Computers and Mathematics with Applications 75 (2018) 2173–2192

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A spatially second order alternating direction implicit (ADI) method for solving three dimensional parabolic interface problems

Zhihan Wei [a], Chuan Li [b], Shan Zhao [a,*]

[a] Department of Mathematics, University of Alabama, Tuscaloosa, AL 35487, USA
[b] Department of Mathematics, West Chester University of Pennsylvania, West Chester, PA 19383, USA

ARTICLE INFO

Article history:
Available online 8 July 2017

Keywords:
Parabolic interface problem
Matched alternating direction implicit (ADI) method
Douglas–Rachford ADI scheme
Matched interface and boundary (MIB)

ABSTRACT

A new matched alternating direction implicit (ADI) method is proposed in this paper for solving three-dimensional (3D) parabolic interface problems with discontinuous jumps and complex interfaces. This scheme inherits the merits of its ancestor for two-dimensional problems, while possesses several novel features, such as a non-orthogonal local coordinate system for decoupling the jump conditions, two-side estimation of tangential derivatives at an interface point, and a new Douglas–Rachford ADI formulation that minimizes the number of perturbation terms, to attack more challenging 3D problems. In time discretization, this new ADI method is found to be of first order and stable in various experiments. In space discretization, the matched ADI method achieves the second order accuracy based on simple Cartesian grids for various irregularly-shaped surfaces and spatial–temporal dependent jumps. Computationally, the matched ADI method is as efficient as the fastest implicit scheme based on the geometrical multigrid for solving 3D parabolic equations, in the sense that its complexity in each time step scales linearly with respect to the spatial degree of freedom N, i.e., $O(N)$. Furthermore, unlike iterative methods, the ADI method is an exact or non-iterative algebraic solver which guarantees to stop after a certain number of computations for a fixed N. Therefore, the proposed matched ADI method provides a very promising tool for solving 3D parabolic interface problems.

1. Introduction

Parabolic interface problems are a large class of problems modeling the change of a physical or biological quantity across a material interface. Numerous prize-winning models, such as the Pennes model for magnetic fluid hyperthermia therapy of human cancers [1], the time-dependent Poisson–Boltzmann equation for simulating electrostatics on biomolecules immersed in water phase [2,3], and the cable equation for studying action potential propagating in active human cardiac tissues [4], can all be categorized to this class of interface problems. A typical parabolic interface problem involves a parabolic partial differential equation (PDE) for a function u

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha \nabla u) + f, \quad \text{in } \Omega \subset R^3, \tag{1}$$

with a proper boundary condition prescribed for u on the boundary $\partial\Omega$. The finite domain Ω is assumed to be split by a closed interface, $\Gamma = \Omega^+ \cap \Omega^-$, into two subdomains, $\Omega = \Omega^+ \cup \Omega^-$. Here $u(\vec{x}, t)$ is a 3D function of interest with

* Corresponding author. Fax: +1 205 348 7067.
E-mail address: szhao@ua.edu (S. Zhao).

http://dx.doi.org/10.1016/j.camwa.2017.06.037
0898-1221/

$\vec{x} = (x, y, z)$. The diffusion coefficient α is discontinuous across the interface Γ, and $f(\vec{x}, t)$ is the source which may be even singular across the interface. On the interface Γ, jump conditions

$$[u] := u^+ - u^- = \phi(\vec{x}, t) \quad \text{and} \quad [\alpha u_n] := \alpha^+ \nabla u^+ \cdot \vec{n} - \alpha^- \nabla u^- \cdot \vec{n} = \psi(\vec{x}, t), \tag{2}$$

relate the solution u on both sides of the interface Γ. In (2), $\vec{n}$ is the unit outer normal direction, and the superscripts, $-$ and $+$, denote the limiting values of a function from one side or the other of the interface. With the nonhomogeneous terms ϕ and ψ being spatially and temporally dependent, (2) takes a quite general form, while for many relatively simple applications, $\phi = \psi = 0$.

The exact solutions of (1)–(2) are not available, especially when the interface Γ is irregularly shaped, while the numerical solutions of parabolic interface problems are highly nontrivial in 3D, with two major bottlenecks being the accuracy and efficiency. In terms of accuracy, the standard numerical treatments are known to perform poorly for delivering accurate approximations, or even fail to converge, if the jump conditions (2) are not treated appropriately in the numerical formulation. This calls for specific treatments to incorporate the jump conditions into the numerical formulation. By using body-fitted grids, the enforcement of jump conditions has been studied in many finite element and finite volume methods [5–9]. By using a Cartesian grid, jump conditions are more difficult to be satisfied because the interface cuts the grid lines arbitrarily and the normal direction is usually not along the Cartesian directions. The immersed interface method (IIM) [10] is known to be one of the most successful Cartesian grid methods for general interface problems. It achieves the second order of accuracy by rigorous imposing jump conditions in finite difference discretization via local Taylor expansions. Several successful IIM methods have been developed for parabolic problems [11–15].

The efficiency is especially concerned for solving the parabolic PDE (1) in steady state simulations and applications with long time evolution. In such problems, an implicit time stepping is indispensable so that a larger time increment Δt or fewer time steps can be employed to save the CPU time. Nevertheless, a linear system has to be solved in every time step in an implicit integration. For 3D problems to be studied in this paper, the computational complexity in solving linear systems is more crucial than that of lower-dimensional problems, because a large spatial degree of freedom N is usually attained for a coarse mesh. By using a generic iterative solver, a typical complexity in solving the sparse systems of various interface schemes is of the order N^2, i.e., $O(N^2)$. The acceleration could be achieved with a proper preconditioner or a geometric multigrid method [16]. Nevertheless, due to special interface treatments, the formulation of the restriction and prolongation in a multigrid cycle is far away from trivial for interface problems. A major achievement in this area is the multigrid IIM method developed by Adams and Li [11], which scales as $O(N)$ computationally.

However, in all iterative methods, the iterations become inevitably longer for algebraic systems with extremely large sizes or very high condition numbers, due to the nature of iterative algorithms. For such challenging problems, an exact algebraic algorithm becomes more desirable, because the exact solver guarantees to stop with a fixed number of steps for a given N. The alternating direction implicit (ADI) methods [17–19], which solve a multi-dimensional parabolic PDE by converting a multidimensional system to multiple sets of independent one-dimensional (1D) systems, can be regarded as exact solvers from the algebra point of view. Furthermore, for Cartesian grid finite difference methods, the 1D ADI systems are tridiagonal and can be efficiently solved by using the Thomas algorithm [20]. As a consequence, the ADI finite difference method is as efficient as iterative multigrid methods with complexity $O(N)$, while being an exact algebraic solver. However, without an interface treatment, the central difference approximation is subject to an accuracy reduction in solving parabolic interface problems [2,3], while the multi-dimensional Cartesian grid interface treatments [11–15] may not be realized in a 1D manner.

This motivates the development of several interface schemes [21–26] which aim to maintain the efficiency and stability of the ADI methods, while restoring the accuracy to the second order near the interface. The first rigorous interface method is the IIM-ADI method introduced by Li and Mayo [22], in which the jump conditions (2) take a simpler form with $\alpha = 1$. The IIM-ADI method enforces the nonzero function and derivative jumps by introducing correction terms on the right hand sides of the 1D linear systems, without altering the central difference operators. Thus, the dimension decomposition can be simply conducted and the resulting IIM-ADI method has been applied to several problems [24,25]. For physical jump conditions involving material coefficient α inside the jumps, the construction of the IIM-ADI scheme through correction terms is intractable [23].

To treat the general jump conditions (2) with α being a piecewisely-defined constant, a completely different strategy has been introduced in the matched ADI method for solving two-dimensional (2D) heat equations [21,26], in which a tensor product decomposition of 2D jump conditions is carried out to generate 1D jump conditions along x and y directions. These 1D jump conditions can then be enforced via the matched interface and boundary (MIB) scheme [27–30] in the ADI framework, and a fast algebraic algorithm has been developed to solve the resulted perturbed tridiagonal linear system with a complexity of $O(N)$. Both Douglas [26] and Peaceman–Rachford [21] ADI schemes have been studied, and a spatial second order of accuracy is numerically confirmed.

The goal of this work is to develop a matched ADI method for solving 3D parabolic interface problems. However, such a development faces many new challenges, which require novel numerical schemes for handling much more complicated interfaces and jump conditions. In particular, generalizing the tensor product decomposition of jump conditions from 2D to 3D is highly non-trivial, because the interface is now a surface, which is geometrically more complicated. Moreover, in the existing matched ADI schemes [21,26], the MIB corrections depend on function values at different time levels so that the spatial approximation is affected by the temporal error. When generating to 3D, an ADI design with less perturbation errors becomes a pressing issue in the presence of such spatial–temporal error interference. To overcome these difficulties,

several major improvement will be conducted in this paper in developing a 3D matched ADI method: (i) In contrast to the 2D schemes [21,26], one grid line is allowed to intersect the interface more than twice in the proposed 3D scheme. This effort is obviously necessary for handling complex interfaces with fast changing curvature. (ii) The proposed 3D method estimates the tangential derivative from both sides of the interface, based on the local curvature of the interface, so that the numerical estimation accuracy is well maintained. (iii) More importantly, a non-orthogonal local coordinate system is proposed to decompose the 3D interface jump conditions, so that all tangential derivatives can be approximated in a 2D manner. If a local orthogonal coordinate is employed as in the usual 3D MIB scheme [27], tangential derivative approximations are of 3D nature and become far more complicated. (vi) Finally, appropriate treatments are employed to reduce the computational cost and to maintain the convergence rate and stability.

The rest of this work is organized as follows. Section 2 is devoted to new numerical treatments of the proposed 3D matched ADI method. The convergence test of the proposed method will be examined numerically in Section 3, which is followed by the conclusions and future developments presented in Section 4.

2. Theory and algorithm

Consider an arbitrarily shaped dielectric interface Γ enclosed within a cubic domain $\Omega \subset \mathbb{R}^3$. The interface Γ is analytically determined as the zero level set of a given function $S(x, y, z)$, i.e., $S(x, y, z) = 0$. We assume a uniform grid partition with mesh size, $\Delta x = h_x$, $\Delta y = h_y$ and $\Delta z = h_z$, resulting in N_x, N_y, and N_z grids in x, y, and z directions, respectively. Being aware of the complex shape of the 3D interface Γ, the grid line is allowed to cross the interface Γ more than twice. Moreover, the temporal domain is partitioned with uniform time increment Δt, and the notation $u^m_{i,j,k} = u(x_i, y_j, z_k, t_m)$ is utilized to denote the value of unknown function on grid (x_i, y_j, z_k) at time step t_m.

2.1. ADI semi-discretization

A Douglas–Rachford ADI method [17–19] will be constructed for solving the parabolic PDE (1). For a piecewise constant α, we first rewrite (1) as

$$\frac{1}{\alpha}\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x^2} + \frac{\partial^2 u}{\partial^2 y^2} + \frac{\partial^2 u}{\partial^2 z^2} + \frac{f}{\alpha}, \quad \text{in } \Omega^- \cup \Omega^+, \tag{3}$$

which allows an easier numerical formulation for temporal discretization, while being equivalent to (1), subject to the same jump conditions (2). Applying the first order Douglas–Rachford ADI method on Eq. (3) yields

$$\begin{aligned}
\left(\frac{1}{\alpha} - \Delta t\frac{\partial^2}{\partial x^2}\right) u^*_{i,j,k} &= \left(\frac{1}{\alpha} + \Delta t\left(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\right) u^m_{i,j,k} + \frac{\Delta t}{\alpha} f^{m+1}_{i,j,k},\\
\left(\frac{1}{\alpha} - \Delta t\frac{\partial^2}{\partial y^2}\right) u^{**}_{i,j,k} &= \frac{1}{\alpha} u^*_{i,j,k} - \Delta t\frac{\partial^2}{\partial y^2} u^m_{i,j,k},\\
\left(\frac{1}{\alpha} - \Delta t\frac{\partial^2}{\partial z^2}\right) u^{m+1}_{i,j,k} &= \frac{1}{\alpha} u^{**}_{i,j,k} - \Delta t\frac{\partial^2}{\partial z^2} u^m_{i,j,k},
\end{aligned} \tag{4}$$

where u^* and u^{**} are two intermediate values. Noticing that the ADI formula (4) is derived from the implicit Euler method

$$\frac{u^{m+1}_{i,j,k} - u^m_{i,j,k}}{\alpha\Delta t} = \frac{\partial^2}{\partial x^2} u^{m+1}_{i,j,k} + \frac{\partial^2}{\partial y^2} u^{m+1}_{i,j,k} + \frac{\partial^2}{\partial z^2} u^{m+1}_{i,j,k} + \frac{f^{m+1}_{i,j,k}}{\alpha}. \tag{5}$$

To see this, one can eliminate u^* and u^{**} in (4), which gives rise to

$$\begin{aligned}
\left(\frac{1}{\alpha} - \Delta t\frac{\partial^2}{\partial x^2}\right)\left(\frac{1}{\alpha} - \Delta t\frac{\partial^2}{\partial y^2}\right)\left(\frac{1}{\alpha} - \Delta t\frac{\partial^2}{\partial z^2}\right) u^{m+1}_{i,j,k} &= \frac{1}{\alpha^3} u^m_{i,j,k} - \Delta t^3\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2}\frac{\partial^2}{\partial z^2} u^m_{i,j,k}\\
&+ \frac{\Delta t^2}{\alpha}\left(\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial y^2}\frac{\partial^2}{\partial z^2}\right) u^m_{i,j,k} + \frac{\Delta t}{\alpha^3} f^{m+1}_{i,j,k}.
\end{aligned} \tag{6}$$

Reorganizing the terms in (6), we have

$$\begin{aligned}
u^{m+1}_{i,j,k} - \alpha\Delta t\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) u^{m+1}_{i,j,k} &= u^m_{i,j,k} + \Delta t f^{m+1}_{i,j,k}\\
- \alpha^2\Delta t^2\left(\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial y^2}\frac{\partial^2}{\partial z^2}\right)(u^{m+1}_{i,j,k} - u^m_{i,j,k}) &+ \alpha^3\Delta t^3\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2}\frac{\partial^2}{\partial z^2}(u^{m+1}_{i,j,k} - u^m_{i,j,k}).
\end{aligned} \tag{7}$$

Since the difference $(u^{m+1}_{i,j,k} - u^m_{i,j,k})$ is on the order of Δt, last two terms on the right hand side of (7) are higher order perturbation terms. After dropping them, (7) is simply (5). Thus, the temporal order of the Douglas–Rachford ADI method (4) is one, which is the same as the implicit Euler scheme (5). Computationally, the Douglas–Rachford ADI scheme is superior to the implicit Euler scheme for the efficiency consideration, because one only needs to solve 1D problems in (4), which leads to a faster and easier-to-implement numerical scheme.

2.2. Spatial discretization of the 3d matched ADI method

The spatial derivatives in (4) or (5) will be approximated by the finite difference operators δ_{xx}, δ_{yy} and δ_{zz}, which must be treated with great caution due to the presence of the interface Γ. Standard central difference formula

$$\begin{aligned}
\frac{\partial^2}{\partial x^2}u^{m+1}_{i,j,k} &\approx \delta_{xx}u^{m+1}_{i,j,k} := \frac{1}{\Delta x^2}(u^{m+1}_{i-1,j,k} - 2u^{m+1}_{i,j,k} + u^{m+1}_{i+1,j,k}),\\
\frac{\partial^2}{\partial y^2}u^{m+1}_{i,j,k} &\approx \delta_{yy}u^{m+1}_{i,j,k} := \frac{1}{\Delta y^2}(u^{m+1}_{i,j-1,k} - 2u^{m+1}_{i,j,k} + u^{m+1}_{i,j+1,k}),\\
\frac{\partial^2}{\partial z^2}u^{m+1}_{i,j,k} &\approx \delta_{zz}u^{m+1}_{i,j,k} := \frac{1}{\Delta z^2}(u^{m+1}_{i,j,k-1} - 2u^{m+1}_{i,j,k} + u^{m+1}_{i,j,k+1}),
\end{aligned} \tag{8}$$

can be adopted at the nodes which are far away from Γ (they are called regular nodes), while finite difference formula (8) must be modified in order to rigorously impose the jump conditions (2) into the numerical formulation at the nodes close to the interface Γ (they are called irregular nodes). The matched ADI method corrects δ_{xx}, δ_{yy} and δ_{zz} in two major stages [21,26]. First, a tensor product decomposition is conducted to transfer 3D jump conditions (2) into 1D ones along Cartesian directions, i.e., $[\alpha u_x] = \psi_x$, $[\alpha u_y] = \psi_y$, and $[\alpha u_z] = \psi_z$, where we denote $u_x = \frac{\partial u}{\partial x}$. Second, one enforces the 1D jump conditions in each alternating direction of the ADI algorithm via the matched interface and boundary (MIB) scheme [27–30].

2.2.1. Local coordinate and non-orthogonal tangential directions

A tensor product decomposition of jump conditions will be conducted at every necessary interface point. Denote $(x_\Gamma, y_\Gamma, z_\Gamma)$ a general intersection point formed by the interface Γ and one Cartesian grid line. Note that one Cartesian grid line has two fixed coordinate values. For instance, consider a y grid line with $x = x_i$ and $z = z_k$, where x_i and z_k are multiples of Δx and Δz. The interface point is then (x_i, y_Γ, z_k), where y_Γ is assumed to be an arbitrary value satisfying $y_{j-1} \le y_\Gamma \le y_j$. At this particular interface point, we need to generate a new 1D jump condition $[\alpha u_y] = \psi_y$. Similarly, at every interface point $(x_\Gamma, y_\Gamma, z_\Gamma)$, one needs to construct a new 1D jump condition. This procedure is demonstrated in y-direction, and the treatments in the other two directions are similar.

A local coordinate is introduced first. At the interface point IPY(x_i, y_Γ, z_k), the outward normal direction $\vec{n} = (n_x, n_y, n_z)$ can be determined by the level set function $S(x, y, z)$. Denote a local coordinate (ξ, η, ζ) with ξ varying along $\vec{n}$ and η and ζ being two tangential directions. Non-orthogonal tangential directions will be constructed in the local coordinate so that approximations to the tangential derivatives can be conducted in a 2D Cartesian grid plane, instead of a 3D space, in the tensor product decomposition. This new idea considerably simplifies the proposed algorithm.

The tangential plane through the interface point IPY(x_i, y_Γ, z_k) can be represented as

$$n_x(x - x_i) + n_y(y - y_\Gamma) + n_z(z - z_k) = 0. \tag{9}$$

We formulate η and ζ by intersecting this tangential plane with the place $x = x_i$ and $z = z_k$, respectively, (see Fig. 1)

$$n_y y + n_z z = n_y y_\Gamma + n_z z_k, \quad \text{and} \quad x = x_i, \tag{10}$$

$$n_x x + n_y y = n_x x_i + n_y y_\Gamma, \quad \text{and} \quad z = z_k. \tag{11}$$

After choosing a positive direction appropriately, we can define two unit directions on these two intersection lines by

$$\eta = \left(0, \frac{n_z}{\sqrt{n_y^2 + n_z^2}}, -\frac{n_y}{\sqrt{n_y^2 + n_z^2}}\right) \quad \text{and} \quad \zeta = \left(\frac{n_y}{\sqrt{n_x^2 + n_y^2}}, -\frac{n_x}{\sqrt{n_x^2 + n_y^2}}, 0\right),$$

as shown in Fig. 1. The local coordinate system (ξ, η, ζ) at the interface point IPY can be related to the global xyz coordinate system by a transformation matrix P

$$\begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = P \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{12}$$

In the y-direction, the transformation matrix P_y is provided by

$$P_y = \begin{bmatrix} n_x & n_y & n_z \\ 0 & \dfrac{n_z}{\sqrt{n_y^2 + n_z^2}} & -\dfrac{n_y}{\sqrt{n_y^2 + n_z^2}} \\ \dfrac{n_y}{\sqrt{n_x^2 + n_y^2}} & -\dfrac{n_x}{\sqrt{n_x^2 + n_y^2}} & 0 \end{bmatrix}. \tag{13}$$

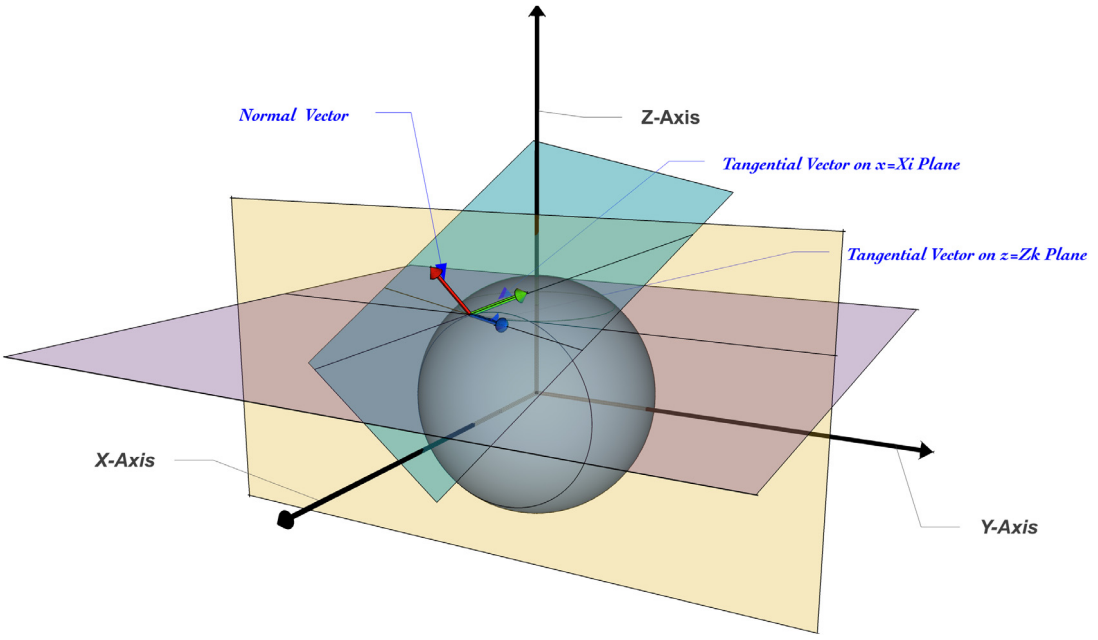

Fig. 1. Non-orthogonal tangential directions η and ζ. Here at an interface point with the normal direction ξ (red), η and ζ are determined by intersecting the tangential plane (turquoise) with the place $x = x_i$ (yellow) and $z = z_k$ (mauve), respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Similarly, one can construct the transformation matrix P_x in the x-direction

$$P_x = \begin{bmatrix} n_x & n_y & n_z \\ \dfrac{n_z}{\sqrt{n_x^2+n_z^2}} & 0 & -\dfrac{n_x}{\sqrt{n_x^2+n_z^2}} \\ \dfrac{n_y}{\sqrt{n_x^2+n_y^2}} & -\dfrac{n_x}{\sqrt{n_x^2+n_y^2}} & 0 \end{bmatrix}, \tag{14}$$

and P_z in the z-direction

$$P_z = \begin{bmatrix} n_x & n_y & n_z \\ 0 & \dfrac{n_z}{\sqrt{n_y^2+n_z^2}} & -\dfrac{n_y}{\sqrt{n_y^2+n_z^2}} \\ \dfrac{n_z}{\sqrt{n_x^2+n_z^2}} & 0 & -\dfrac{n_x}{\sqrt{n_x^2+n_z^2}} \end{bmatrix}, \tag{15}$$

respectively, for interface point (x_I, y_j, z_k) and (x_i, y_j, z_I).

2.2.2. Tensor product decomposition of jump conditions

After the local $\xi\eta\zeta$-coordinate system at the interface point IPY is obtained, one can differentiate the zeroth order jump condition $[u] = \phi$ in (2) along the two tangential directions (η and ζ) to obtain two additional jump conditions

$$[u_\eta] = \phi_\eta, \quad \text{and} \quad [u_\zeta] = \phi_\zeta. \tag{16}$$

Note that the known jump conditions (2) and (16) are in normal and tangential directions, and cannot be applied in a 1D manner in the ADI algorithm, because they are not along the x-, y- or z-axis.

The desired Cartesian jump conditions, i.e., $[\alpha u_x]$, $[\alpha u_y]$ and $[\alpha u_z]$, can be obtained through coordinate transformation,

$$\begin{pmatrix} \psi_x \\ \psi_y \\ \psi_z \end{pmatrix} := \begin{pmatrix} [\alpha u_x] \\ [\alpha u_y] \\ [\alpha u_z] \end{pmatrix} = P^{-1} \begin{pmatrix} [\alpha u_\xi] \\ [\alpha u_\eta] \\ [\alpha u_\zeta] \end{pmatrix}, \tag{17}$$

where $[\alpha u_\xi] = \psi$ is known in (2), while $[\alpha u_\eta]$ and $[\alpha u_\zeta]$ are unknown. Notice that the transformation matrix P is singular only when the tangential plane is parallel to one of xy-, xz-, or yz- plane. In such a case, the flux jump condition $[\alpha u_\xi] = \psi$

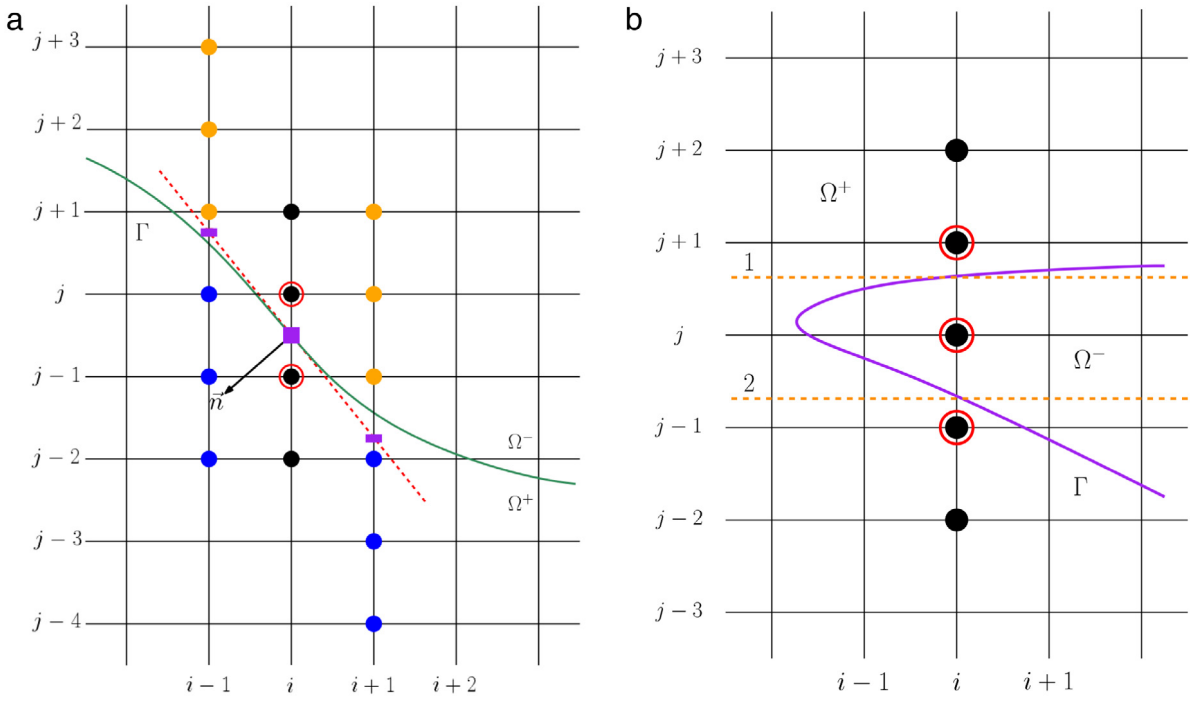

Fig. 2. Tangential derivative approximation and MIB interface treatment. (a) The tangential derivative at the interface point (purple square) is approximated through two auxiliary points (purple rectangles). Each auxiliary value will be extrapolated by three supporting nodes from the same side (positive or negative). An optimization procedure is designed to automatically select either positive or negative side for approximating tangential derivatives. The MIB 1D jump condition enforcement is also illustrated, which involves four nodes (black filled circles) and two fictitious nodes (red open circles). (b) In a corner MIB treatment of 1D jump conditions, five nodes (black filled circles) and three fictitious nodes (red open circles) are involved. Here two interface points $(x_i, y_{\Gamma 1}, z_k)$ and $(x_i, y_{\Gamma 2}, z_k)$ are the intersection points of Γ with two dashed lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is simply the desired 1D jump condition in z, y or x direction. Besides this degenerate case, P^{-1} in (17) always exists and can be calculated either analytically or numerically.

The unknown jumps of $[\alpha u_\eta]$ and $[\alpha u_\zeta]$ can be calculated with the aid of (16)

$$[\alpha u_\eta] = \alpha^+ \phi_\eta + (\alpha^+ - \alpha^-) u_\eta^- \quad \text{or} \quad [\alpha u_\eta] = \alpha^- \phi_\eta + (\alpha^+ - \alpha^-) u_\eta^+, \tag{18}$$

$$[\alpha u_\zeta] = \alpha^+ \phi_\zeta + (\alpha^+ - \alpha^-) u_\zeta^- \quad \text{or} \quad [\alpha u_\zeta] = \alpha^- \phi_\zeta + (\alpha^+ - \alpha^-) u_\zeta^+. \tag{19}$$

Computationally, we will approximate two tangential derivatives u_η and u_ζ, in either $+$ or $-$ side, for determining $[\alpha u_\eta]$ and $[\alpha u_\zeta]$ numerically. Once $[\alpha u_\eta]$ and $[\alpha u_\zeta]$ are obtained, they are coupled with $[\alpha u_\xi] = \psi$ to deliver 1D jump conditions in Cartesian directions: $[\alpha u_x] = \psi_x$, $[\alpha u_y] = \psi_y$, and $[\alpha u_z] = \psi_z$. Thus, the key in the tensor product decomposition of 3D jump conditions is how to approximate two tangential derivatives accurately.

2.2.3. Tangential derivative approximations

A new algorithm is proposed in this paper to approximate tangential derivatives, which is novel in two folds. First, since each of the non-orthogonal tangential directions is contained in a 2D Cartesian grid plane, the tangential derivatives can be computed in a 2D manner. If the orthogonal tangential directions are adopted as in the classical MIB 3D scheme [27], one tangential derivative has to be calculated as a 3D problem. Obviously, the present algorithm is simpler. Second, information from both positive and negative sides will be employed for calculating tangential derivatives, while in our previous 2D matched ADI methods [21,26], the tangential derivatives are calculated only from the positive side. In 3D, the interface Γ becomes geometrically more complicated, so that a more robust algorithm has to be developed in this work.

We illustrate the proposed algorithm by considering u_ζ at the interface point (x_i, y_Γ, z_k), and u_η can be treated similarly. Since the ζ direction is contained in a plane $z = z_k$, all involved points are in this plane. Thus, the numerical scheme can be shown in a x–y coordinate, see Fig. 2(a). Here we assume $y_{j-1} \le y_\Gamma \le y_j$ and denote the interface point (and other points) simply as (x_i, y_Γ) by dropping $z = z_k$. If Γ is locally convex with respect to Ω^-, the approximation of u_ζ^+ by nearby on-grid values from the outer domain Ω^+ produces a better precision, while for a local concave curve, it is better to approximate u_ζ^- from Ω^-. For a situation shown in Fig. 2(a) where the interface Γ has an inflection, an optimization procedure is designed to select a better approximation from either $+$ or $-$ side. In fact, we apply the same optimization procedure for all interface points so that one can automatically select u_ζ^+ or u_ζ^- based on the local geometry.

In our algorithm, the auxiliary and supporting nodes for approximating u_ζ^+ and u_ζ^- are first determined. As shown in Fig. 2(a), the interface point (x_i, y_Γ) is marked with a purple square. Two auxiliary points $(x_{i-1}, y_{\Gamma-1})$ and $(x_{i+1}, y_{\Gamma+1})$ (purple rectangles in Fig. 2(a)) along the ζ line are chosen to form a central difference approximation for the u_ζ

$$u_\zeta^\pm(x_i, y_\Gamma) \approx \frac{u^\pm(x_{i+1}, y_{\Gamma+1}) - u^\pm(x_{i-1}, y_{\Gamma-1})}{2\Delta\zeta}, \tag{20}$$

where $\Delta\zeta$ is the distance between (x_i, y_Γ) and $(x_{i-1}, y_{\Gamma-1})$. The auxiliary values will be interpolated/extrapolated by using supporting nodal values from the same side. Referring to Fig. 2(a), $u^+(x_{i+1}, y_{\Gamma+1})$ and $u^+(x_{i-1}, y_{\Gamma-1})$ will be extrapolated by six supporting nodes below Γ (blue circles), while $u^-(x_{i+1}, y_{\Gamma+1})$ and $u^-(x_{i-1}, y_{\Gamma-1})$ are approximated by six supporting nodes above Γ (orange circles). The selection of u_ζ^+ or u_ζ^- is according to the total distance between supporting nodes and auxiliary nodes. For Fig. 2(a), we define

$$d^+ = \sum_{J=2}^{4} |y_{\Gamma+1} - y_{j-J}| + \sum_{J=0}^{2} |y_{\Gamma-1} - y_{j-J}|, \tag{21}$$

$$d^- = \sum_{J=-1}^{1} |y_{\Gamma+1} - y_{j+J}| + \sum_{J=1}^{3} |y_{\Gamma-1} - y_{j+J}|. \tag{22}$$

If $d^+ > d^-$, we calculate u_ζ^- by using (20) from the negative side. Otherwise, we approximate u_ζ^+ by using (20) from the positive side. Then the auxiliary values are extrapolated by six supporting values.

The time interference is a subtle issue in this process. Note that at this stage of approximation, function values at the future time $t = t_{m+1}$ are unknown. So, we can only use the known function values at the current time step $t = t_m$ for the six supporting nodes. Consequently, the calculated tangential derivatives are held at t_m, and we denote them as $(u_\zeta^-)^m$ and $(u_\zeta^+)^m$. However, the finite difference correction (8) is intended to be conducted at t_{m+1}. Hence, in (17), (18), and (19), we specify analytically known jumps at t_{m+1}, i.e., ψ^{m+1}, ϕ_η^{m+1}, and ϕ_ζ^{m+1}, for a better accuracy. Mathematically, the calculated jumps ψ_x, ψ_y, and ψ_z can be expressed as a linear combination of some supporting nodal values at t_m, and ψ^{m+1}, ϕ_η^{m+1}, and ϕ_ζ^{m+1} at t_{m+1}. With hybrid contributions from two time levels, the Cartesian jump values ψ_x, ψ_y, and ϕ_z mainly depend on the supporting function values at t_m. Thus, we will denote them as ψ_x^m, ψ_y^m, and ψ_z^m. This finishes the first stage of the spatial discretization of the matched ADI method.

2.2.4. 1D jump conditions enforcement

In the second stage of the matched ADI method, we impose the decomposed 1D jump conditions

$$[u] = \phi^{m+1}, \qquad [\alpha u_x] = \psi_x^m, \qquad [\alpha u_y] = \psi_y^m, \qquad [\alpha u_z] = \psi_z^m, \tag{23}$$

to correct the finite difference operators δ_{xx}, δ_{yy} and δ_{zz} in (8) by using the MIB scheme [27–30]. We will also only consider δ_{yy} here for simplicity. For two irregular points, δ_{yy} at time step t_{m+1} is modified to be

$$\begin{aligned} \delta_{yy} u_{i,j,k}^{m+1} &:= \frac{1}{\Delta y^2}(\tilde{u}_{i,j-1,k}^{m+1} - 2u_{i,j,k}^{m+1} + u_{i,j+1,k}^{m+1}), \quad \text{in } \Omega^- \\ \delta_{yy} u_{i,j-1,k}^{m+1} &:= \frac{1}{\Delta y^2}(u_{i,j-2,k}^{m+1} - 2u_{i,j-1,k}^{m+1} + \tilde{u}_{i,j,k}^{m+1}), \quad \text{in } \Omega^+ \end{aligned} \tag{24}$$

where $\tilde{u}_{i,j-1,k}^{m+1}$ and $\tilde{u}_{i,j,k}^{m+1}$ are two fictitious values at nodes (x_i, y_{j-1}, z_k) and (x_i, y_j, z_k), respectively, as shown in Fig. 2(a). These two fictitious values $\tilde{u}_{i,j-1,k}^{m+1}$ and $\tilde{u}_{i,j,k}^{m+1}$ are determined by discretizing two jump conditions in (23).

$$\omega_{0,1}^+ u_{i,j-2,k}^{m+1} + \omega_{0,2}^+ u_{i,j-1,k}^{m+1} + \omega_{0,3}^+ \tilde{u}_{i,j,k}^{m+1} = \omega_{0,1}^- \tilde{u}_{i,j-1,k}^{m+1} + \omega_{0,2}^- u_{i,j,k}^{m+1} + \omega_{0,3}^- u_{i,j+1,k}^{m+1} + \phi^{m+1}, \tag{25}$$

$$\alpha^+(\omega_{1,1}^+ u_{i,j-2,k}^{m+1} + \omega_{1,2}^+ u_{i,j-1,k}^{m+1} + \omega_{1,3}^+ \tilde{u}_{i,j,k}^{m+1}) = \alpha^-(\omega_{1,1}^- \tilde{u}_{i,j-1,k}^{m+1} + \omega_{1,2}^- u_{i,j,k}^{m+1} + \omega_{1,3}^- u_{i,j+1,k}^{m+1}) + \psi_y^m, \tag{26}$$

where $\omega_{i,j}^+$ and $\omega_{i,j}^-$ are finite different weights [31] for interface surrounding nodes from outer domain Ω^+ or inner domain Ω^-. The subscript i represents the interpolation when $i = 0$ and first order approximation when $i = 1$, and the subscript j represents the indices of related grid nodes. Solving (25) and (26) allows $\tilde{u}_{i,j-1,k}^{m+1}$ and $\tilde{u}_{i,j,k}^{m+1}$ to be written as linear combinations of $u_{i,j-2,k}^{m+1}$, $u_{i,j-1,k}^{m+1}$, $u_{i,j,k}^{m+1}$, $u_{i,j+1,k}^{m+1}$, and two jump values ϕ^{m+1} and ψ_y^m. By substituting $\tilde{u}_{i,j-1,k}^{m+1}$ and $\tilde{u}_{i,j,k}^{m+1}$ into (24), $\delta_{yy} u_{i,j,k}^{m+1}$ and $\delta_{yy} u_{i,j-1,k}^{m+1}$ are represented by four surrounding on-grid values at t_{m+1}, i.e., $u_{i,j-2,k}^{m+1}$, $u_{i,j-1,k}^{m+1}$, $u_{i,j,k}^{m+1}$, $u_{i,j+1,k}^{m+1}$, four jump values at t_{m+1}, i.e., ϕ^{m+1}, ψ^{m+1}, ϕ_η^{m+1}, and ϕ_ζ^{m+1}, six values at nodes on plane $z = z_k$ at t_m, and six values at nodes on plane $x = x_i$ at t_m.

It is possible that the interface Γ changes rapidly and cuts one grid line twice within a short distance. If there are at least two grid nodes in between these two intersection points, the above mentioned approximation can be conducted. When there is only one node (it is called a corner point), a MIB corner treatment has to be adopted. For instance, in Fig. 2(b), a corner treatment is needed for (x_{i-1}, y_j, z_k) and (x_i, y_j, z_k) along y direction, while the usual MIB scheme can be used for

other irregular points in Ω^-. Since the present jump condition enforcement is of 1D nature, the corner treatment developed in our previous 2D matched ADI method [26] can be simply utilized for 3D problems. Essentially, we need to enforce jump conditions at two interface points simultaneously. We omit the details but briefly describe the idea here by considering (x_i, y_j, z_k) in Fig. 2(b). At this corner point, three fictitious values, $\tilde{u}^{m+1}_{i,j-1,k}$, $\tilde{u}^{m+1}_{i,j,k}$ and $\tilde{u}^{m+1}_{i,j+1,k}$, are calculated together with one additional fictitious value, either $\tilde{u}^{m+1}_{i,j-2,k}$ or $\tilde{u}^{m+1}_{i,j+2,k}$ depending on which node is closer, in order to maintain the convergence rate. The four fictitious values are solved by a system of four equations, built from imposing the jump conditions at the interface points on both sides of the corner point, following similar formulation as in (25)–(26). By substituting fictitious values into (24), $\delta_{yy}u^{m+1}_{i,j,k}$ is represented by five surrounding on-grid values at t_{m+1}, i.e., $u^{m+1}_{i,j-2,k}$, $u^{m+1}_{i,j-1,k}$, $u^{m+1}_{i,j,k}$, $u^{m+1}_{i,j+1,k}$ and $u^{m+1}_{i,j+2,k}$, eight jump values at t_{m+1}, i.e., ϕ^{m+1}, ψ^{m+1}, ϕ^{m+1}_η, and ϕ^{m+1}_ζ at two interface points $(x_i, y_{\Gamma 1}, z_k)$ and $(x_i, y_{\Gamma 2}, z_k)$, and a total of 24 supporting nodal values at t_m. This finishes the second stage of the spatial discretization of the matched ADI method. In this entire process, a spatially second order of accuracy is guaranteed throughout the domain.

Symbolically, the aforementioned MIB corrections of the finite difference operators δ_{xx}, δ_{yy}, and δ_{zz} can be expressed as [21,26]

$$\delta_{xx}u^{m+1}_{i,j,k} = D_{xx}u^{m+1}_{i,j,k} + B_x u^m_{i,j,k} + \Phi^{m+1}_x, \tag{27}$$

$$\delta_{yy}u^{m+1}_{i,j,k} = D_{yy}u^{m+1}_{i,j,k} + B_y u^m_{i,j,k} + \Phi^{m+1}_y, \tag{28}$$

$$\delta_{zz}u^{m+1}_{i,j,k} = D_{zz}u^{m+1}_{i,j,k} + B_z u^m_{i,j,k} + \Phi^{m+1}_z, \tag{29}$$

where D_{xx}, D_{yy}, and D_{zz} are perturbed operators which have a band-width 4 for a usual irregular node, and a band-width 5 for a corner node. The operators B_x, B_y, and B_z are due to tangential derivative approximations, which have a band-width 12 for a usual irregular node, and a band-width 24 for a corner node. The jump terms Φ^{m+1}_x, Φ^{m+1}_y, and Φ^{m+1}_z are the sums of 4 and 8 jump values, respectively, for a usual irregular node and corner node.

2.3. A new ADI method with less perturbation terms

Because the MIB corrected finite difference operators in (27)–(29) involve two time levels t_m and t_{m+1}, a direct application of them to approximate derivatives in the standard ADI scheme (4) is questionable. In particular, the intermediate functions u^* and u^{**} are at unknown fractional time instants between t_m and t_{m+1}, so that one does not know how to specify a time for them. One possible choice here is to use t_{m+1} for approximating derivatives of u^* and u^{**} in (4), while derivatives of u^m on the right hand sides of (4) are evaluated at t_m. As a consequence, when one eliminates the intermediate functions u^* and u^{**}, similar to (6) and (7) in the semi-discretized form, the difference between such an ADI scheme and the implicit Euler scheme involves many perturbation terms.

To overcome the aforementioned difficulty, we propose a new ADI method which minimizes the number perturbation terms when comparing it with the implicit Euler scheme. Our idea is to apply the finite difference operators (27)–(29) to the implicit Euler scheme (5), which then can be rewritten as

$$\frac{u^{m+1}_{i,j,k} - u^m_{i,j,k}}{\alpha \Delta t} = D_{xx}u^{m+1}_{i,j,k} + D_{yy}u^{m+1}_{i,j,k} + D_{zz}u^{m+1}_{i,j,k} + F^{m+1}_{i,j,k}, \tag{30}$$

where $F^{m+1}_{i,j,k} = (B_x + B_y + B_z)u^m_{i,j,k} + \Phi^{m+1}_x + \Phi^{m+1}_y + \Phi^{m+1}_z + \frac{1}{\alpha}f^{m+1}_{i,j,k}$. The Douglas–Rachford ADI method for the implicit Euler scheme (30) is simply given as

$$\begin{aligned}
\left(\frac{1}{\alpha} - \Delta t D_{xx}\right) u^*_{i,j,k} &= \left(\frac{1}{\alpha} + \Delta t(D_{yy} + D_{zz})\right) u^m_{i,j,k} + \Delta t F^{m+1}_{i,j,k}, \\
\left(\frac{1}{\alpha} - \Delta t D_{yy}\right) u^{**}_{i,j,k} &= \frac{1}{\alpha}u^*_{i,j,k} - \Delta t D_{yy}u^m_{i,j,k}, \\
\left(\frac{1}{\alpha} - \Delta t D_{zz}\right) u^{m+1}_{i,j,k} &= \frac{1}{\alpha}u^{**}_{i,j,k} - \Delta t D_{zz}u^m_{i,j,k}.
\end{aligned} \tag{31}$$

Similarly, we can show the difference between such an ADI and the implicit Euler schemes is

$$-\alpha^2 \Delta t^2 (D_{xx}D_{yy} + D_{xx}D_{zz} + D_{yy}D_{zz})(u^{m+1}_{i,j,k} - u^m_{i,j,k}) + \alpha^3 \Delta t^3 D_{xx}D_{yy}D_{zz}(u^{m+1}_{i,j,k} - u^m_{i,j,k}), \tag{32}$$

which is in consistent with the semi-discretized form. The temporal order of the proposed ADI scheme is obviously one, i.e., $O(\Delta t)$.

In our computations, the boundary conditions for u^* and u^{**} in (31) are assumed to be the same as u^{m+1}, i.e., at time step $t = t_{m+1}$. We note that the matrix coefficients of D_{xx}, D_{yy}, D_{zz}, B_x, B_y, and B_z only require to be calculated once before the time evolution, because these spatial discretization coefficients only depend on interface geometry and grid nodes, which are time invariant. Then, these sparse coefficients and their column and row indices are stored and reused in each time step, without too much computational overhead. Algebraically, the corrected operators D_{xx}, D_{yy}, and D_{zz} are perturbed tridiagonal matrices with only a few rows having a band-width four (usual irregular node) or five (corner node). A fast algorithm has

been developed in [26] to invert such matrices efficiently at each time step, in which these systems are first converted to tridiagonal ones by row-eliminations and solved by the Thomas algorithm [20]. Taking the advantage of the Thomas algorithm, the proposed 3D matched ADI method has a computational cost of order $O(N)$ for each time step, where N is the total spatial degree of freedom.

Neglecting the perturbation terms (32), the stability of the ADI scheme (31) is essentially governed by that of the implicit Euler scheme (30). Denote $\mathbf{U}^m$ a vector of dimension N containing all u values at the time t_m. One can rewrite the implicit Euler scheme (30) into the matrix form

$$\mathbf{D}\mathbf{U}^{m+1} = \mathbf{B}\mathbf{U}^m + \mathbf{C}, \tag{33}$$

where $\mathbf{D}$ and $\mathbf{B}$ are matrices of dimension $N \times N$, and $\mathbf{C}$ is a vector of dimension N. Note that $\mathbf{B}$ is due to the approximation of tangential derivatives by some function values of u at time t_m. The sparse elements of $\mathbf{B}$ are distributed in a rather random fashion—their locations depend on the interface geometry and grid size. Generally speaking, a von Neumann type stability analysis is infeasible. Alternatively, the stability analysis of the proposed ADI scheme can be conducted as in [26], through numerically verifying whether the spectral radius of the magnifying matrix $\mathbf{D}^{-1}\mathbf{B}$ is not greater than one. However, since N is a very large number for 3D problems, the eigenvalue computation is time consuming, and is thus skipped in the present study. Instead, the stability of the 3D matched ADI scheme is numerically examined in next section.

3. Numerical experiments

In this section, we examine both temporal and spatial convergence rates of the proposed 3D matched ADI algorithm with various interface shapes and jump conditions. The code is written in C++ with the aid of objected oriented programming. The data structure "vector" in C++ STL is selected to perform the linked-list feature. With such a new data structure, the grid line can cross the interface many times without restriction, while in our previous 2D matched ADI methods [21,26], it allowed to cross at most twice. The present data structure enables us to treat more complicated geometry in 3D. Reported CPU time is recorded on the master node of SGI Ultraviolet in Alabama Supercomputer Center (https://www.asc.edu/hpc/services/hpc-services) with a single Xeon E5-4640 CPU core operating at 2.4 GHz clock speed.

In all examples, a piecewisely defined analytical solution $u(x, y, z)$ is constructed so that both the initial condition at time $t = 0$ and Dirichlet boundary conditions on the boundary of a cubic computational domain Ω can be obtained from the constructed analytical solution directly. Unless specified individually, the integration will be carried out until a stopping time $t = 1$ and the diffusion coefficients are selected as $\alpha^- = 1$ and $\alpha^+ = 10$. The domain size is chosen as a non-integer so that one can avoid the situation, in which the interface intersects the Cartesian grid lines through one grid node, because such a situation rarely happens for a complex geometry in practice. Moreover, by taking the domain size very close to an integer, such as 1.99, the corner interface case could be encountered in a coarse mesh. This enables us to fully validate the proposed matched ADI algorithm.

The numerical performance of the proposed matched MIB scheme is measured by two error measurements, L_∞ and L_2 norms, which are defined as

$$L_\infty = \max_{i,j,k} \mid u(x_i, y_j, z_k) - u_h(x_i, y_j, z_k) \mid, \tag{34}$$

$$L_2 = \sqrt{\frac{\sum_{i,j,k}(u(x_i, y_j, z_k) - u_h(x_i, y_j, z_k))^2}{N_x * N_y * N_z}} \tag{35}$$

where $u_h(x_i, y_j, z_k)$ is the numerical solution, and N_x, N_y and N_z represent the numbers of grid nodes on x, y and z directions. The order of convergence for both L_2 and L_∞ is reported in all tested cases. For the surface solution and error maps, the values on the interface points are mapped by the closest grid node from exterior subdomain Ω^+, and the numbers of mesh grid are chosen to be large so that the exterior value of the surface will be extremely close to its closest outside nodal value.

Example 1. The first example has a simple smooth ellipsoid surface, which is defined as the zero level set of

$$S(x, y, z) = \frac{x^2}{4} + y^2 + z^2 - 1. \tag{36}$$

This interface is regular and convex so that each grid line cuts the interface at most twice. The analytical solution to Eq. (1) is constructed to be

$$u(x, y, z, t) = \begin{cases} 10e^{-x^2}e^{-y^2}e^{-z^2} - e^{t-a}, & \text{in } \Omega^- \\ 5e^{-x^2}e^{-y^2}e^{-z^2} + e^{t-a}, & \text{in } \Omega^+ \end{cases} \tag{37}$$

with $a = 2$. The source term is

$$f(x, y, z, t) = \begin{cases} -10\alpha^-(-6 + 4(x^2 + y^2 + z^2))e^{-x^2}e^{-y^2}e^{-z^2} - e^{t-a}, & \text{in } \Omega^- \\ -5\alpha^+(-6 + 4(x^2 + y^2 + z^2))e^{-x^2}e^{-y^2}e^{-z^2} + e^{t-a}. & \text{in } \Omega^+, \end{cases} \tag{38}$$

so that the zeroth order jump condition on the interface is time-dependent, while the first order jump condition is time-independent. The computational domain is fixed to be $[-3.99, 3.99] \times [-1.99, 1.99] \times [-1.99, 1.99]$.

Table 1
Spatial convergence for ellipsoid surface.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	1.507e−01		2.016e−02	
[40, 40, 40]	4.521e−02	1.74	4.324e−03	2.22
[80, 80, 80]	9.833e−03	2.20	1.288e−03	1.75
[160, 160, 160]	2.187e−03	2.17	3.743e−04	1.78

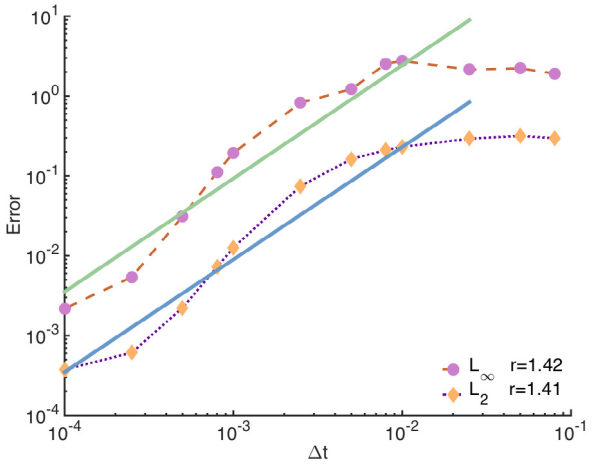

Fig. 3. Temporal convergence for ellipsoid surface.

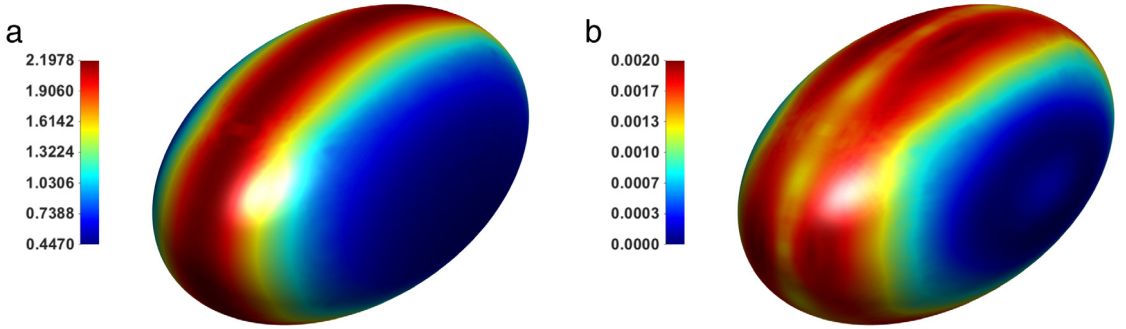

Fig. 4. Color maps for ellipsoid surface. (a) numerical solution; (b) numerical error. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The proposed matched ADI algorithm is stable and convergent for all tested combinations of h and Δt. In order to demonstrate the spatial convergence rate, $\Delta t = 10^{-4}$ is set to be small. Numerical results obtained by varying the number of grids per direction are demonstrated in Table 1, and it is obvious that the spatial convergence rate is close to 2 for both L_∞ and L_2 norms.

The next set of tests is performed with a dense space mesh $[N_x, N_y, N_z] = [160, 160, 160]$, and various time steps. The results are depicted in Fig. 3 with dashed lines. One can observe that the convergence is not evident for $\Delta t > 10^{-2}$, probably because of the relatively large errors in approximating tangential derivatives for a large Δt. Once the convergence starts at $\Delta t = 10^{-2}$, the rate is pretty high. To see this, a least-squares fitting [32] is conducted in the log–log scale to analyze the temporal convergence rate, which is demonstrated by the solid lines in Fig. 3. Their slopes shown in the legend of Fig. 3 represent the temporal convergence rate of the scheme. It shows that both solid lines for L_2 and L_∞ errors have slopes about 40% higher than their theoretical orders (one).

The color maps of numerical solutions and errors of Example 1 are depicted in Fig. 4 for $[N_x, N_y, N_z] = [160, 160, 160]$ and $\Delta t = 10^{-4}$. It is found that large errors occur in the places where the solutions have large magnitudes. This is natural due to the simple shape of the ellipsoid surface.

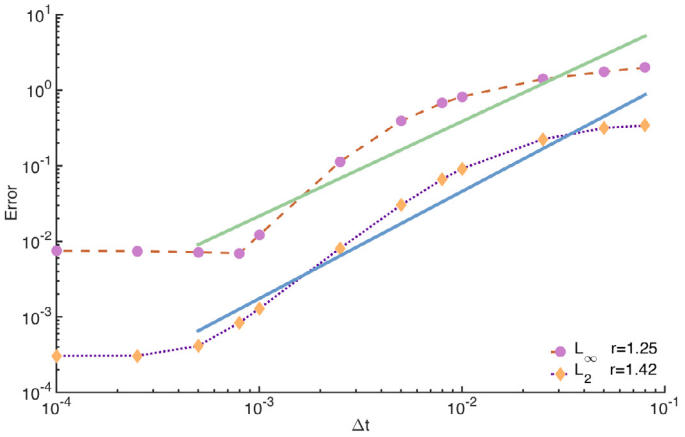

Fig. 5. Temporal convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 10$.

Table 2
Spatial convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 10$.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	4.548e−01		2.262e−02	
[40, 40, 40]	1.215e−01	1.90	5.122e−03	2.14
[80, 80, 80]	3.020e−02	2.00	1.227e−03	2.06
[160, 160, 160]	7.491e−03	2.11	3.035e−04	2.02

Table 3
Spatial convergence for cylinder surface with $\alpha^- = 10$ and $\alpha^+ = 1$.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	5.271e−01		3.802e−02	
[40, 40, 40]	1.074e−01	2.30	7.891e−03	2.27
[80, 80, 80]	2.571e−02	2.06	2.388e−03	1.72
[160, 160, 160]	6.788e−03	1.92	4.558e−04	2.39

Example 2. The second example has a simple cylinder surface, which is defined as an implicit function

$$\frac{x^2}{4} + \frac{y^2}{4} = 1, \quad -3 \leqslant z \leqslant 3. \tag{39}$$

In two-dimensional views, this interface is either a rectangle or a circle. The same analytical solution (37) and source term (38) with $a = 1$ as in Example 1 are studied. The computational domain is fixed to be $[-3.99, 3.99] \times [-3.99, 3.99] \times [-3.99, 3.99]$.

We are interested in studying different combinations of diffusion coefficients from this example. Besides the usual combination with $(\alpha^-, \alpha^+) = (1, 10)$, two more cases with $(\alpha^-, \alpha^+) = (10, 1)$ and $(\alpha^-, \alpha^+) = (1, 100)$ are examined. The 3D matched ADI method is found to be stable for all the tested h and Δt for these three pairs of diffusion coefficients. From Tables 2–4, to avoid the influence of temporal numerical errors, a small fixed time increment $\Delta t = 10^{-4}$ is adopted for all spatial convergence tests. The results of all three tables indicate that both L_2 and L_∞ errors achieve second order convergence rate regardless the diffusion coefficient's combinations. By fixing the number of grids in each direction to be 160, the temporal convergence is investigated by the least-square fitting analysis in Figs. 5–7. It is clear that L_2 and L_∞ errors yield a similar behavior, and both converge at a rate slightly higher than first order.

Example 3. The third example studies the impact of the curvature of the interface on the proposed numerical scheme. To this end, a smooth molecular surface of two atoms is constructed by the zero level set of

$$S(x, y, z) = \left(x^2 + y^2 + z^2 + \frac{3}{5}\right)^2 - \frac{7}{2}y^2 - \frac{3}{5} \tag{40}$$

within a cubic computational domain Ω of dimension $[-1.99, 1.99] \times [-2.99, 2.99] \times [-1.99, 1.99]$, as shown in Fig. 8. The concave region which connects the two atoms makes it impossible to derive the jump conditions $[\alpha u_\eta]$ and $[\alpha u_\zeta]$ without

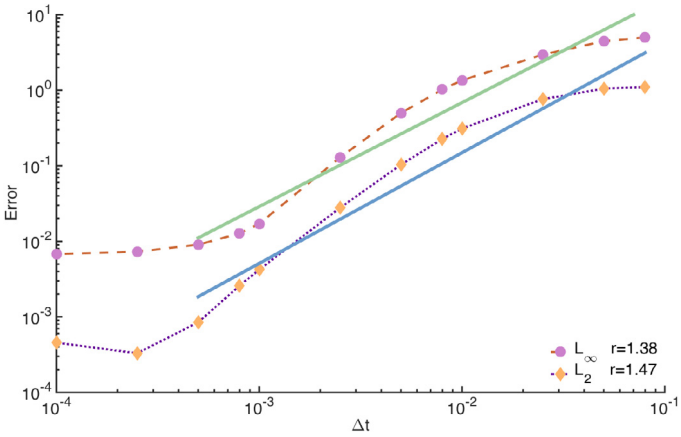

Fig. 6. Temporal convergence for cylinder surface with $\alpha^- = 10$ and $\alpha^+ = 1$.

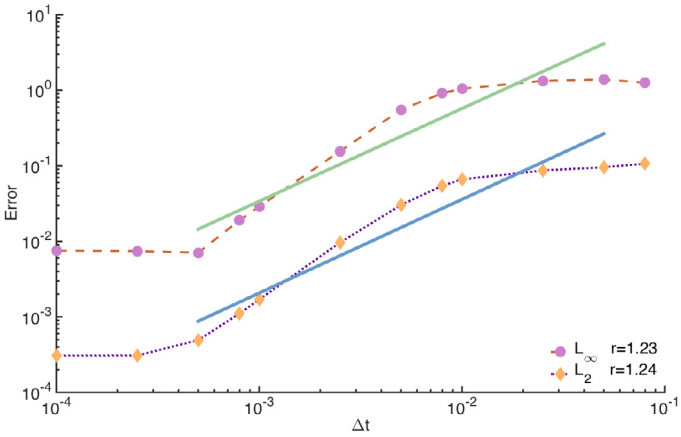

Fig. 7. Temporal convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 100$.

Table 4
Spatial convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 100$.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	4.562e−01		2.275e−02	
[40, 40, 40]	1.224e−01	1.90	5.241e−03	2.12
[80, 80, 80]	3.038e−02	2.01	1.243e−03	2.08
[160, 160, 160]	7.525e−03	2.01	3.058e−04	2.02

Table 5
Spatial convergence for molecular surface.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	2.135e−01		2.140e−02	
[40, 40, 40]	4.341e−02	2.30	3.833e−03	2.48
[80, 80, 80]	1.128e−02	1.94	1.143e−03	1.75
[160, 160, 160]	2.645e−03	2.09	2.503e−04	2.19

approximating the tangential derivatives u_η and u_ζ in both outer and inner domains (Ω^+ and Ω^-) simultaneously. The same analytical solution (37) and source term (38) with $a = 1$ are used.

Once again, the proposed matched ADI method is stable and converges for all tested h and Δt. To demonstrate the results, $\Delta t = 10^{-4}$ is fixed for spatial convergence tests and the results are shown in Table 5. The results clearly conclude that the

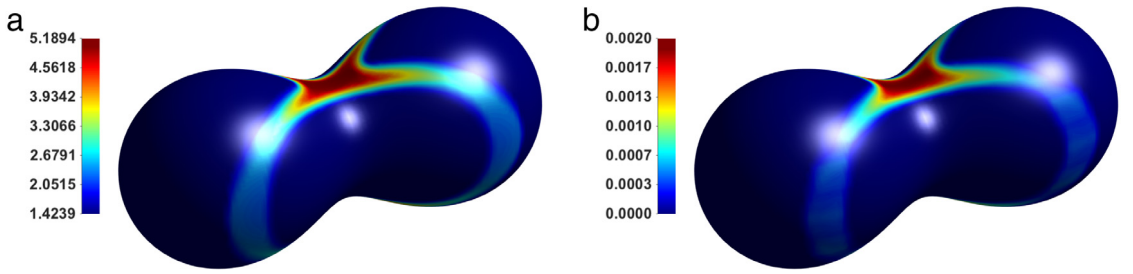

Fig. 8. Color maps for molecular surface. (a) Numerical solution; (b) numerical error. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

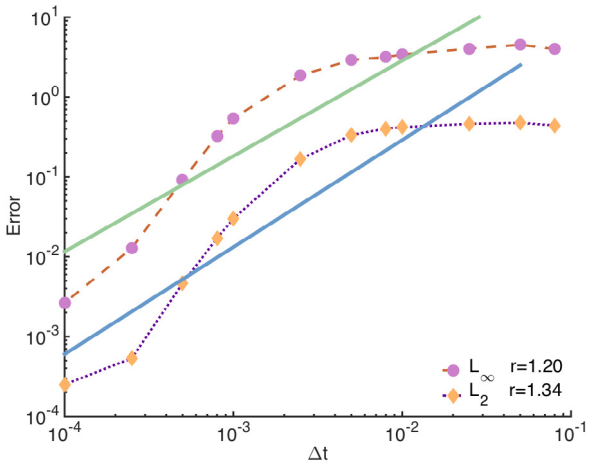

Fig. 9. Temporal convergence for molecular surface.

spatial convergence rate is two. The results obtained for the temporal convergence test by fixing the number of grids per direction to be 160 are shown in Fig. 9, together with the results of lease-squares error analysis. In this case, both L_∞ and L_2 errors decrease in a similar manner. Although it is not very uniform for all Δt's, the overall order of the matched ADI method is still higher than one for both error measurements. The color map of numerical solutions and errors are illustrated in Fig. 8 for $[N_x, N_y, N_z] = [160, 160, 160]$ and $\Delta t = 10^{-4}$. The error distribution pattern is almost the same as that of the solution.

To appreciate the numerical difficulty of the present parabolic interface problem, the slice plots of the matched ADI solutions at the cross section $z = 0$ are depicted in Fig. 10. For this purpose, we take $[N_x, N_y, N_z] = [161, 161, 161]$ and carry out the integration from $t = 0$ to $t = 2$ with $\Delta t = 10^{-3}$. As shown in Fig. 10, a strong jump discontinuity is developed in the solution as time increases. To view the hidden solution in the other side, the corresponding contour plots are shown in Fig. 11. Without plotting the interface Γ explicitly, these contour lines clearly show the location of Γ. Even though the solution pattern inside Γ is unchanged, the contrast of the solutions inside and outside Γ is variant. A strong jump discontinuity can also be sensed in the contour plot at time $t = 2$, because the contour lines now undergo a fast and sharp change near the interface.

Example 4. This example examines the performance of the proposed method on fully time-dependent jump conditions. To this end, we consider a smooth torus, which is given as the zero level set of

$$S(x, y, z) = \left(\frac{3}{2} - \sqrt{x^2 + y^2}\right)^2 + z^2 - \frac{49}{100} \tag{41}$$

in a computational domain of dimension $[-3.99, 3.99] \times [-3.99, 3.99] \times [-3.99, 3.99]$. The analytical solution is constructed in the form of

$$u(x, y, z, t) = \begin{cases} \cos(ax)\sin(ay)\cos(az)\cos(t) - 1, & \text{in } \Omega^- \\ \sin(ax)\cos(ay)\sin(az)\cos(t) + 1, & \text{in } \Omega^+ \end{cases} \tag{42}$$

so that $a = 2$ yields a periodic solution with a period of π. The source term is given by

$$f(x, y, z, t) = \begin{cases} (3\alpha^- a^2 \cos(t) - \sin(t))\cos(ax)\sin(ay)\cos(az) & \text{in } \Omega^- \\ (3\alpha^+ a^2 \cos(t) - \sin(t))\sin(ax)\cos(ay)\sin(az) & \text{in } \Omega^+. \end{cases} \tag{43}$$

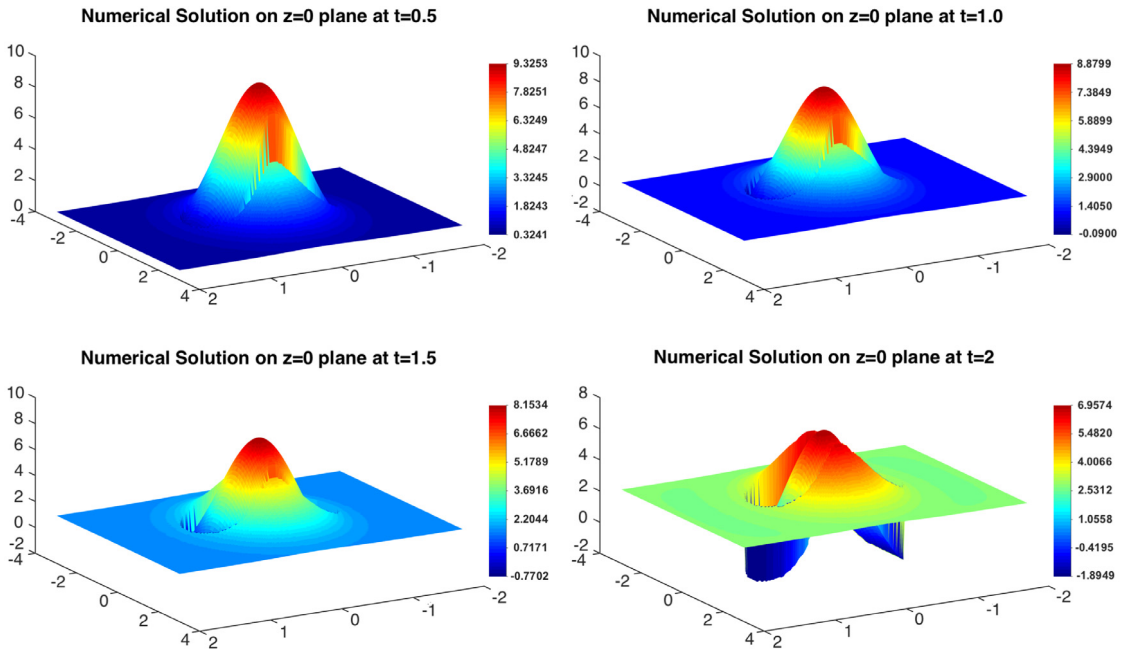

Fig. 10. Slice plots of the numerical solutions on $z = 0$ plane in Example 3.

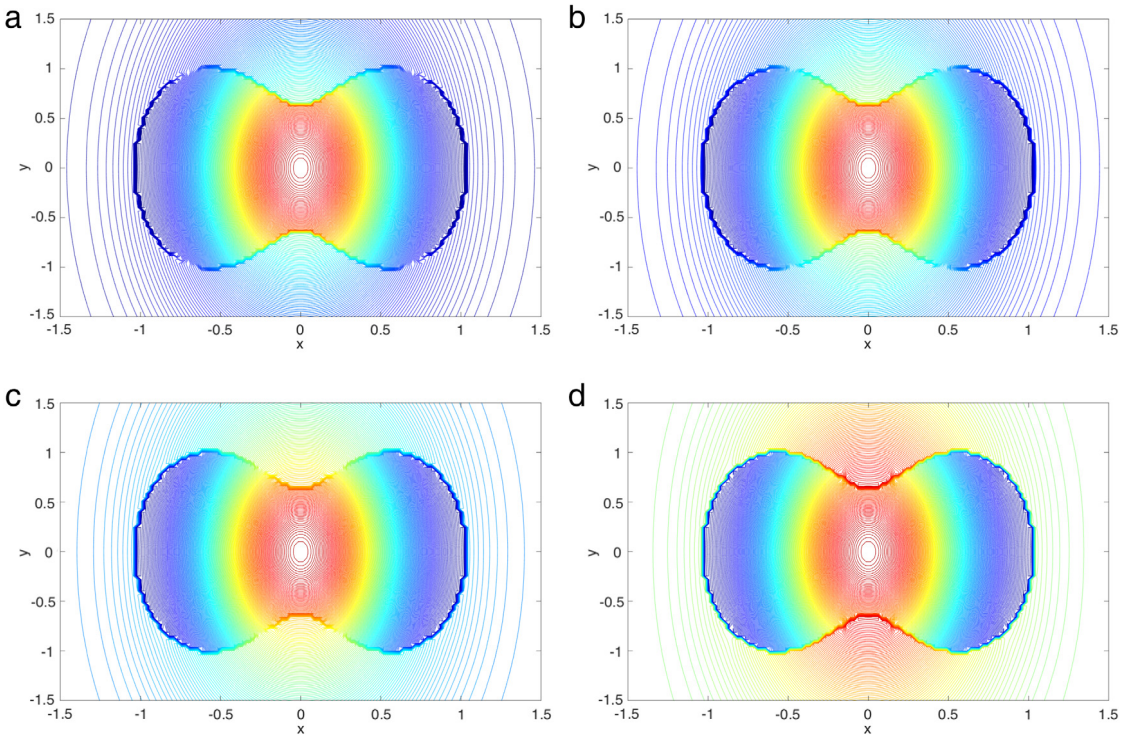

Fig. 11. Contour plots of the numerical solutions on $z = 0$ plane in Example 3. (a) $t = 0.5$; (b) $t = 1$; (c) $t = 1.5$; (d) $t = 2$.

In this example, the interface jump conditions are time-and-space dependent, which represents the most general physical jump conditions for parabolic interface problems.

Table 6
Spatial convergence for torus surface.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	2.774e−01		1.908e−02	
[40, 40, 40]	3.803e−02	2.87	2.767e−03	2.79
[80, 80, 80]	1.146e−02	1.73	6.554e−04	2.08
[160, 160, 160]	8.254e−03	0.47	1.602e−04	2.03

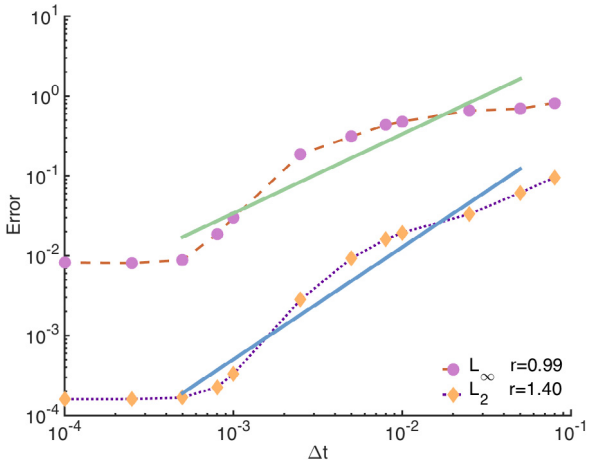

Fig. 12. Temporal convergence for torus surface.

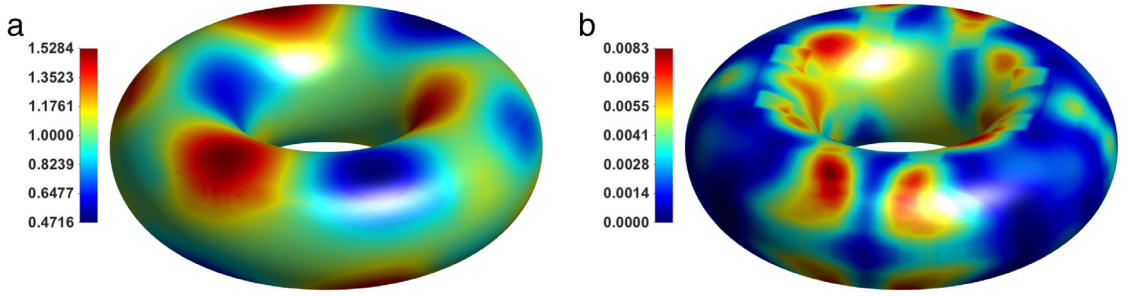

Fig. 13. Color maps for torus surface. (a) Numerical solution; (b) numerical error. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The 3D matched ADI method is stable for this example, and the results obtained by the spatial convergence tests are presented in Table 6. It can be observed that the convergence rate in the L_2 norm is still two. However, the L_∞ convergence becomes a little deteriorated in this example, especially for the last entry, even though the overall rate is still about 1.6. The temporal tests are shown in Fig. 12 which reveals a similar pattern, i.e., the L_2 order is similar to the previous examples, while the L_∞ rate is affected somehow. For more detailed insights, the color maps are demonstrated in Fig. 13. We can see that the error distribution pattern is quite random in this example, which means that some large errors occur on or near the interface Γ, even when the solution or its gradient is not large. The L_∞ convergence patterns shall be related to such an irregular error distribution.

Example 5. This example deals with a complicated topological shape of a tanglecube, defined as the zero level set of

$$S(x, y, z) = x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 10 \tag{44}$$

embedded in a domain of dimension $[-3.99, 3.99] \times [-3.99, 3.99] \times [-4.99, 4.99]$. The tanglecube surface has several convex and concave regions which pose great challenges for the tangential derivative approximations. The previous analytical solution (42) is reused with a different coefficient $a = 1$ so that all jump conditions are still time dependent. For this example, one grid line will cut the interface in zero, two or four times.

Table 7
Spatial convergence for tanglecube surface.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	2.394e−01		1.743e−02	
[40, 40, 40]	6.940e−02	1.79	4.890e−03	1.83
[80, 80, 80]	2.162e−03	5.00	2.688e−04	4.19
[160, 160, 160]	1.112e−03	0.96	7.762e−05	1.79

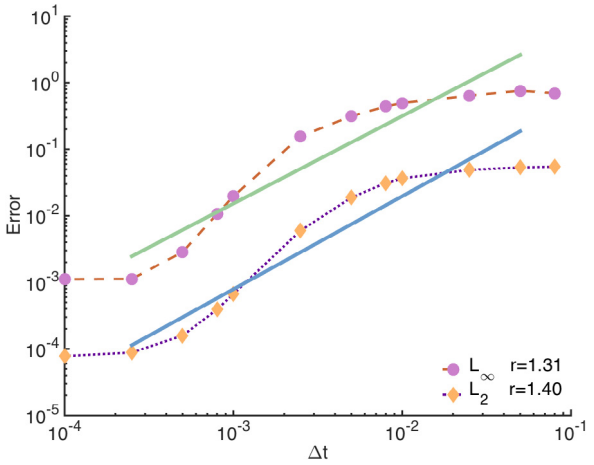

Fig. 14. Temporal convergence for tanglecube surface.

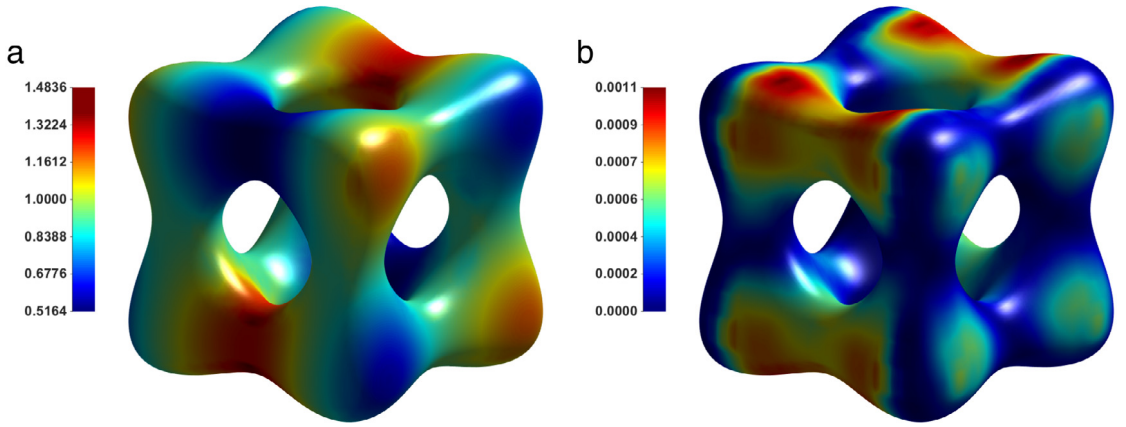

Fig. 15. Color maps for tanglecube surface. (a) Numerical solution; (b) numerical error. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The matched ADI method is stable for this example. Similar spatial and temporal tests are repeated and the results are shown in Table 7 and Fig. 14, respectively, together with the color maps shown in Fig. 15. Even though the convergence is not uniform, the averaged spatial convergence rates in both error norms are above two. The temporal convergence rate is similar to what obtained in previous examples. Moreover, the color maps show that the maximum errors occur not only in the regions where the solution achieves maximum values, but also in the regions where the largest jumps take place across the interface.

We also examine the efficiency of the 3D matched ADI method for this example in two aspects, CPU time and memory usage. For the CPU time, the complexity of the proposed method is on the order of $O(N)$ with $N = N_x * N_y * N_z$ per time step. In Fig. 16(a), the CPU time is plotted against N for evolving 1000 time steps. The mesh grid per direction is chosen from 20 to 220 with an increment of 10. Computationally, the total number of interface points which require special interface treatments is different for different mesh sizes. However, the interface treatments are conducted only on irregular nodes near the surface. Moreover, they need to be done only once before the time integration. Thus, the CPU time shown in Fig. 16(a)

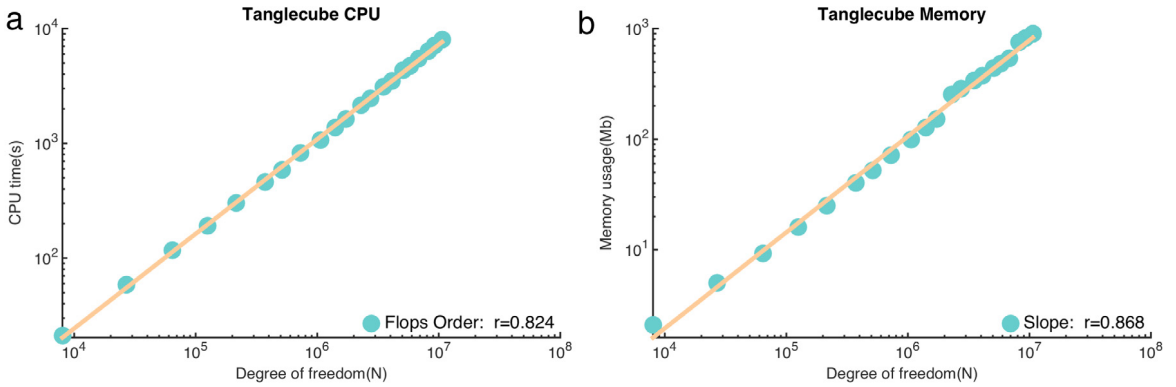

Fig. 16. CPU time and memory usage of tanglecube (a) CPU time; (b) memory usage.

Table 8
Spatial convergence for heart surface.

$[N_x, N_y, N_z]$	L_∞		L_2	
	Error	Order	Error	Order
[20, 20, 20]	1.456e−01		3.560e−02	
[40, 40, 40]	4.490e−02	1.70	8.364e−03	2.09
[80, 80, 80]	2.158e−02	1.06	2.073e−03	2.01
[160, 160, 160]	5.701e−03	1.92	5.186e−04	2.00

is still dominated by the 3D ADI computation, not the MIB interface treatments. Consequently, the execution time scales linearly against N in Fig. 16(a). A least-square fitting in log–log scale is conducted with a slope $r = 0.824$. On the other hand, the memory usage mainly consists of two parts, the structure which stores the interface informations and the sparse matrices for computation at each time step. Obviously, the denser the mesh is, the more interface locations need to be constructed. Moreover, all discretization matrices are perturbed tridiagonal ones. We only need to store the non-zero elements of these sparse matrices in our computation. Accordingly, the overall storage for both interface locations and ADI matrices increases linearly with respect to the degree of freedom N, as shown in Fig. 16(b) with slope $r = 0.868$. The present study demonstrates that the ADI method is an extremely efficient non-iterative algebraic solver for parabolic problems.

Example 6. The last example considers a heart-shaped interface, which is the zero level set of

$$S(x, y, z) = \left(x^2 + \frac{9}{4}y^2 + z^2 - 1\right)^3 - x^2z^3 - \frac{9}{80}y^2z^3 \tag{45}$$

embedded in the domain of dimension $[-3.99, 3.99] \times [-1.99, 1.99] \times [-3.99, 3.99]$. The analytical solution is constructed by the formula

$$u(x, y, z, t) = \begin{cases} (\cos(ax) + \sin(ay) + \cos(az))\sin(t), & \text{in } \Omega^- \\ (\sin(ax) + \cos(ay) + \sin(az))\sin(t), & \text{in } \Omega^+ \end{cases} \tag{46}$$

where $a = 2$ in this example. The source term is given by

$$f(x, y, z, t) = \begin{cases} \alpha^- a^2(\cos(ax) + \sin(ay) + \cos(az))\sin(t) + (\cos(ax) + \sin(ay) + \cos(az))\cos(t), & \text{in } \Omega^- \\ \alpha^+ a^2(\sin(ax) + \cos(ay) + \sin(az))\sin(t) + (\sin(ax) + \cos(ay) + \sin(az))\cos(t). & \text{in } \Omega^+ \end{cases} \tag{47}$$

The significance of this example is that its interface has two singularities, one on the top and the other at the bottom, as shown in Fig. 17. The singularities have not been considered in previous examples, and it is of great interest to see how well the proposed matched ADI scheme performs on this example. Due to the proper corner treatment, it is found that the singularities on the interface have little impact on the spatial convergence rate, as shown in Table 8. The spatial convergence rate is close to 2 in the L_2 norm, and is slightly reduced for the L_∞ norm. The temporal convergence test in Fig. 18 indicates that the temporal order is still higher than one.

The CPU time and memory usage for this example are reported in Fig. 19, with the same setting as in Example 5. Again both of them are found to be linearly scaled with slope rates $r = 0.934$ and $r = 0.925$, respectively. The 3D matched ADI method is founded to be stable for the present example with singularities, which is demonstrated in Fig. 20. By considering a very large time step $\Delta t = 10$, we calculate $N_x = N_y = N_z = 20, 40, 80, 160$ in each direction and choose the stopping time varying from 10^3 to 10^5. It can be seen that the errors do not grow with respect to the time for all mesh sizes. Great stability has been observed in all tested cases.

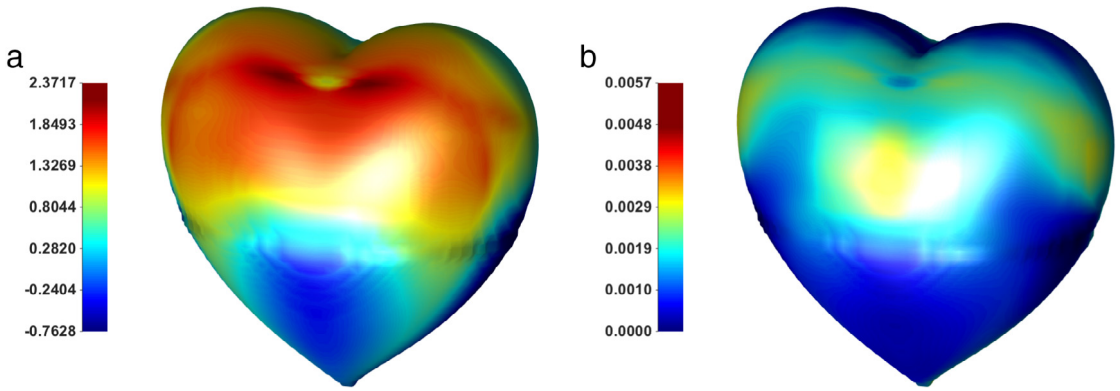

Fig. 17. Color maps for heart surface. (a) Numerical solution; (b) numerical error. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

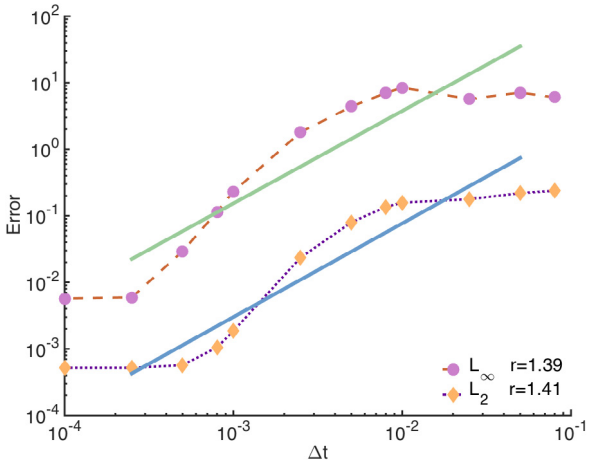

Fig. 18. Temporal convergence for heart surface.

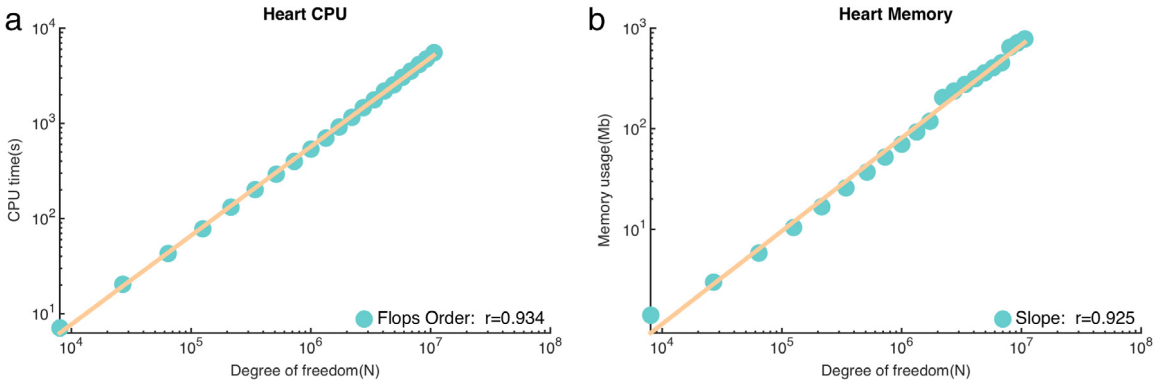

Fig. 19. CPU time and memory usage of heart (a) CPU time; (b) memory usage.

4. Conclusion

In this work, we propose a novel matched ADI method for solving 3D parabolic interface problems with discontinuous diffusion coefficient and complex interfaces. The newly developed scheme inherits the merits of its ancestor for solving 2D interface problems, while possesses unique features, such as the non-orthogonal local coordinate system for decoupling the

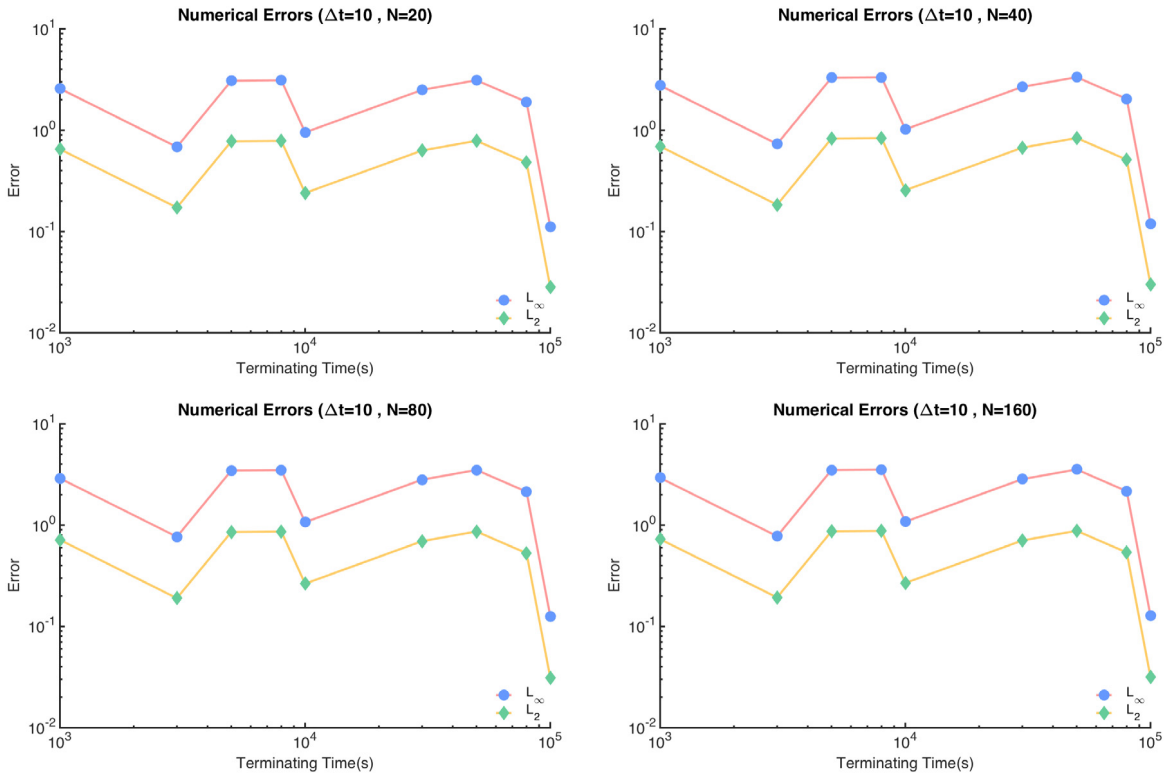

Fig. 20. Stability test for the heart surface in Example 6.

jump conditions, two-side estimation of tangential derivative at the interface point, and the capability of handling arbitrary many interface points on one grid line, for solving more challenging 3D problems. Moreover, a new Douglas–Rachford ADI scheme is designed, which minimizes the perturbation terms, compared with the implicit Euler scheme, and is of first order in time.

The proposed method is found to be convergent and stable for various complex-shaped interfaces and time-and-space dependent jump conditions in the numerical experiments. Being highly accurate and efficient, this new ADI method provides a very promising numerical algorithm for solving 3D parabolic PDEs. In terms of accuracy, the matched ADI method yields the spatially second order of accuracy for all tested interface problems. Due to strong stability, a large time increment can be used in the ADI method for an efficient time integration. Moreover, in each time step, the ADI method provides an exact or non-iterative algebraic solver whose complexity scales linearly with respect to the spatial degree of freedom. Therefore, the matched ADI method is definitely one of the fastest implicit algorithms for solving 3D parabolic interface problems.

Our plan of the next stage is to further improve its convergence rate in time and apply it to more complicated problems.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) grant DMS-1318898 and the University of Alabama Research Stimulation Program (RSP) award.

References

[1] H.H. Pennes, Analysis of tissue and arterial blood temperatures in the resting human forearm, J. Appl. Physiol. 1 (2) (1948) 93–122.
[2] W. Geng, S. Zhao, Fully implicit ADI schemes for solving the nonlinear Poisson-Boltzmann equation, Mol. Based Math. Biol. 1 (2013) 109–123.
[3] S. Zhao, Operator splitting ADI schemes for pseudo-time coupled nonlinear solvation simulations, J. Comput. Phys. 257 (2014) 1000–1021.
[4] J.J.B. Jack, D. Noble, R.W. Tsien, Electric Current Flow in Excitable Cells, Clarendon Press Oxford, 1975.
[5] C. Attanayake, D. Senaratne, Convergence of an immersed finite element method for semilinear parabolic interface problems, Appl. Math. Sci. 5 (3) (2011) 135–147.
[6] Z. Chen, J. Zou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numer. Math. 79 (2) (1998) 175–202.
[7] R.K. Sinha, B. Deka, Optimal error estimates for linear parabolic problems with discontinuous coefficients, SIAM J. Numer. Anal. 43 (2) (2005) 733–749.
[8] R.K. Sinha, B. Deka, Finite element methods for semilinear elliptic and parabolic interface problems, Appl. Numer. Math. 59 (8) (2009) 1870–1883.
[9] S. Wang, R. Samulyak, T. Guo, An embedded boundary method for elliptic and parabolic problems with interfaces and application to multi-material systems with phase transitions, Acta Math. Sci. 30 (2) (2010) 499–521.

[10] R.J. Leveque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044.
[11] L. Adams, Z. Li, The immersed interface/multigrid methods for interface problems, SIAM J. Sci. Comput. 24 (2) (2002) 463–479.
[12] F. Bouchon, G.H. Peichl, The immersed interface technique for parabolic problems with mixed boundary conditions, SIAM J. Numer. Anal. 48 (6) (2010) 2247–2266.
[13] F. Bouchon, G.H. Peichl, An immersed interface technique for the numerical solution of the heat equation on a moving domain, in: Numerical Mathematics and Advanced Applications 2009, Springer, 2010, pp. 181–189.
[14] J.D. Kandilarov, L.G. Vulkov, The immersed interface method for a nonlinear chemical diffusion equation with local sites of reactions, Numer. Algorithms 36 (4) (2004) 285–307.
[15] J.D. Kandilarov, L.G. Vulkov, The immersed interface method for two-dimensional heat-diffusion equations with singular own sources, Appl. Numer. Math. 57 (5) (2007) 486–497.
[16] G. Horton, S. Vandewalle, A space-time multigrid method for parabolic partial differential equations, SIAM J. Sci. Comput. 16 (4) (1995) 848–864.
[17] J. Douglas, D. Peaceman, Numerical solution of two-dimensional heat-flow problems, AIChE J. 1 (4) (1955) 505–512.
[18] J. Douglas Jr., On the Numerical Integration of ˆ2uxˆ2+ˆ2uyˆ2 = ut by Implicit Methods, J. Soc. Ind. Appl. Math. 3 (1) (1955) 42–65.
[19] D. Paeceman, H. Rachford, The numerical solution of parabolic and elliptic equations, J. Soc. Ind. Appl. Math. 3 (1955) 28–41.
[20] J.C. Strikwerda, Finite Difference Schemes and Partial Differential Equations, SIAM, 2004.
[21] C. Li, S. Zhao, A matched Peaceman–Rachford ADI method for solving parabolic interface problems, Appl. Math. Comput. 299 (2017) 28–44.
[22] Z. Li, A. Mayo, ADI methods for heat equations with discontinuous along an arbitrary interface, in: Proc. Sympos. Appl. Math., Vol. 48, 1993, pp. 311–315.
[23] Z. Li, Y.-Q. Shen, A numerical method for solving heat equations involving interfaces, in: Electronic Journal of Differential Equations, Conf., Vol. 3, AMS, 1999, pp. 100–108.
[24] J. Liu, Z. Zheng, IIM-based ADI finite difference scheme for nonlinear convection–diffusion equations with interfaces, Appl. Math. Model. 37 (3) (2013) 1196–1207.
[25] J. Liu, Z. Zheng, A dimension by dimension splitting immersed interface method for heat conduction equation with interfaces, J. Comput. Appl. Math. 261 (2014) 221–231.
[26] S. Zhao, A matched alternating direction implicit (ADI) method for solving the heat equation with interfaces, J. Sci. Comput. 63 (1) (2015) 118–137.
[27] S. Yu, G. Wei, Three-dimensional matched interface and boundary (MIB) method for treating geometric singularities, J. Comput. Phys. 227 (1) (2007) 602–632.
[28] S. Yu, Y. Zhou, G.-W. Wei, Matched interface and boundary (MIB) method for elliptic problems with sharp-edged interfaces, J. Comput. Phys. 224 (2) (2007) 729–756.
[29] S. Zhao, G. Wei, High-order FDTD methods via derivative matching for Maxwell's equations with material interfaces, J. Comput. Phys. 200 (1) (2004) 60–103.
[30] Y. Zhou, S. Zhao, M. Feig, G.-W. Wei, High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources, J. Comput. Phys. 213 (1) (2006) 1–30.
[31] B. Fornberg, Classroom note: Calculation of weights in finite difference formulas, SIAM Rev. 40 (3) (1998) 685–691.
[32] S. Zhao, G. Wei, Matched interface and boundary (MIB) for the implementation of boundary conditions in high-order central finite differences, Internat. J. Numer. Methods Engrg. 77 (12) (2009) 1690–1730.