}(x_i) + O(h^4). \tag{7}$$

The above formulation (7) is applicable to the case of regular points, while approximation in irregular points needs different treatment due to the fact that function loses regularity across the interface. Corrected differences are employed below [37]:

Jump-corrected difference. Suppose $u \in C^4[x_i - h, \alpha) \cap C^4(\alpha, x_{i+1} + h]$ and $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$, where derivatives extend continuously up to α. The

following approximations hold to $O(h^2)$ when $K = 3$:

$$u_{xx}(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - \frac{1}{h^2}\sum_{k=0}^{K}\frac{(h^+)^k}{k!}[u^{(k)}], \tag{8}$$

$$u_{xx}(x_{i+1}) \approx \frac{u(x_{i+2}) - 2u(x_{i+1}) + u(x_i)}{h^2} + \frac{1}{h^2}\sum_{k=0}^{K}\frac{(h^-)^k}{k!}[u^{(k)}], \tag{9}$$

where $h^- = x_i - \alpha$, $h^+ = x_{i+1} - \alpha$ with $x_i \le \alpha < x_{i+1}$, and $[u^{(k)}]_\alpha = \lim_{x\to\alpha^+} u^{(k)}(x) - \lim_{x\to\alpha^-} u^{(k)}(x)$.

The global regularity for u will be assumed to be piecewise C^4 continuous, as required locally for jump-corrected differences. Note that the leading error term in (7) involves the fourth order derivative u_{xxxx}. So C^4 continuity is needed in each subdomain for maintaining the second order accuracy everywhere, except near the interface. In corrected differences, the Cartesian derivative jumps of up to order $K = 2$ is sufficient to guarantee a global second order accuracy despite being first order locally [14].

The corrected differences in Eqns.(8) and (9) will reduce to Eq. (7) with the introduced jump terms vanishing if no interface is met. Such jump quantities are compensating for the function discontinuity across the interface. Since the derivative jumps in the correction terms are not analytically available, numerical approximations are needed for the corrected differences to be applied at irregular points. In following subsection, a systematical procedure is discussed for reconstruction of the derivative jumps with aid of fictitious values generated by the matched interface and boundary (MIB) method [42].

2.3. **Reconstructing the Cartesian derivative jumps.** The derivative jumps in the above corrected differences are defined as

$$[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} = \lim_{x\to\alpha^+}\frac{\partial^k u}{\partial x^k} - \lim_{x\to\alpha^-}\frac{\partial^k u}{\partial x^k}, \tag{10}$$

where $k = 0, 1, 2$ indicates the order of derivative. We will construct an effective numerical approximation to such derivative jumps except for the zeroth order one, which is known in the interface condition (2). It is natural to approximate the one-side derivative jumps in Eq.(10) using polynomials from each side. Since the derivative order in Eq.(10) is up to two, a polynomial of degree two on each side of the interface is sufficient to guarantee the local first order accuracy in the corrected difference with each order derivative obtained via differentiation on the polynomial.

For the sake of accuracy and stability, it is better to approximate each one-sided derivative limit with central or pseudo-central difference. As shown in Fig.1, if one layer of fictitious values is available on each side of the interface $x = \alpha$, we can combine those with two real function values on the other side of the interface to yield the desired two Lagrange polynomials. The derivative jumps can be approximated as below:

$$[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} \approx (w^k_{i,j}\hat{u}_{i,j} + \sum_{l=1}^{2} w^k_{i+l,j}u_{i+l,j}) - (\sum_{l=1}^{2} w^k_{i-2+l,j}u_{i-2+l,j} + w^k_{i+1,j}\hat{u}_{i+1,j}), \tag{11}$$

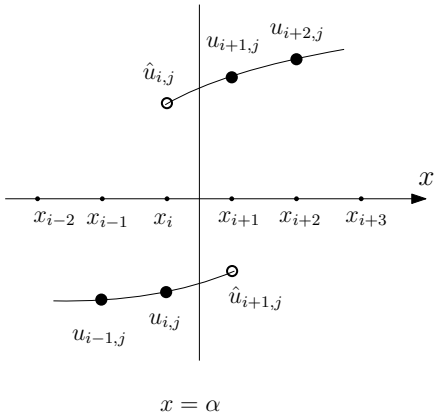


Figure 1. Black dots denote real function values, while empty circles indicate fictitious values. They are used in the polynomials for approximating derivative jump across the interface at $x = \alpha$.

where the linear combinations in the two parenthesis denote the derivatives of the two constructed Lagrange polynomials for approximating one-sided derivative limits at $x = \alpha$.

It is possible that only one real function value is available on one side of the interface due to sharp topology change of the interface. To deal with such a corner case [14], a fictitious value is employed at where the needed real value is missing. Just consider the derivative jumps at $x = \alpha_2$ in the two scenarios shown in Fig.2. The fictitious value at $x = x_{i-1}$ is used for the case in the right chart in comparison with the case in the left chart where the real value at $x = x_{i-1}$ is adopted.

Likewise, the discretization for the $y-$direction derivative jump can be formulated when the interface intersects with the x grid line. To apply the approximation (11), we need to construct fictitious values at all irregular points. In the following subsection, the MIB method is employed to generate the needed fictitious values.

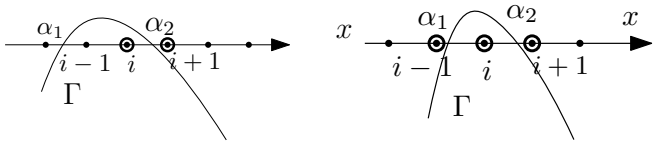


Figure 2. Derivative jump approximation in two scenarios. Empty circles denote the fictitious values at the grid points, while black dots indicate real function values.

2.4. **Fictitious values formulation.** The jump conditions (2) and (3) are given on the interface Γ. One more interface condition can be analytically derived by differentiating Eq. (2) along the tangential direction τ of the interface as below,

$$[\![u_\tau]\!] = u_\tau^+ - u_\tau^- = \rho(x, y). \tag{12}$$

Consider a point (x^*, y^*) on the interface. Define θ as the angle between positive $x-$direction and normal direction. Then the tangential and normal direction could be expressed as $\vec{\tau} = (-\sin\theta, \cos\theta)$ and $\vec{n} = (\cos\theta, \sin\theta)$, respectively. See Fig. 3 in

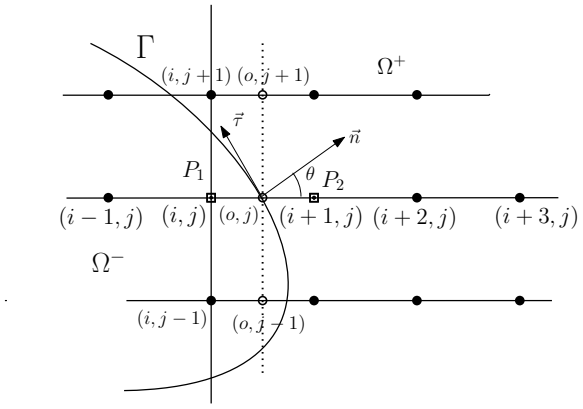


Figure 3. Consider the case where the interface intersects $y = y_j$ at (x_o, y_j). At the irregular points $P_1(i,j)$ and $P_2(i+1,j)$, fictitious values (in box) can be constructed. Here θ is the angle between positive $x-$direction and the normal vector $\vec{n}$.

case that (x^*, y^*) is located on a y grid line. At (x^*, y^*), the three known interface conditions can be reinterpreted in the Cartesian form [42],

$$[\![u]\!] = u^+ - u^- = \phi(x^*, y^*), \tag{13}$$

$$[\![u_\tau]\!] = (-u_x^+ \sin\theta + u_y^+ \cos\theta) - (-u_x^- \sin\theta + u_y^- \cos\theta) = \rho(x^*, y^*), \tag{14}$$

$$[\![\beta u_n]\!] = \beta^+(u_x^+ \cos\theta + u_y^+ \sin\theta) - \beta^-(u_x^- \cos\theta + u_y^- \sin\theta) = \psi(x^*, y^*). \tag{15}$$

In this way, jump conditions from tangential or normal direction are transformed into $x-$or $y-$direction involving four partial derivatives u_x^+, u_x^-, u_y^+, and u_y^-. The following formulations (16) or (17) can be obtained by eliminating u_y^- or u_x^- from the four quantities.

If u_y^- is eliminated, the following equations can be derived

$$[\![u]\!] = u^+ - u^-, \quad [\![\beta u_n]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ u_x^+ - C_x^- u_x^- + C_y^+ u_y^+, \tag{16}$$

where $C_x^+ = \beta^+ \cos\theta + \beta^- \tan\theta \sin\theta$, $C_x^- = \beta^- \cos\theta + \beta^- \tan\theta \sin\theta$ and $C_y^+ = \beta^+ \sin\theta - \beta^- \sin\theta$.

If u_x^- is removed, we can obtain that

$$[\![u]\!] = u^+ - u^-, \quad [\![\beta u_n]\!] + \beta^- \cot\theta [\![u_\tau]\!] = D_y^+ u_y^+ - D_y^- u_y^- + D_x^+ u_x^+, \tag{17}$$

where $D_x^+ = (\beta^+ - \beta^-)\cos\theta, D_y^+ = \beta^- \cos\theta \cot\theta + \beta^+ \sin\theta$ and $D_y^- = \beta^-(\cos\theta \cot\theta + \sin\theta)$.

Eqns. (16) and (17) allow us to reduce the 2D jump conditions to 1D ones along Cartesian direction provided that u_y^+ and u_x^+ can be determined, respectively. Suppose the interface intersects with $y = y_j$ at (x_o, y_j) as shown in Fig. 3. To generate a pair of fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ at points P_1 and P_2 on each side of the interface, discretization is carried out on Eqn.(16) as below,

$$[\![u]\!] = W_0^+ U^+ - W_0^- U^- \tag{18}$$

$$[\![\beta u_n]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ W_1^+ U^+ - C_x^- W_1^- U^- + C_y^+ P^+ U_o, \tag{19}$$

with the following vector representations

$$\begin{cases} U^+ = (\hat{u}_{i,j}, u_{i+1,j}, u_{i+2,j})^T \\ U^- = (u_{i-1,j}, u_{i,j}, \hat{u}_{i+1,j})^T \end{cases},$$

$$\begin{cases} W_0^+ = (w_{0,i}^+, w_{0,i+1}^+, w_{0,i+2}^+) \\ W_0^- = (w_{0,i-1}^-, w_{0,i}^-, w_{0,i+1}^-) \end{cases},$$

$$\begin{cases} W_1^+ = (w_{1,i+2}^+, w_{1,i+3}^+, w_{1,i+4}^+) \\ W_1^- = (w_{1,i-1}^-, w_{1,i}^-, w_{1,i+1}^-) \end{cases},$$

$$\begin{cases} U_o = (u_{o,j-1}, u_{o,j}, u_{o,j+1})^T \\ P^+ = (p_{1,j-1}^+, p_{1,j}^+, p_{1,j+1}^+) \end{cases}.$$

Here U^+, U^- indicate the involved function and fictitious values and the four vectors W_0^+,W_0^-,W_1^+,W_1^- represent the Lagrange interpolation weights when discretizing (16) along $x-$direction. Note that fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ in U^+ and U^- participate in the process of approximation to the $x-$direction interpolation. Analogously, U_o and P^+ are the needed auxiliary function values and corresponding weights to approximate u_y^+. The superscripts $-$ and $+$ in the above vector notations or the elements in each vector signify the Ω^- and Ω^+ domain. Beside, the subscript 0 or 1 indicates the zeroth or first order derivative. The index i or its variance tells the information located at $x = x_i$. Note that three auxiliary function values in U_o are at points $(x_o, y_{j-1}), (x_o, y_j), (x_o, y_{j+1})$, see Fig 3. Additionally, each of these three auxiliary values are interpolated or extrapolated by three function values in Ω^+. For example, in Fig. 3, $u_{o,j+1}$ is interpolated by function values $u_{i,j+1}, u_{i+1,j+1}$ and $u_{i+2,j+1}$. Similarly, $u_{o,j}, u_{o,j-1}$ will be extrapolated by three function values from their right side in the Ω^+ domain.

With the function values in U_o appropriately approximated, (18) and (19) produce fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ at points (x_i, y_j) and (x_{i+1}, y_j) represented as a linear combination of surrounding points near the interface and three jump conditions at interface point (x^*, y^*). For instance, the representation of $\hat{u}_{i,j}$ can take a general form as

$$\begin{aligned} \hat{u}_{i,j} &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 [\![u]\!] + W_1 [\![u_\tau]\!] + W_2 [\![\beta u_n]\!] \\ &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 \phi(x^*, y^*) + W_1 \rho(x^*, y^*) + W_2 \psi(x^*, y^*), \end{aligned} \tag{20}$$

where $\mathbb{S}_{i,j}$ represents a set of surrounding nodes involved in (18) and (19).

In this manner, the needed two layers of fictitious values can be generated along $x-$direction by enforcing the jump conditions (16) or along $y-$direction by enforcing (17). Interested readers are referred to [42] for more details.

Remark 1. The above MIB formulation utilizes extrapolations in Ω^+ only, i.e., u_y^+ and u_x^+ in (16) and (17) are perpendicularly approximated by using auxiliary values, and each auxiliary value will be extrapolated by using function values in Ω^+. Recall that Eqs.(16) and (17) are derived after u_y^- or u_x^- being eliminated from Eqs.(13) - (15). On the other hand, if we remove u_y^+ or u_x^+ from Eqns (13) - (15), we can have another set of formulations involving u_y^- or u_x^-. Numerically, u_y^- or u_x^- needs to be extrapolated by only using function values in Ω^-. We call the first approach as MIB-PLUS and second one as MIB-MINUS in terms of function values used from Ω^+ or Ω^- in the extrapolation process.

Remark 2. It is found in our experiments that the accuracy of derivative jump reconstruction in Eq. (11) could be affected by the fictitious value generated by MIB-PLUS or MIB-MINUS depending on β^+ and β^-. When $\beta^+ > \beta^-$, the accuracy could be better if MIB-PLUS is used. Similarly, when $\beta^+ < \beta^-$, the accuracy could be better if MIB-MINUS is adopted. Nevertheless, for efficiency concern, we will use fictitious values obtained by MIB-PLUS in the case when $\beta^- > \beta^+$, while MIB-MINUS for the case with $\beta^- < \beta^+$ [16]. This strategy can reduce the iteration number in the overall algorithm.

2.5. **Formulation of augmented system.** Great details have been given to demonstrate how corrected differences are used to approximate the Laplacian operator. The right side of the Crank-Nicolson scheme (6) can be easily determined once the information of solution u^n, the interface conditions ϕ, ρ, ψ at current step t^n, and the source term f at t^n and t^{n+1} are gathered. Especially, corrected differences with the jump approximation method (11) are utilized to deal with approximation at irregular points on both sides of the time-stepping scheme (6). However, it will be inefficient to apply iterative algorithms to solve the linear system underlying (6) at t^{n+1}. It is noted that the standard second order central difference matrix structure with some additional diagonal entries is obtained if the corrected difference is neglected. Due to such special matrix structure, a multigrid-based fast solver is established to fulfill the fast solution of the parabolic interface problem. Prior to the introduction of fast multigrid solver, we need to introduce the auxiliary variables in order to take advantage of the special matrix structure.

2.5.1. *Auxiliary variables.* The right hand side of the time-stepping scheme (6) can be explicitly calculated with the known information on boundary or interface conditions, while the left hand side of (6) needs efficient algorithm design. For this purpose, we introduce auxiliary variables to form an augmented system.

With the general formulation of fictitious values in (20), the approximation (11) can be well defined. At each intersection point between the interface Γ and $x-$or $y-$grid line, auxiliary variables $[u_x], [u_{xx}]$ or $[u_y], [u_{yy}]$ are introduced. Then approximation (11) can be reformulated as

$$C_k U + [\frac{\partial^k u}{\partial x^k}] = \Phi_k, \tag{21}$$

where C_k is the corresponding weight coefficient, U signifies the needed function values in (11), and Φ_k is from the known quantities of the fictitious values in (11). The subscript k indicates the jump of kth derivative at time t^{n+1}. Then the derivative jumps at all intersecting points can contribute a general matrix-vector form

$$CU + IQ = \Phi, \tag{22}$$

where C stands for the weight coefficients as in (21), Q denotes all the auxiliary variables.

2.5.2. *Augmented system.* Let $U_{i,j}$ indicate the discrete solution while $u(x_i, y_j)$ is continuous solution at (x_i, y_j). Based on the corrected difference analysis for spatial discretization, the scheme (6) is expanded at all interior nodes as

$$D_h U_{i,j}^{n+1} - L_h U_{i,j}^{n+1} - C_{i,j}^{n+1} = D_h U_{i,j}^{n} + L_h U_{i,j}^{n} + C_{i,j}^{n} + \frac{1}{\beta_{i,j}}(f_{i,j}^{n+1} + f_{i,j}^{n}),$$
$$\text{for } \ 1 \le i \le n_x - 1, \ \ 1 \le j \le n_y - 1, \tag{23}$$

where $C_{i,j}$ is the correction term, and $L_h U_{i,j}$ is the standard five points central difference scheme

$$L_h U_{i,j} := \frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2}, \tag{24}$$

$$D_h U_{i,j} := \frac{2}{\beta_{i,j} \triangle t} U_{i,j}. \tag{25}$$

Note that the subscript (i, j) indicates the gird node (x_i, y_j), and the superscript n and $n+1$ imply the time t^n and t^{n+1}, respectively. The degree of freedom of this system is $N = (n_x + 1)(n_y + 1)$. The correction term $C_{i,j}$ only exists at irregular points but vanishes at regular points. The discretization (23) gives rise to the matrix-vector form

$$DU^{n+1} + \hat{A}U^{n+1} + BQ^{n+1} = DU^n - \hat{A}U^n - BQ^n + \hat{F}^n \tag{26}$$

where D is a matrix with non-zero entries only along the diagonal, $\hat{A}$ is a N-by-N non-symmetric and diagonally dominant matrix, the unknown U is for solution at all grid nodes and Q is for all the needed auxiliary variables. The diagonal entries of matrix D are non-constant due to the different values of β in different subdomain. As the right hand side of (26) can be determined in each time step, we can denote the right hand side of the above system as F. The above system can be further simplified as

$$AU + BQ = F, \tag{27}$$

after we combine the coefficient matrix D and $\hat{A}$ into A, i.e., $A = D + \hat{A}$, and ignore the superscript $n+1$. Both U and Q are to be solved at time t^{n+1}. Supposed Q is already determined, then the time stepping process continues by updating right side U^n with obtained solution U^{n+1} in (26). The Dirichlet boundary condition is imposed in the vector F with some rows of matrix A composed by vector $(0, 0, \cdots, 0, 1, 0, \cdots, 0)$, i.e., only the diagonal element is one while other entries are zero. The total number of auxiliary variables Q, denoted as M here, is double of that of all intersection points, while the latter is usually proportional to n_x or n_y. In other words, M is usually one-dimensional smaller than $N = (n_x + 1)(n_y + 1)$. The matrix B with dimension N-by-M contains coefficients from correction terms. In particular, matrix B is a sparse matrix, since the correction terms only have impact on the irregular points which account for a small portion in the whole computation grid points.

We need to determine Q first to solve for U in (27). It is necessary to combine (27) and (22) to form augmented equation system

$$KW = R, \tag{28}$$

where

$$K = \begin{pmatrix} A & B \\ C & I \end{pmatrix}, W = \begin{pmatrix} U \\ Q \end{pmatrix}, \quad \text{and} \quad R = \begin{pmatrix} F \\ \Phi \end{pmatrix}.$$

Note that the dimensions of matrix C and I match those of A and B. Here C is a M-by-N sparse matrix, and I is a M-by-M identity matrix.

The matrix structure of A allows for fast inversion with aid of multigrid method, which will be discussed later. Bearing this in mind, we will formulate two efficient solvers for the parabolic interface problem by using the following Schur complement strategy.

2.5.3. *Schur complement.* In each time step, we aim at solving U from

$$AU = F - BQ, \tag{29}$$

after we determine Q, and move all the known quantities to right hand side. Then the solution U could be quickly obtained by applying the multigrid solver on the above equation thanks to the matrix structure of A. Such step is preceded by the determination of Q from the Schur complement

$$(I - CA^{-1}B)Q = \Phi - CA^{-1}F, \tag{30}$$

which is obtained by eliminating U from the augmented system (28).

Note that the linear system (30) for Q has a much smaller degree of freedom than U, i.e. M. We propose two approaches to solve Q from Eq. (30) in terms of iterative solver or the LU decomposition on the coefficient matrix of the left hand side in Eq.(30). The following steps are used to illustrate the procedure. One is by iterative algorithm of GMRES method(GMRES approach), while the other is by LU decomposition(LU approach).

- *GMRES approach.*
  1. As a linear system, the right hand side RHS=$\Phi - CA^{-1}F$ is determined by applying multigrid solver on $A^{-1}F$ and some arithmetic operations.
  2. For the left hand side(LHS), we may consider an iteration approach by the GMRES method. In this approach, the matrix vector product of $(I - CA^{-1}B)Q$ is fulfilled in a couple of steps. Due to its equivalence to $IQ - CA^{-1}BQ$, a multigird solver is carried out on the product of BQ, i.e., $A^{-1}(BQ)$. Then it is followed by some more arithmetic operations on $IQ - CA^{-1}(BQ)$.
  3. For the first time step $t = 0$, an initial guess $Q = (0, 0, \cdots, 0)^T$ is used to start the GMRES iteration. For the rest time steps, the initial guess of Q is taken as the computed Q from the previous time step. Either the maximal iteration number 5000 or error tolerance equal to 10^{-10} terminates the GMRES iteration. The termination criteria are flexible according to actual computational needs.
- *LU approach.*
  1. The first step is the same as that in the above GMRES approach.
  2. For the left hand side(LHS), we will explicitly construct the matrix $I - CA^{-1}B$, where the inverse of A, i.e. A^{-1}, should not be formed, but is applied on the columns of matrix B. Then the explicit matrix is decomposed into the matrix product of X and Y from a LU decomposition solver. Such matrix construction and decomposition need to conducted only once before the time stepping.
  3. The solution Q in each time step can be simply obtained via the LU forward and backward substitutions.

Remark 3. The efficiency of GMRES approach depends on the iteration number for the GMRES iteration process. Note that each iteration involves one multigrid inversion with a complexity $O(N) = C_2N$ for some constant C_2. Fortunately, the iteration is for solving (30), whose dimension $M \ll N$. Generally, for any iteration solver, the better conditioned the iteration matrix is, the smaller the iteration number is. The selection of MIB-PLUS or MIB-MINUS mentioned in Remark 2 provides a good conditioning strategy. This makes the iteration number increase moderately as the mesh is refined.

Remark 4. The strategy of LU approach is different from GMRES approach in the sense that the Schur complement matrix is formed with multigrid algorithm applied for each column of matrix B. The more grid lines intersect the interface, the more columns matrix B has. The computing cost is mainly contributed by the multigrid inversion on all the columns, before the time stepping. In the time-stepping process, only the forward and backward substitution of LU are applied to solve for auxiliary variable, and one multigrid inversion is deployed on (29) for solution of u. Moreover, the LU approach is not impacted by the conditioner number of the linear system as it does for GMRES approach, since the LU decomposition is a direct rather than an iterative solver.

Remark 5. The complexities of the two approaches are in the same magnitude. As it is second order accurate for both temporal and spatial discretization, it is reasonable to set time increment Δt proportional to spatial partition size h to ensure the overall second order convergence. Suppose the total degree of freedom $N = n^2$, where n is the grid number in each direction. The intersection number M is hence equal to $C_0 n$ for some constant C_0, and the temporal evolution number $N_t = C_3 n$ for another constant C_3. We may roughly have an estimate of the complexities of the two approaches. The complexity of the LU approach is

$$\begin{aligned}
&O(N) \times M + O(M^2) \times N_t + \frac{2}{3}M^2 \\
=&C_2 N M + C_1 M^2 N_t + \frac{2}{3}M^2 \\
=&C_1 C_0^2 n^2 C_3 n + C_2 n^2 C_0 n + \frac{2}{3}C_0^3 n^2 \\
=&C_1 C_0^2 C_3 n^3 + C_2 C_0 n^3 + \frac{2}{3}C_0^3 n^2 \\
=&C_0(C_0 C_1 C_3 + C_2 + \frac{2}{3}C_0^2)n^3,
\end{aligned}$$

where the first and second term in the first equality indicates the total sum of matrix generation cost arising from multigrid solver and the cost of multigrid for inversion deployed on (29) and right side of (30), and the LU forward and backward substitution cost in all time evolution steps. Besides, the last term in the first equality denotes the LU decomposition for the generated matrix in the first time stepping.

The computational complexity of the GMRES approach is

$$O(N) \times C_4 \times N_t = C_2 n^2 C_4 C_3 n = C_2 C_3 C_4 n^3 = C_3(C_2 C_4)n^3,$$

where C_4 indicates the sum of total times multigrid solver is call in the GMRES subroutine, and multigrid for inversion deployed on (29) and right side of (30).

The computational complexities of the two approaches are all of $O(n^3)$. Note that the number C_4 is a constant under the assumption that the iteration number of GMRES solver does not depend on mesh size. But in the practical implementation of our algorithm, the iteration number depends on the mesh size weakly. This can be seen in the numerical examples of the latter section. This makes AMIB-LU slightly more efficient than AMIB-GMRES when achieving second order accurate solution.

Remark 6. When an explicit finite difference method is employed to solve parabolic PDEs, a stability condition with $\Delta t < Ch^2/\beta_{max}$ has to be satisfied, where $\beta_{max} =$

$\max_{x\in\Omega}\beta(x)$. This means that one has to take $N_t = O(n^2)$ to ensure stability. It is reasonable to assume the complexity in each time step of this explicit scheme to be on the order of $O(n^2)$, since there are n^2 degree of freedom. Thus, the overall complexity of explicit finite difference solution of the 2D parabolic interface problems is about $O(n^4)$. This means that the proposed AMIB methods could be asymptotically faster than the explicit schemes.

2.5.4. *Multigrid algorithm.* In the preceding section, a multigrid approach has been assumed in the discussion of building efficient algorithm. In this section, details on this solver are provided.

The multigrid approach is an efficient algebraic algorithm for solving a linear system

$$AU = B \tag{31}$$

with known matrix A and vector B. Recall in our study, A has a special matrix structure: $A = D + \hat{A}$. Thus, Eq. (31) can be regarded as obtaining from a discretization to negative of Laplacian operator plus solution with variable coefficient as below:

$$-\Delta u + a(x,y)u = b(x,y), \tag{32}$$

where the domain and its partition are the same as that for the problem in this work, the discrete values of $a(x,y)$ on the given partition are the same as diagonal values of D, U is the discrete solution to u in (32), and function b has its discrete data equal to B at the interior grid nodes and for boundary conditions.

To be more clear, at the interior node,

$$\begin{aligned} a(x_i,y_j) =& D((j-1)n_x + i, (j-1)n_x + i),\\ b(x_i,y_j) =& B((j-1)n_x + i, (j-1)n_x + i), \end{aligned}$$

for $1 \le i \le n_x - 1$, and $1 \le j \le n_y - 1$, while at the grid node on boundary of Ω,

$$b(x_i,y_j) = B((j-1)n_x + i, (j-1)n_x + i), \tag{33}$$

for (i,j) such that $(x_i,y_j) \cap \partial\Omega \neq \emptyset$.

The discrete problem can then be reformulated as below according to the discrete data,

$$-L_h U_h + D_h U_h = b_h \quad \text{in} \quad \Omega_h, \tag{34}$$

$$B_h U_h = b_h, \qquad \text{on} \quad \partial\Omega_h, \tag{35}$$

where L_h and D_h are the discrete versions of Laplacian operator, and the coefficient function $a(x,y)$ as defined in last section, and B_h is the boundary operator. The L_h and D_h in (34) share the same stencil structure as in (24) and (25). When the mesh is refined by doubling h to H, the value of b_H will be sampled directly on the refined interior node, but the boundary value of b_H is collected in the refined grid nodes on the boundary in the same way. The corresponding coefficient matrix for the discrete governing equation (34) on coarse mesh H is a submatrix of that for mesh h.

This makes it possible for applying multigrid method to solve the linear system (31) efficiently. Once U_h is determined, the solution of (31) is set as $U = U_h$.

The main idea of multigrid method lies in reducing the high frequency components of solution error on fine mesh, and that the low frequency errors are solved on coarse grids. Due to the fact that low frequency components of the errors can not

be effectively reduced, the residual is restricted to coarser grids such that the low frequency errors are solved therein. Such determined errors are then interpolated onto fine meshes to provide some correction to the already obtained solution on fine meshes. Over the process, relaxation is needed on fine mesh to improve the solution accuracy, while restriction operator is imposed from fine mesh to coarse mesh and interpolation is needed from coarse mesh to fine mesh. Such process is iterated until the solution meets certain accuracy.

The relaxation is used to reduce errors in two aspects. One is for smoothing the high frequency component on the fine mesh. The other is used after the obtained errors are interpolated to add correction on solution on fine mesh. In this case, the relaxation is to reduce the error introduced from interpolation. In this work, we adopt the red-black Gauss-Seidal smoother as the relaxation scheme. The two-grid correction scheme as the simplest multigrid adopts only two level of meshes to achieve the solution correction via iteration between restriction and interpolation. We use such two-grid correction scheme to illustrate the way we apply multigrid scheme to obtain the solution of the linear system (31).

Algorithm 1 Two-grid correction scheme

1: The initial guess for U_h is set as 0.
2: Relax Eq.(34) v_1 times on the fine grid.
3: Compute the residual on fine grid, $r_n = b_h + L_h U_h - D_h U_h$.
4: **for** L_2 error of residual $r_n >$ certain tolerance **do**
5: Restrict the residual to coarse grid $r_H = R_h^H r_h$ with the restriction operator R_h^H, where $H = 2h$.
6: Exactly solve for the solution e_H of the residual problem on coarse grid

$$-L_H e_H + D_H e_H = r_H. \tag{36}$$

7: Interpolate the correction to fine grid $e_h = I_H^h e_H$.
8: Apply the correction to get a better approximation $U_h := U_h + e_h$.
9: Relax the Eq.(34) v_2 times on the fine grid.
10: Compute the residual on fine grid, $r_n = b_h + L_h U_h - D_h U_h$.
11: **end for**

The above process continues until convergence. In our multigrid approach, we generalize the above process to more grid levels to avoid the exact correction solution for Eq.(36) in case that the cost is still very high on that coarse grid. This promotes the necessity of even coarser grids for correction solutions. The above procedure demonstrates the two-grid correction scheme using V-cycle. In our algorithm, we deploy V-cycle for the multigrid solver rather than the full multigrid approach since V-cycle tends to be more efficient in our practical implementation.

Restriction and interpolation are two needed transfer operators in the correction procedure. In our implementation, it is sufficient to adopt the following standard full-weighting stencil for restriction,

$$R_h^H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^H, \tag{37}$$

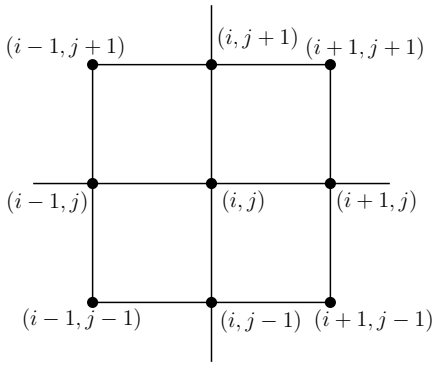


Figure 4. The needed nine points on fine mesh for restriction or interpolation in the multigrid approach.

and the following standard interpolation scheme

$$I_H^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_H^h. \tag{38}$$

The subscript h and superscript H in R_h^H indicate restriction from fine mesh of mesh size h to coarse mesh H, where $H = 2h$. As shown in Fig.4, the restricted function value at (i, j) on coarse mesh comes from the nine function values at the highlighted nine grid points on fine mesh with the corresponding weights in matrix R_h^H. On the other hand, the subscript H and superscript h in I_H^h indicate interpolation from coarse mesh H to fine mesh of mesh size h, where $H = 2h$. Through weight matrix I_H^h , function value at (i, j) contributes to nine interpolated function values at the nine grid points shown in Fig.4. To be clear, the value at (i, j) on coarse mesh is multiplied respectively by the nine weights in matrix I_H^h to give a contribution to the to-be-determined values at the nine grid points on fine mesh.

For the practical implementation of our multigrid algorithm for problem (34)(35), we set both v_1 and v_2 equal to 2. For the exact solution on the coarsest grid, we use biconjugate gradient iterative solver [31]. The iteration stopping criteria is set as error tolerance equal to 10^{-10}, or maximal iteration number equal to 5000. Besides, the multigrid stopping criteria is the residual L_2 error less than 10^{-10}. We may use number of grid levels flexibly for the multigrid algorithm based on the mesh refinement.

3. Numerical experiments.

In this section, numerical experiments are carried out to demonstrate the accuracy, efficiency and stability of the proposed two AMIB algorithms for solving parabolic interface problems. For sake of notation, we name the AMIB method coupled with the GMRES approach on Schur complement as AMIB-GMRES, while the AMIB method associated with LU decomposition is called as AMIB-LU. For comparison purpose, the standard MIB method [42] with Crank-Nicolson scheme for time-stepping is also implemented.

For simplicity, a square domain with uniform number of partitions in each direction is assumed. We consider the number as $n_x = x_y$ in each direction. Time integration will be carried out from $t = 0$ until a stopping time $t = T$ with n_t

steps and $\Delta t = T/n_t$. The numerical accuracy and convergence will be tested by comparing the numerical solutions with exact solutions under the L_∞ and L_2 norm.

The iteration solver used in MIB method is biconjugate gradient iterative solver [31]. The maximal iteration number is 5000, and the tolerance error is set as 10^{-10}. The number of mesh levels in the multigrid algorithm for AMIB is chosen as 4 for $n_x = 16$, as 5 for $n_x = 32$, as 6 for $n_x = 64$, and as 7 for all $n_x > 64$. For multigrid algorithm application, we have to set n_x to be 2^k for a positive integer k.

All the experiments were carried out on MacBook Pro with 8.00 RAM and Intel 2.3 GHz Intel Core i5.

3.1. **Numerical accuracy. Example 1.** In this example, we consider a parabolic problem on a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with a circular interface defined by $r^2 := x^2 + y^2 = \frac{1}{4}$. The exact solution to this problem is

$$u(x,y) = \begin{cases} \sin(t) + e^{-x-y}, & r \le 0.5, \\ \sin(t) + e^{x+y}, & \text{otherwise}, \end{cases} \tag{39}$$

with the diffusion coefficients

$$\beta = \begin{cases} 1, & r \le 0.5, \\ 10, & \text{otherwise}. \end{cases}$$

Here we call β as β^+ when it is outside of Γ, and β^- when it is inside of the interface Γ. The interface jump conditions on an interface point $(x,y) = (\frac{1}{2}\cos(\theta), \frac{1}{2}\sin(\theta))$ can be derived as below:

$$\begin{aligned}
[u] =& e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))} - e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))} \\
[\beta u_n] =& \beta^+(\cos(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))} + \sin(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))}) \\
&+ \beta^-(\cos(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))} + \sin(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))}) \\
[u_\tau] =& \beta^+(\sin(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))} - \cos(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))}) \\
&+ \beta^-(\sin(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))} - \cos(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))})
\end{aligned}$$

It follows the same principle in below notations. The source term $f(x,y)$ is related to the above designated solution,

$$f(x,y) = \begin{cases} (\cos(t) - 2\beta^- \sin(t))e^{-x-y}, & r \le 0.5, \\ (\cos(t) - 2\beta^+ \sin(t))e^{x+y}, & \text{otherwise}. \end{cases}$$

The initial condition and boundary conditions can be easily determined by the analytical solutions. Beside, we may observe that both the zeroth and first order jump conditions on the interface are time-independent. To test spatial convergence rate, we consider the fixed stopping time at $T = 1$ with the time increment $\Delta t = 10^{-3}$ being small enough such that temporal error will not interfere with the examination of the spatial convergence. Table 1 clearly reveals that the spatial discretization reaches the second order convergence in space.

On the other hand, the temporal convergence needs to be tested. For such concern, the spatial partition in each direction is set to be $n_x = 512$, and various time steps are deployed. The results in Table 2 indicates that the second order temporal convergence can be monitored from the second row to the last second row. Such second order convergence starts to occur from $\Delta t = 0.25$. The total CPU time of each time evolution methods is listed in the last column in Table 2, where we can compare the efficiency between these method. It can be observed

that if the number of time steps is small, AMIB-GMRES is superior than AMIB-LU. However, in the long run, AMIB-LU outperforms both AMIB-GMRES and MIB methods. Take the case of $n_t = 128$ for instance, CPU time of MIB is about 12 times larger than that of AMIB-LU. Similar results can also be found in the last row of Table 1 of spatial convergence test when partition number is 512 in each direction. The number of time evolution steps is 1000, leads to CPU cost of MIB being 14 times larger than that of AMIB-LU. The inefficiency of MIB or other interface algorithms implemented with iterative solvers for parabolic interface problems leads to the demand for the development of efficient solvers.

Table 1. Spatial convergence for example 1.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	2.08E-4	–	8.82E-4	–	0.94
[64, 64]	3.21E-5	2.70	1.69E-4	2.38	5.48
[128, 128]	7.52E-6	2.09	3.13E-5	2.43	26.6
[256, 256]	1.94E-6	1.95	9.07E-6	1.79	175
[512, 512]	5.66E-7	1.95	2.97E-6	1.79	1143
	AMIB-LU				
	L_2		L_∞		
	Error	Order	Error	Order	
[32, 32]	2.08E-4	–	8.82E-4	–	0.30
[64, 64]	3.19E-5	2.70	1.68E-4	2.38	1.49
[128, 128]	7.52E-6	2.09	3.13E-5	2.43	6.50
[256, 256]	1.94E-6	1.95	9.08E-6	1.79	29.8
[512, 512]	5.71E-7	1.95	2.99E-6	1.79	200
	MIB				
	L_2		L_∞		
	Error	Order	Error	Order	
[32, 32]	2.07E-4	–	8.75E-4	–	0.62
[64, 64]	3.19E-5	2.70	1.68E-4	2.38	4.76
[128, 128]	7.50E-6	2.08	3.13E-5	2.42	35.4
[256, 256]	1.94E-6	1.94	9.09E-6	1.79	336
[512, 512]	5.66E-7	1.95	2.97E-6	1.79	2935

Example 2. In this example, we consider a parabolic problem on a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with a circular interface defined by $r^2 := x^2 + y^2 = \frac{1}{4}$. The exact solution to this problem is

$$u(x,y) = \begin{cases} \sin(kx)\cos(ky)\cos(t), & r \le 0.5, \\ \cos(kx)\sin(ky)\cos(t), & \text{otherwise}, \end{cases} \tag{40}$$

The source terms can be correspondingly determined,

$$f(x,y) = \begin{cases} (2k^2\beta^- \cos(t) - \sin(t))\sin(kx)\cos(ky), & r \le 0.5, \\ (2k^2\beta^+ \cos(t) - \sin(t))\cos(kx)\sin(ky), & \text{otherwise}. \end{cases}$$

The zeroth and first order interface can be found to be time and space dependent, representing the most general jump conditions. In this example, we consider $k = 2$.

It is also reasonable to approximate the equation (4) with

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\beta_{i,j} \triangle t} = \frac{1}{2}\Delta u_{i,j}^n + \frac{1}{2}\Delta u_{i,j}^{n+1} + \frac{1}{\beta_{i,j}} f_{i,j}^{n+\frac{1}{2}}, \tag{41}$$

with the difference to Eq.(5) on the approximation to f at time $t = t_n + 0.5\Delta t$ by $f_{i,j}^{n+\frac{1}{2}}$ in Eq.(41), while $\frac{1}{2}(f_{i,j}^n + f_{i,j}^{n+1})$ is used in Eq.(6). We are interested in the numerical impacts with such different treatment for the source term f. Table

Table 2. Temporal convergence for example 1.

n_t	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	1.32E-4	–	5.46E-4	–	2.76
4	8.98E-5	0.56	3.25E-4	0.75	5.28
8	2.48E-5	1.86	7.42E-5	2.13	9.73
16	6.21E-6	2.00	1.87E-5	1.99	19.36
32	1.56E-6	1.99	4.88E-6	2.02	37.88
64	6.26E-7	1.32	3.21E-6	0.60	76.9
128	5.60E-7	0.16	2.99E-6	0.10	148.4
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	1.32E-4	–	5.46E-4	–	58
4	8.98E-5	0.56	3.25E-4	0.75	59.4
8	2.48E-5	1.86	7.42E-5	2.13	61.5
16	6.21E-6	2.00	1.87E-5	1.99	63.2
32	1.56E-6	1.99	4.88E-6	1.94	64.7
64	6.34E-7	1.30	3.24E-6	0.59	66.8
128	5.76E-7	0.13	3.04E-6	0.09	74
	MIB				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	1.32E-4	–	5.46E-4	–	21.97
4	8.98E-5	0.56	3.25E-4	0.75	42.34
8	2.48E-5	1.86	7.42E-5	2.14	81.72
16	6.21E-6	2.00	1.87E-5	1.98	153.34
32	1.56E-6	1.99	4.88E-6	1.94	287.03
64	6.27E-7	1.31	3.22E-6	0.60	533
128	5.69E-7	0.14	3.02E-6	0.09	863

3 shows their numerical comparisons between scheme (41) and (5) with $f^{n+\frac{1}{2}}$ indicating scheme (41) and the other part for scheme (5). In this experiment, the partition number in each direction is $n_x = 512$. The temporal convergence can all be determined according to the results. It can be found that the solution accuracy from (5) with $n_t = 128$ is comparable to that of (41) with $n_t = 256$. Such accuracy improvement justifies that scheme (5) is superior to (41) in terms of accuracy when dealing with the interface jump conditions that are time and space independent. On the other hand, it can be found in Tables 4 and 5 that the time costs from scheme (5) and (41) are basically the same for each n_t. From this point of view, scheme (5) is a better choice for time evolution than (41). Thus we will focus on scheme (5) in the following discussion to deal with the parabolic interface problem under all kinds of circumstances.

The error versus time partition increment Δt on log-log scale from scheme (5) is plotted on Fig.5, from which we can observe that the temporal second order accuracy is obtained for both AMIB-GMRES and AMIB-LU methods.

On the other hand, the spatial convergence are investigated on the two methods. The stopping time is fixed as $T = 2$, and the time increment is set as 0.002. The second order spatial convergence of the two approaches concerned with source term f can be observed from the results in Table 4. The comparison shows no much difference on spatial accuracy by scheme (5) and (41).

We then want to study the impact of ratio of $\beta^+ : \beta^-$ on the stability for scheme (5). We will fix one of two β values equal to 1, then vary the value of the other one to make the ratio large or small enough. We fix the partition number in each direction as 128, and the stopping time equal to 2 with time increment 0.002. It can be observed from the Table 5 that the two AMIB methods are all stable enough to deal with large contrast of the coefficients. On the other hand, the

Table 3. Temporal convergence for example 2.

n_t	AMIB-GMRES-$f^{n+\frac{1}{2}}$					AMIB-GMRES				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
2	3.22E-2	–	7.55E-2	–	3.04	1.39E-3	–	3.43E-3	–	2.81
4	5.56E-3	2.53	1.57E-2	2.27	5.9	3.14E-4	2.15	7.77E-4	2.14	5.69
8	9.47E-4	2.55	2.94E-3	2.42	12.5	7.59E-5	2.05	1.88E-4	2.05	11.2
16	1.67E-4	2.50	4.35E-4	2.76	23	1.89E-5	2.01	4.68E-5	2.01	22
32	3.82E-5	2.13	8.32E-5	2.39	45	5.00E-6	1.92	1.21E-5	1.95	45.4
64	9.68E-6	1.99	2.13E-5	1.98	87	1.44E-6	1.80	3.30E-6	1.87	96.3
128	2.579E-6	1.91	6.05E-6	1.82	168	6.60E-7	1.13	2.84E-6	0.22	170
256	9.03E-7	1.53	3.25E-6	0.91	334					
	AMIB-LU-$f^{n+\frac{1}{2}}$					AMIB-LU				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
2	3.22E-2	–	7.55E-2	–	65	1.40E-3	–	3.44E-3	–	57
4	5.56E-3	2.53	1.57E-2	2.27	69	3.15E-4	2.15	7.79E-4	2.15	59.6
8	9.47E-4	2.55	2.94E-3	2.42	62	7.64E-5	2.04	1.89E-4	2.04	58.9
16	1.67E-4	2.50	4.35E-4	2.76	62	1.90E-5	2.01	4.70E-5	2.01	57.7
32	3.82E-5	2.13	8.32E-5	2.39	60.6	4.82E-6	1.98	1.17E-5	2.01	60.7
64	9.65E-6	1.99	2.11E-5	1.98	66	1.34E-6	1.85	3.05E-6	1.94	64.6
128	2.57E-6	1.91	5.98E-6	1.82	75.1	6.16E-7	1.12	2.70E-6	0.18	75.9
256	8.88E-7	1.53	3.18E-6	0.91	91					
512	5.69E-7	0.64	2.76E-6	0.20	129					

Table 4. Spatial convergence for example 2.

$[n_x, n_y]$	AMIB-GMRES-$f^{n+\frac{1}{2}}$					AMIB-GMRES				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
[32, 32]	1.15E-4	–	5.16E-4	–	1.11	2.25E-4	–	5.16E-4	–	1.19
[64, 64]	1.10E-4	1.03	3.42E-4	0.59	7.39	1.10E-4	1.03	3.42E-4	0.59	7.86
[128, 128]	1.44E-5	2.93	6.40E-5	2.42	42	1.44E-5	2.93	6.40E-5	2.42	43
[256, 256]	2.06E-6	2.82	1.54E-5	2.61	264	2.04E-6	2.82	1.04E-5	2.61	258
[512, 512]	5.18E-7	2.01	2.65E-6	1.99	1457	5.05E-7	2.01	2.62E-6	1.99	1463
	AMIB-LU-$f^{n+\frac{1}{2}}$					AMIB-LU				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
[32, 32]	1.15E-4	–	5.16E-4	–	0.34	2.25E-4	–	5.16E-4	–	0.7
[64, 64]	1.10E-4	0.06	3.42E-4	0.59	1.66	1.10E-4	1.03	3.42E-4	0.59	1.78
[128, 128]	1.44E-5	2.93	6.40E-5	2.42	8.04	1.44E-5	2.93	6.40E-5	2.43	8.76
[256, 256]	2.06E-6	2.81	1.05E-5	2.61	39.4	2.05E-6	2.91	1.04E-5	1.79	41.6
[512, 512]	5.28E-7	1.96	2.68E-6	1.97	231	5.15E-7	1.99	2.65E-6	1.99	227

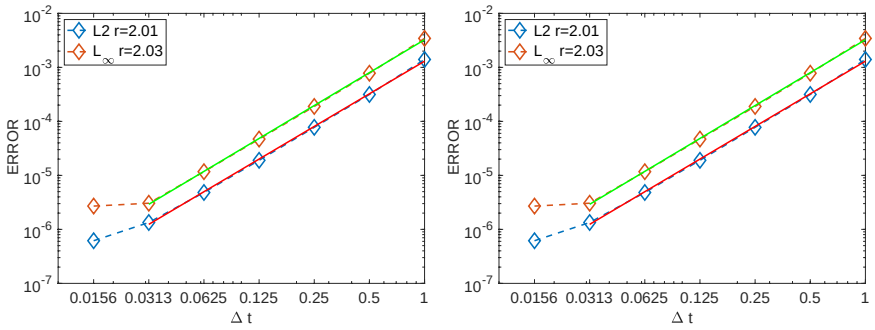


Figure 5. The temporal convergence for example 2. The left is for AMIB-GMRES, and the right is for AMIB-LU.

varying coefficient contrasts seem to have larger impact on the CPU time of AMIB-GMRES than AMIB-LU. The CPU time of AMIB-GMRES become twice expansive

when the ratio of $\beta^+ : \beta^-$ is raised from $10:1$ to $10000:1$. This is also the same case for AMIB-GMRES when ratio of $\beta^+ : \beta^-$ is raised from $1:10$ to $1:10000$. The milder impact on AMIB-LU is due to the direct solver by LU decomposition rather than iterative solvers whose conditional number can be affected by the ratio of $\beta^+ : \beta^-$. Such conclusion is based on the comparison between the two AMIB methods vertically in Table 5.

Note that the "Small β side" indicates the the MIB extrapolation part is fulfilled by the function values from the domain related to the small β. The "Large β" side indicates the domain related to the large β in a similar fashion. From the horizontal comparison for each AMIB method in Table 5, MIB fictitious values with the "Small β side" extrapolation is superior to the one with the "Large β side", because the CPU time is less than that in the other way around for AMIB-GMRES. This justifies the selection of β in remark 2 for concern of being well-conditioned. Hence, it is preferable to adopt the fictitious values with the extrapolation part fulfilled using the function values from domain of "small β." The accuracy of the approximate solutions concerned with the "Small" or "Large" β has no much big difference using AMIB-GMRES or AMIB-LU in this example. In the following examples, we will adopt the "Small β side" strategy for both AMIB schemes.

Table 5. Spatial convergence for example 2 with respect to different coefficient contrasts. $n_x = 128$, $\Delta t = 0.002$.

$[\beta^+, \beta^-]$	AMIB-GMRES					
	Small β side			Large β side		
	L_2	L_∞	CPU time(s)	L_2	L_∞	CPU time(s)
[10000, 1]	1.63E-5	5.11E-5	71.4	1.86E-5	6.70E-5	92.6
[1000, 1]	1.63E-5	5.10E-5	65.4	1.86E-5	6.69E-5	92.8
[100, 1]	1.61E-5	5.04E-5	52.4	1.84E-5	6.60E-5	69.3
[10, 1]	1.47E-5	4.43E-5	31.4	1.65E-5	5.74E-5	33.8
[1, 10]	9.48E-6	4.71E-5	36.2	7.01E-6	3.44E-5	36.2
[1, 100]	6.03E-5	1.56E-4	58.3	6.20E-5	1.45E-4	66.4
[1, 1000]	4.66E-4	9.00E-4	67.8	6.38E-4	1.19E-3	81.6
[1, 10000]	5.26E-4	8.52E-4	71.2	5.62E-3	1.02E-2	89.5
$[\beta^+, \beta^-]$	AMIB-LU					
	Small β side			Large β side		
	L_2	L_∞	CPU time(s)	L_2	L_∞	CPU time(s)
[10000, 1]	1.63E-5	5.11E-5	10.9	1.86E-5	6.72E-5	10.9
[1000, 1]	1.63E-5	5.10E-5	10.58	1.86E-5	6.71E-5	10.46
[100, 1]	1.61E-5	5.04E-5	8.8	1.84E-5	6.61E-5	9.1
[10, 1]	1.47E-5	4.43E-5	6.8	1.66E-5	5.75E-5	6.9
[1, 10]	9.48E-6	4.71E-5	7.0	7.03E-6	3.43E-5	6.9
[1, 100]	6.03E-5	1.56E-4	9.2	6.20E-5	1.45E-4	8.5
[1, 1000]	4.66E-4	9.00E-4	8.6	6.38E-4	1.19E-3	8.6
[1, 10000]	5.29E-4	8.56E-4	9.0	5.63E-3	1.02E-2	8.7

Example 3. We are interested in the performance of the two AMIB methods on complicated interface shape. In this example, we focus on the case of interface shape governed by parametric function

$$r = 0.5 + 0.1\sin(3\theta), \tag{42}$$

embedded in the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The solution and all the related boundary and interface conditions are set as in the last example. The stopping time is given as $T = 1$, and the coefficients are $(\beta^+, \beta^-) = (100, 1)$. The spatial accuracy and convergence are examined with the small time increment $\Delta t = 0.001$. The numerical results in Table 6 on the successively refined mesh demonstrate the second order spatial accuracy.

Table 6. Spatial convergence for example 3.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	6.70E-4	–	2.17E-3	–	1.67
[64, 64]	1.52E-4	2.14	5.05E-4	2.10	9.82
[128, 128]	3.63E-5	2.07	1.39E-4	1.86	53
[256, 256]	8.62E-6	2.07	3.52E-5	1.98	399
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	6.70E-4	–	2.17E-3	–	0.42
[64, 64]	1.52E-4	2.14	5.05E-4	2.10	1.94
[128, 128]	3.63E-5	2.07	1.39E-4	1.86	8.54
[256, 256]	8.62E-6	2.07	3.52E-5	1.98	45.5

We next examine the temporal order of the two AMIB methods by setting $n_x = 512$, and the stopping time as $T = 1$. As the time evolution scheme (5) is efficient enough for the temporal convergence, the numerical results in Table 7 show that both AMIB methods obtain their limiting convergence very quickly, leading to the temporal second order observed from $n_t = 2$ to $n_t = 4$, and the reduced order in further steps.

Table 7. Temporal convergence for example 3.

n_t	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	4.83E-5	–	2.18E-4	–	4.1
4	1.39E-5	1.80	5.61E-5	1.96	7.9
8	5.22E-6	1.41	1.92E-5	1.55	15.9
16	2.68E-6	0.96	9.40E-6	1.03	31.5
32	2.15E-6	0.32	8.56E-6	0.14	66.6
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	4.62E-5	–	2.13E-4	–	62.7
4	1.23E-5	1.91	5.46E-5	1.96	63.2
8	4.25E-6	1.53	1.60E-5	1.77	63.5
16	2.57E-6	0.73	9.15E-6	0.81	67.6
32	2.23E-6	0.20	8.88E-6	0.04	68.1

Since the AMIB algorithm achieves second order accuracy for both spatial and temporal discretization, we test the computing efficiency by refining the spatial and temporal steps simultaneously. See the Table 8 with stopping time $T = 1$. We can observe that the second overall accuracy of the three methods are all achieved after the spatial and temporal partition are refined at the same time. Such detected second order accuracy can validate the temporal second order despite the reduced order observation in the Table 7. The CPU time comparison of the three methods further demonstrate the better efficiency of the two AMIB method over MIB method. Besides, the time ratios of MIB method with mesh refinement are higher than the two AMIB methods, meaning the MIB is the most expansive algorithm among the three. AMIB-LU seems a little more efficient than AMIB-GMRES thanks to its relatively smaller time ratio when the mesh is refined. The reason for its better efficiency is mainly due to a smaller average number of multigrid solver (ANMG) being used in each time step as recorded in the last column in Table 8. It can be seen that the average number for AMIB-GMRES mildly increases as the mesh is refined. This is due to the fact that the iteration number of GMRES solver in solving (30) weakly

Table 8. Efficiency test for example 3.

(n_x, n_t)	AMIB-GMRES						
	L_2		L_∞		CPU time(s)	time ratio	ANMG
	Error	Order	Error	Order			
(65, 20)	1.53E-4	–	5.08E-4	–	0.25		31.3
(129, 40)	3.63E-5	2.08	1.39E-4	1.87	2.22	8.9	35.2
(257, 80)	8.63E-6	2.07	3.52E-5	1.98	26.7	12.0	48.1
(513, 160)	2.16E-6	2.00	8.88E-6	1.99	342.9	12.8	57.3
	AMIB-LU						
	L_2		L_∞		CPU time(s)	time ratio	
	Error	Order	Error	Order			
(65, 20)	1.53E-4	–	5.08E-4	–	0.15		15.2
(129, 40)	3.63E-5	2.08	1.39E-4	1.87	1.15	7.7	15.2
(257, 80)	8.63E-6	2.07	3.52E-5	1.98	10.1	8.8	15.2
(513, 160)	2.14E-6	2.01	8.79E-6	2.00	100.8	10	15.2
	MIB						
	L_2		L_∞		CPU time(s)	time ratio	
	Error	Order	Error	Order			
(65, 20)	1.47E-4	–	5.05E-4	–	0.43		
(129, 40)	3.62E-5	2.08	1.42E-4	1.87	5.94	13.8	
(257, 80)	8.53E-6	2.07	3.56E-5	1.98	113.4	19.1	
(513, 160)	NA	NA	NA	NA	NA	NA	

depends on the grid numbers. However, the average number remains constant for AMIB-LU, since the majority of the number of multigrid solver used is on forming the explicit matrix of left side of (30).

It can be noticed that the numerical results for MIB in the last row of Table 8 are not available due to failure of convergence from the biconjugate gradient solver [31]. The non-convergence is not caused by stability issue on the algorithm, as a different iterative solver, such as GMRES, can produce convergent iterative solution in our numerical test. This demonstrated that the performance of the AMIB methods does not heavily depend on the iterative solvers, while the MIB method does.

Example 4. We are interested in the stability performance of the two methods on complicated interface shape. In this example, we focus on the case of interface shape governed by parametric function

$$r = 0.5 + 0.1\sin(5\theta), \tag{43}$$

embedded in the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$.

The exact solution to this problem is

$$u(x,y) = \begin{cases} \cos(t)e^{-x-y}, & \text{in } \Omega^-, \\ \cos(t)e^{x+y}, & \text{in } \Omega^+, \end{cases} \tag{44}$$

with the diffusion coefficients

$$\beta = \begin{cases} 1, & \text{in } \Omega^-, \\ 100, & \text{in } \Omega^+. \end{cases}$$

The spatial convergence is validated by refining the mesh with the stopping time $T = 2$, and time increment $\Delta t = 0.002$. In Table 9, the second order accuracy can be observed for both AMIB methods.

We next examine the temporal order of the two AMIB methods by setting $n_x = 512$, and the stopping time as $T = 0.001$. The second order temporal convergence can be observed in Table 10 from n_t equal to 2 up to 16.

To numerically analyze the stability of the two AMIB schemes, we examine different time stepping. We choose $n_x = 128$ and stopping time T equal to $10^4\Delta t$ as shown in Fig. 6. All numerical solutions are found to be stable with large temporal

Table 9. Spatial convergence for example 4.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.08E-3	–	6.81E-3	–	1.67
[64, 64]	6.26E-4	2.70	1.12E-3	2.60	12.5
[128, 128]	2.26E-4	1.47	3.92E-4	1.51	89
[256, 256]	9.01E-5	1.33	1.54E-4	1.35	524
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.08E-3	–	6.81E-3	–	0.48
[64, 64]	6.26E-4	2.70	1.12E-3	2.60	2.50
[128, 128]	2.25E-4	1.47	3.91E-4	1.52	12.4
[256, 256]	8.83E-5	1.33	1.51E-4	1.37	57

Table 10. Temporal convergence for example 4.

n_t	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	3.28E-3	–	5.60E-3	–	5.2
4	8.22E-4	2.00	1.38E-3	2.02	10.1
8	2.23E-4	1.88	3.65E-4	1.92	20.2
16	7.45E-5	1.58	1.16E-4	1.65	40.1
32	3.75E-5	0.99	6.05E-5	0.94	76.6
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	3.28E-3	–	5.60E-3	–	75.8
4	8.21E-4	2.00	1.38E-3	2.02	75.3
8	2.23E-4	1.88	3.64E-4	1.92	77.7
16	7.41E-5	1.59	1.15E-4	1.66	80.1
32	3.75E-5	0.98	6.10E-5	0.94	82.7

increments. And the corresponding numerical errors are bounded as shown in Fig. 6. This justifies the reliable stability of our methods for solving parabolic interface problem.

Example 5. We next study a problem with multiple subdomains. Consider a 2D parabolic equation

$$u_t = (\beta u_x)_x + (\beta u_y)_y + q(x, y)$$

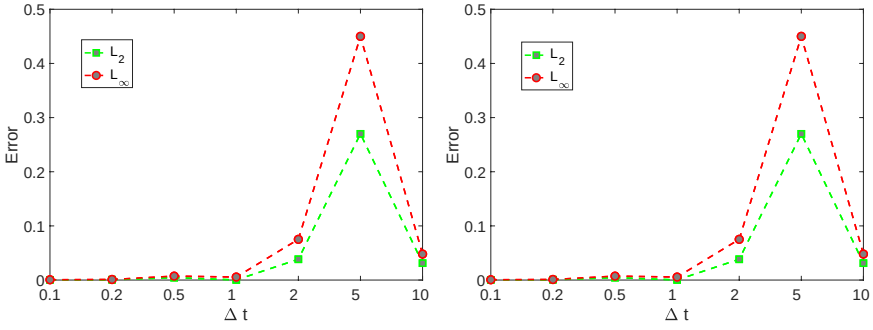


Figure 6. Stability test for the two AMIB methods. The left one is for AMIB-GMRES, and the right one is for AMIB-LU. All the numerical errors are bounded with $n_x = 128$, and $T = 10^4 \Delta t$.

defined in a square domain $\Omega = [-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, which is divided into three regions by two ellipse shape interface:

$$\Gamma_L: \left(\frac{x+\frac{\pi}{6}}{\frac{\pi}{9}}\right)^2 + \left(\frac{y+\frac{\pi}{7}}{\frac{\pi}{12}}\right)^2 = 1$$
$$\Gamma_R: \left(\frac{x-\frac{\pi}{6}}{\frac{\pi}{12}}\right)^2 + \left(\frac{y-\frac{\pi}{7}}{\frac{\pi}{9}}\right)^2 = 1$$

Three pieces of solutions are defined in each subdomain

$$u(x,y) = \begin{cases} \cos(t)\cos(kx)\sin(ky) & \text{inside } \Gamma_L, \\ \cos(t)\cos(kx)\sin(ky) & \text{inside } \Gamma_R, \\ \cos(t)e^{x+y} & \text{otherwise}, \end{cases}$$

where parameter k is set to be 5. The diffusion coefficients are also defined respectively as

$$\beta = \begin{cases} 100, & \text{inside } \Gamma_L, \\ 10, & \text{inside } \Gamma_R, \\ 1, & \text{otherwise}, \end{cases}$$

and the parameter k is set as 5. The source term $q(x,y)$ can be determined by the analytical solutions, and Dirichlet boundary conditions are assumed.

The second order spatial and temporal accuracy of solutions have been examined through above examples. Besides, we want to further investigate the convergence rate of the gradient recovery. Second order central difference is adopted to approximate the gradients with fictitious values replacing obtained numerical real function values when the stencil cuts through the interface. Table 11 shows that second order convergence is achieved for both solutions and gradients from the three methods in solving the multi-domain problem. With the confirmed second orders of the three methods, we pay attention to the comparison of the methods on overall efficiency for solving multi-domain problem . Table 11 demonstrates the efficiency test, where it can be observed that AMIB-LU outperform the other methods for solving such problems. The last column of Table 11 shows that AMIB-LU has smaller time ratio than AMIB-GMRES or AMIB with mesh refinement. In particular, the computing time by MIB is almost 19 times larger than that of AMIB-LU in dealing the multi-domain problem on the $(n_x, n_t) = (513, 160)$ mesh, further demonstrating the efficiency improvement from AMIB-LU.

We note that there exist more challenging multi-material interface problems, in which two or more material interfaces join together or cross each other. A second order accurate MIB method has been developed in [38] for elliptic equations with intersecting interfaces. Based on such a MIB method, an augmented MIB method could be developed for intersecting interface problems. This is an interesting direction to explore in the future.

Example 6. We investigate a physical problem modeled by the heat equation without analytical solutions

$$u_t = \nabla \cdot (\beta \nabla u)$$

subject to the homogeneous interface condition

$$[u] = 0 \tag{45}$$
$$[\beta u] = 0 \tag{46}$$

Table 11. Efficiency test for example 5..

(n_x, n_t)	AMIB-GMRES					
	Solution					
	L_2		L_∞		CPU time(s)	time ratio
	Error	Order	Error	Order		
$(65, 20)$	3.00E-2	–	7.43E-2	–	0.37	
$(129, 40)$	8.96E-3	1.74	2.24E-2	1.73	3.96	10.7
$(257, 80)$	1.57E-3	2.51	4.02E-3	2.48	44.46	11.2
$(513, 160)$	3.23E-4	2.28	8.28E-4	2.28	562	12.6
	Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$(65, 20)$	9.38E-2	–	0.37	–		
$(129, 40)$	2.83E-2	1.732	0.12	1.62		
$(257, 80)$	5.07E-3	2.48	2.15E-2	2.48		
$(513, 160)$	1.21E-3	2.07	5.29E-3	2.02		
	AMIB-LU					
	Solution					
	L_2		L_∞		CPU time(s)	time ratio
	Error	Order	Error	Order		
$(65, 20)$	3.00E-2	–	7.43E-2	–	0.15	
$(129, 40)$	8.95E-3	1.75	2.23E-2	1.74	1.22	8.1
$(257, 80)$	1.58E-3	2.50	4.05E-3	2.46	11.7	9.6
$(513, 160)$	3.71E-4	2.09	9.54E-4	2.09	127	10.9
	Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$(65, 20)$	9.38E-2	–	0.37	–		
$(129, 40)$	2.82E-2	1.73	0.12	1.62		
$(257, 80)$	5.11E-3	2.46	2.17E-2	2.47		
$(513, 160)$	1.21E-3	2.08	5.29E-3	2.04		
	MIB					
	Solution					
	L_2		L_∞		CPU time(s)	time ratio
	Error	Order	Error	Order		
$(65, 20)$	3.61E-2	–	9.21E-2	–	0.44	
$(129, 40)$	1.10E-2	1.74	2.87E-2	1.68	7.38	16.8
$(257, 80)$	1.41E-3	2.46	3.90E-3	2.88	143	19.4
$(513, 160)$	2.17E-4	2.09	9.89E-4	1.98	2372	16.6
	Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$(65, 20)$	0.12	–	1.93	–		
$(129, 40)$	3.53E-2	1.76	0.44	2.13		
$(257, 80)$	4.51E-3	2.97	0.13	1.76		
$(513, 160)$	9.73E-4	2.21	2.42E-2	2.43		

on the interface $r^2 := x^2 + y^2 = \frac{1}{4}$ embedded in the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The diffusion coefficients

$$\beta = \begin{cases} 100, & \text{in } \Omega^-, \\ 1, & \text{in } \Omega^+. \end{cases}$$

The Dirichlet boundary condition equal to zero is imposed at the boundary of given domain. Besides, the initial condition is defined as

$$u(x, y) = \begin{cases} e^{-(x^2+y^2)/\sigma^2} & \text{in } \Omega^-, \\ 0, & \text{in } \Omega^+, \end{cases} \tag{47}$$

with parameter $\sigma = 0.1$. As the analytical solution is not readily available, we generate the reference solution on certain mesh to examine the numerical performance of the two AMIB methods.

The spatial convergence is validated by refining the mesh under time increment $\Delta t = 0.001$ for stopping time $T = 1$. The reference solution is created on the mesh

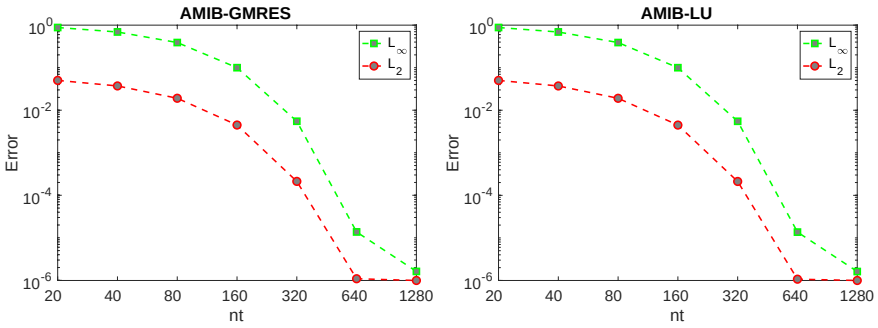


Figure 7. The numerical errors from two AMIB methods.

$(n_x, \Delta t) = (512, 0.0005)$. The second order accuracy can be observed for both AMIB methods in Table 12.

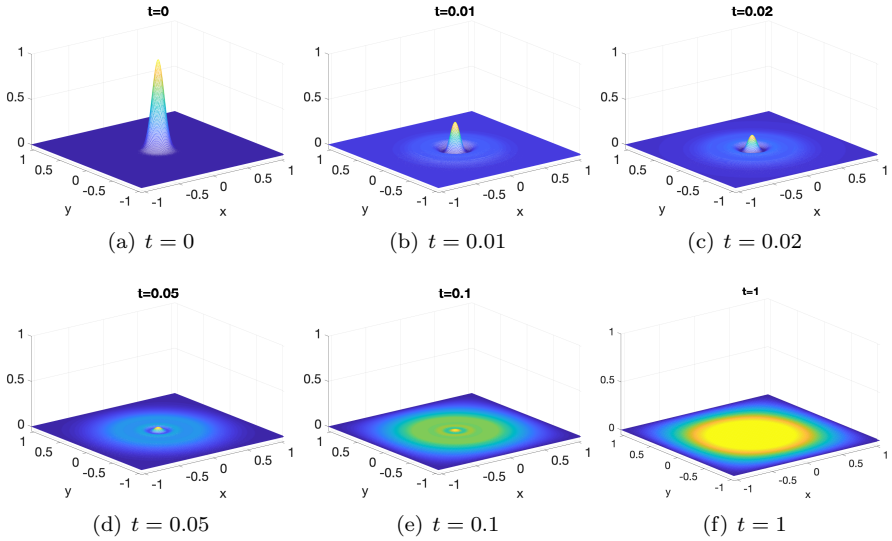
Table 12. Spatial convergence for example 6.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.09E-5	–	6.66E-5	–	0.65
[64, 64]	2.24E-5	0.87	3.63E-5	0.88	4.02
[128, 128]	5.04E-6	2.15	8.19E-6	2.15	22.3
[256, 256]	9.47E-7	2.41	1.56E-6	2.39	151
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.09E-5	–	6.66E-5	–	0.25
[64, 64]	2.23E-5	0.88	3.62E-5	0.88	1.36
[128, 128]	5.06E-6	2.14	8.22E-6	2.24	7.06
[256, 256]	9.97E-7	2.34	1.63E-6	2.33	34.9

The temporal convergence is validated in a similar fashion by refining time increment for a fixed stopping time $T = 1$, and the spatial discretization is based on mesh $(n_x, n_y) = (256, 256)$. The reference solution is obtained on the mesh $(n_x, \Delta t) = (512, 0.0005)$. Due to the practical physical phenomenon modeled by heat equation, the temperature quickly drops in a very short time, leading to large approximation errors when a large time step size is adopted as shown in Fig. 7. Nevertheless, small time steps can produce well approximated solutions. This is true for both AMIB-GMRES and AMIB-LU methods, especially when $n_t > 160$ is employed. For such reason, it is difficult to observe the expected second order temporal convergence.

The figure 8 shows solution evolution over time for this physical model. As discussed before, the temperature drops quickly in a short time. This can be observed in Fig. 7 when time is changing from 0 to 0.1. When time equals to 1, the temperature has dropped to a magnitude of 10^{-4}.

4. **Summary.** In this work, a new finite difference method has been proposed to solve parabolic interface problem with second order accuracy achieved in both time and space. The efficiency is fulfilled with multigrid method built in the Schur complement process in the framework of an augmented system. A simple multigrid solver is designed to solve the linear system due to the special structure of coefficient



(a) $t = 0$ (b) $t = 0.01$ (c) $t = 0.02$

(d) $t = 0.05$ (e) $t = 0.1$ (f) $t = 1$

Figure 8. The solution of heat equation evolves within time equal to 1. The rapid change during initial period indicates quick temperature drop.

matrix as part of the augmented system. Two approaches based on the multigrid solver are proposed in the time-step process, with one incorporated in GMRES iteration for each time evolution, and the other one used in the coefficient matrix formation of LU decomposition solver in the first time step. The latter approach with LU decomposition can be more efficient especially when a lot of time-stepping evolutions are needed for solving the given parabolic interface problem. Both AMIB methods are significantly faster than the standard MIB method. Moreover, being free of stability constraint, the present implicit method could be asymptotically faster than explicit schemes.

The immersed interface method (IIM) has been applied to variable coefficient elliptic equations [4, 21], and these studies demonstrated how to adopt numerical components, such as jump-corrected differences, augmented system, and multigrid solver, for variable coefficient problems. Thus, based on these studies, it is possible to generalize the present mulitgrid AMIB method for variable coefficient interface problems. This will be explored in the future.

Acknowledgments. This research is supported in part by the National Science Foundation (NSF) of USA under grant DMS-1812930.

REFERENCES

[1] L. Adams and T. P. Chartier, New geometric immersed interface multigrid solvers, *SIAM J. Sci. Comput.*, **25** (2004), 1516–1533.

[2] L. Adams and Z. Li, The immersed interface/multigrid methods for interface problems, *SIAM J. Sci. Comput.*, **24** (2002), 463–479.

[3] C. Attanayake and D. Senaratne, Convergence of an immersed finite element method for semilinear parabolic interface problems, *Appl. Math. Sci. (Ruse)*, **5** (2011), 135–147.

[4] P. A. Berthelsen, A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions, *J. Comput. Phys.*, **197** (2004), 364–386.
[5] F. Bouchon and G. H. Peichl, An immersed interface technique for the numerical solution of the heat equation on a moving domain, *Numerical Mathematics and Advanced Applications 2009*, Springer Berlin Heidelberg, (2010), 181–189.
[6] F. Bouchon and G. H. Peichl, The immersed interface technique for parabolic problems with mixed boundary conditions, *SIAM J. Numer. Anal.*, **48** (2010), 2247–2266.
[7] T. F. Chan and W. L. Wan, Robust multigrid methods for nonsmooth coefficient elliptic linear systems, *J. Comput. Appl. Math.*, **123** (2000), 323–352.
[8] Z. Chen and J. Zou, Finite element methods and their convergence for elliptic and parabolic interface problems, *Numer. Math.*, **79** (1998), 175–202.
[9] A. Coco and G. Russo, Finite-difference ghost-point multigrid methods on Cartesian grids for elliptic problems in arbitrary domains, *J. Comput. Phys.*, **241** (2013), 464–501.
[10] A. Coco and G. Russo, Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface, *J. Comput. Phys.*, **361** (2018), 299–330.
[11] J. Douglas Jr., On the numerical integration of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}$ by implicit methods, *J. Soc. Indust. Appl. Math.*, **3** (1955), 42–65.
[12] J. Douglas and D. Peaceman, Numerical solution of two-dimensional heat-flow problems, *AIChEJ.*, **1** (1955), 505–512.
[13] W. Feng, X. He, Y. Lin and X. Zhang, Immersed finite element method for interface problems with algebraic multigrid solver, *Commun. Comput. Phys.*, **15** (2014), 1045–1067.
[14] H. Feng, G. Long and S. Zhao, An augmented matched interface and boundary (MIB) method for solving elliptic interface problem, *J. Comput. Appl. Math.*, **361** (2019), 426–443.
[15] H. Feng and S. Zhao, FFT-based high order central difference schemes for the three-dimensional Poisson's equation with various types of boundary conditions, *J. Comput. Phys.*, **410** (2020), 109391, 24 pp.
[16] H. Feng and S. Zhao, A fourth order finite difference method for solving elliptic interface problems with the FFT acceleration, *J. Comput. Phys.*, **419** (2020), 109677, 25 pp.
[17] F. Gibou and R. Fedkiw, A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem, *J. Comput. Phys.*, **202** (2005), 577–601.
[18] J. D. Kandilarov and L. G. Vulkov, The immersed interface method for a nonlinear chemical diffusion equation with local sites of reactions, *Numer. Algorithms*, **36** (2004), 285–307.
[19] J. D. Kandilarov and L. G. Vulkov, The immersed interface method for two-dimensional heat-diffusion equations with singular own sources, *Appl. Numer. Math.*, **57** (2007), 486–497.
[20] Z. Li, X. Chen and Z. Zhang, On multiscale ADI methods for parabolic PDEs with a discontinuous coefficient, *Multiscale Model. Simul.*, **16** (2018), 1623–1647.
[21] Z. Li, H. Ji and X. Chen, Accurate solution and gradient computation for elliptic interface problems with variable coefficients, *SIAM J. Numer. Anal.*, **55** (2017), 570–597.
[22] Z. Li and A. Mayo, ADI methods for heat quations with discontinuous along an arbitrary interface, *Proc. Sympos. Appl. Math.*, **48** (1993), 311–315.
[23] C. Li, Z. Wei, G. Long, C. Campbell, S. Ashlyn and S. Zhao, Alternating direction ghost-fluid methods for solving the heat equation with interfaces, *Comput. Math. Appl.*, **80** (2020), 714–732.
[24] C. Li and S. Zhao, A matched Peaceman–Achford ADI method for solving parabolic interface problems, *Appl. Math. Comput.*, **299** (2017), 28–44.
[25] T. Lin, Q. Yang and X. Zhang, Partially penalized immersed finite element methods for parabolic interface problems, *Numer. Methods Partial Differential Equations*, **31** (2015), 1925–1947.
[26] J. Liu and Z. Zheng, IIM-based ADI finite difference scheme for nonlinear convection–diffusion equations with interfaces, *Appl. Math. Model.*, **37** (2013), 1196–1207.
[27] J. Liu and Z. Zheng, A dimension by dimension splitting immersed interface method for heat conduction equation with interfaces, *J. Comput. Appl. Math.*, **261** (2014), 221–231.
[28] S. F. McCormick, *Multigrid Methods*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1987.

[29] D. W. Paeceman and H. H. Rachford Jr., The numerical solution of parabolic and elliptic equations, *J. Soc. Indust. Appl. Math.*, **3** (1955), 28–41.
[30] J. Papac, F. Gibou and C. Ratsch, Efficient symmetric discretization for the Poisson, heat and Stefan-type problems with Robin boundary conditions, *J. Comput. Phys.*, **229** (2010), 875–889.
[31] W. H. Press and S. A. Teukolsky, *Numerical Recipes in FORTRAN, the Art of Scientific Computing*, 2^{nd} edition, Cambridge University Press, New York, 1992.
[32] R. K. Sinha and B. Deka, Optimal error estimates for linear parabolic problems with discontinuous coefficients, *SIAM J. Numer. Anal.*, **43** (2005), 733–749.
[33] R. K. Sinha and B. Deka, Finite element methods for semilinear elliptic and parabolic interface problems, *Appl. Numer. Math.*, **59** (2009), 1870–1883.
[34] L. Song and S. Zhao, Symmetric interior penalty Galerkin approaches for two-dimensional parabolic interface problems with low regularity solutions, *J. Comput. Appl. Math.*, **330** (2018), 356–379.
[35] J. W. L. Wan and X.-D. Liu, A boundary condition-capturing multigrid approach to irregular boundary problems, *SIAM J. Sci. Comput.*, **25** (2004), 1982–2003.
[36] Z. Wei, C. Li and S. Zhao, A spatially second order alternating direction implicit (ADI) method for three dimensional parabolic interface problems, *Comput. Math. Appl.*, **75** (2018), 2173–2192.
[37] A. Wiegmann and K. P. Bube, The explicit-jump immersed interface method: Finite difference methods for PDEs with piecewise smooth solutions, *SIAM J. Numer. Anal.*, **37** (2000), 827–862.
[38] K. Xia, M. Zhan and G.-W. Wei, MIB method for elliptic equations with multi-material interfaces, *J. Comput. Phys.*, **230** (2011), 4588–4615.
[39] Q. Yang and X. Zhang, Discontinuous Galerkin immersed finite element methods for parabolic interface problems, *J. Comput. Appl. Math.*, **299** (2016), 127–139.
[40] S. Zhao, A matched alternating direction implicit (ADI) method for solving the heat equation with interfaces, *J. Sci. Comput.*, **63** (2015), 118–137.
[41] S. Zhao and G. W. Wei, High-order FDTD methods via derivative matching for Maxwell's equations with material interfaces, *J. Comput. Phys.*, **200** (2004), 60–103.
[42] Y. C. Zhou, S. Zhao, M. Feig and G. W. Wei, High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular source, *J. Comput. Phys.*, **213** (2006), 1–30.

Received November 2020; revised February 2021.

E-mail address: hfeng5@crimson.ua.edu
E-mail address: szhao@ua.edu