Noname manuscript No.
(will be inserted by the editor)

A Fast Sine Transform Accelerated High-Order Finite Difference Method for Parabolic Problems over Irregular Domains

Chuan Li · Yiming Ren · Guangqing Long · Eric Boerman · Shan Zhao

June 10, 2022

Abstract In this paper, a new Cartesian grid finite difference scheme is introduced for solving parabolic initial-boundary value problems involving irregular domains and Robin boundary condition in two and three dimensions. In spatial discretization, a ray-casting matched interface and boundary (MIB) method is utilized to enforce different types of boundary conditions, including Dirichlet, Neumann, Robin, and their mixed combinations, along the normal direction to generate necessary fictitious values outside the irregular domain. This allows accurate approximations of jumps in derivatives at various boundary locations so that the fourth-order central difference can be corrected at all Cartesian nodes. By treating such corrections as additional unknowns, the order of finite difference discretization of the Laplacian operator can be preserved. Moreover, by constructing corrections for different types of irregular

Chuan Li
Department of Mathematics, West Chester University of Pennsylvania, West Chester, PA 19383, USA
E-mail: cli@wcupa.edu

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

Guangqing Long
Guangxi Key Lab of Human-Machine Interaction and Intelligent Decision, Nanning Normal University, Nanning 530001, PR China

Eric Boerman
Department of Mathematics, West Chester University of Pennsylvania, West Chester, PA 19383, USA

Shan Zhao
Department of Mathematics, University of Alabama, Tuscaloosa, AL 35487, USA
E-mail: szhao@ua.edu

and corner points, the proposed augmented MIB (AMIB) method can accommodate complicated geometries, while maintaining the fourth order of accuracy in space. In temporal discretization, the standard Crank-Nicolson scheme is employed, which is second-order in time and unconditionally stable. Furthermore, a Fast Sine Transform acceleration algorithm is employed to efficiently invert the discrete Laplacian, so that the augmented linear system in each time step can be solved with a complexity of $O(N \log N)$, where N stands for the total spatial degree-of-freedom. The accuracy, stability and efficiency of the proposed AMIB method are numerically validated by considering various parabolic problems in two and three dimensions.

Keywords Parabolic initial-boundary value problems · Irregular domains · Robin boundary condition · Matched interface and boundary (MIB) · Fast Fourier transform

Mathematics Subject Classification (2020) 65M06 · 65M85 · 65M12

1 Introduction

This work focuses on solving a parabolic Partial Differential Equation (PDE),

$$\frac{\partial u}{\partial t} = \beta \Delta u + f, \tag{1}$$

over an irregularly shaped domain Ω in two (2D) or three (3D) dimensional spaces. The boundary of Ω, denoted by $\Gamma = \partial\Omega$, is governed by a level set function or prescribed in a parametric form. See Fig. 1a for a 2D demonstration. While Γ is fixed, the unknown function u and the source term f in eqn. (1) could be both time and space dependent, and the diffusion coefficient $\beta > 0$ is a constant. An initial condition at $t = 0$ is given as

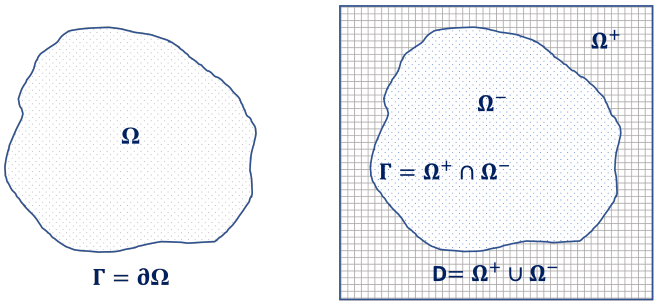
$$u(0, \mathbf{x}) = u_0(\mathbf{x}), \tag{2}$$

and a generic boundary condition on the boundary Γ is given as

$$\alpha_\Gamma u + \beta_\Gamma \frac{\partial u}{\partial n} = \phi(t, \mathbf{x}), \tag{3}$$

where $\mathbf{x} = (x, y)$ in two dimensions and $\mathbf{x} = (x, y, z)$ in three dimensions. Notice that eqn. (3) represents three commonly used boundary conditions, i.e., Dirichlet ($\alpha_\Gamma \neq 0$ and $\beta_\Gamma = 0$), Neumann ($\alpha_\Gamma = 0$ and $\beta_\Gamma \neq 0$), and Robin ($\alpha_\Gamma \neq 0$ and $\beta_\Gamma \neq 0$).

In this work, in order to numerically solve the initial and boundary value problem (IBVP) formed by eqn. (1) - (3), the domain Ω is embedded inside a larger regular computational domain D. For instance, the rectangular domain could be $D = [a, b] \times [c, d]$ in 2D. The original IBVP can be reformulated as an immersed parabolic problem to be solved in the new domain D as shown in Fig. 1b. In Fig. 1b, the original domain Ω is renamed as Ω^-. We also denote $\Omega^+ = D/\Omega^-$ so that $D = \Omega^+ \cup \Omega^-$ and $\Gamma = \Omega^+ \cap \Omega^-$. The original boundary



(a) An IBVP defined in a 2D complex-shaped domain Ω.

(b) A corresponding immersed problem defined in a 2D rectangular domain D.

Fig. 1: A demonstration of the original and corresponding immersed problems.

Γ is then treated as an *interface* splitting the outer subdomain Ω^+ and the inner subdomain Ω^-. Points on Γ are then renamed *interface points* to be distinguished from boundary points on ∂D. The original boundary condition (3) is also renamed to be an *interface condition* which is still required to hold on the interface Γ.

The solution u and source term f of eqn. (1) are assumed to be extended to the new domain D as

$$u(t,\mathbf{x}) = \begin{cases} u(t,\mathbf{x}), & \text{if } \mathbf{x} \in \Omega^- \\ 0, & \text{if } \mathbf{x} \in \Omega^+, \end{cases} \quad \text{and} \quad f(t,\mathbf{x}) = \begin{cases} f(t,\mathbf{x}), & \text{if } \mathbf{x} \in \Omega^- \\ 0, & \text{if } \mathbf{x} \in \Omega^+. \end{cases} \tag{4}$$

In what follows, initial condition (2) is extended in a similar manner,

$$u(0,\mathbf{x}) = \begin{cases} u_0(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega^- \\ 0, & \text{if } \mathbf{x} \in \Omega^+, \end{cases} \tag{5}$$

and a zero Dirichlet boundary condition,

$$u(t,\mathbf{x}) = 0, \text{ for } \mathbf{x} \in \partial D, \tag{6}$$

is imposed. The reformulated *immersed parabolic problem* consists of one PDE (1) with solution and source term piecewisely defined in eqn. (4) in domain D, an initial condition (5), an interface condition (3) imposed on Γ, and a boundary condition (6) on ∂D. The solution to eqn. (4) is expected to solve the original IBVP of eqn. (1) - (3) in subdomain Ω^-, as shown in Fig. 1b.

The goal of this work is to develop a new finite difference method for solving the immersed parabolic problem based on unconditionally stable Crank-Nicolson time stepping. This new method will not only achieve a fourth order of accuracy in space while handling irregular domains and complex boundary conditions, but also can deliver a $O(N \log N)$ efficiency in solving the linear system in each time step, where N stands for the total spatial degree-of-freedom. We note that several high-order numerical methods [23,34,31,37,

21] have already been developed in the literature for similar problems. A more detailed review about the existing literature will be offered in the next section.

On the other hand, we note that the present immersed parabolic problem shares some similarities with the parabolic interface problem [10,45,30], in which the solution to the heat equation is defined piecewisely for both Ω^- and Ω^+, together with two interface conditions governing function jump and flux jump over Γ and a nontrivial boundary condition on ∂D. Because in parabolic interface problems more boundary/interface conditions should be satisfied and the solution in Ω^+ needs to be approximated, the numerical development becomes more challenging in comparison to the present study of irregular domain IBVPs. Various jump capturing numerical schemes have been successfully developed for parabolic interface problems, including finite element methods [10, 30,42,36,50] and finite difference methods [29,45,27,28,39,26,20]. To the best of our knowledge, while some of these methods could deliver a $O(N \log N)$ efficiency, none of them achieves a fourth order of convergence in space for curved interfaces.

The rest of this work is organized as follows. In Section 2, we review relevant prior works for immersed parabolic problems. Section 3 provides detailed descriptions, theoretical results, and performance analysis on the proposed method. Multiple 2D and 3D numerical examples are tested for convergence orders, stability, and computational complexity in section 4, followed by the conclusion of this work in section 5.

2 Relevant literature

Numerous numerical methods have been developed for solving parabolic IBVP over irregular domain in an immersed setting. In the following, we will highlight some computational methods that can handle complex geometry and boundary conditions, or achieve a high order of convergence, or is equipped with fast algebraic solvers. Some advanced methods for the parabolic interface problem will be reviewed as well because, though they are designed for more difficult problems, they could be reformulated to solve the immersed parabolic IBVP too.

The ghost-fluid method (GFM) was first proposed by Fedkiw et al. [15] for solving interface problems in multi-material flows. Due to its flexibility in handling boundary conditions and irregular domains, the GFM has been successfully applied for solving parabolic PDEs. In [23], Gibou and Fedkiw introduced a fourth-order finite difference for diffusion equations with Dirichlet boundary conditions, which is the first high-order (more than second-order) method for immersed parabolic problems in the literature. The main idea is to generate ghost values outside Ω along Cartesian directions by using the Dirichlet boundary condition to construct high-order polynomials. This GFM has been further improved in [9] by allowing the use of non-graded adaptive Cartesian grids for capturing small length scales. A review of GFM, as well as

related level-set methods and quadtree/octree adaptive mesh refinements, is offered in [24].

In comparison to the Dirichlet boundary condition, the Robin and Neumann conditions are more difficult to be treated numerically. A second-order accurate finite volume scheme was first developed in [34] for solving Robin boundary condition over complex domains, which has been generalized to provide a second-order gradient approximation and for handling piecewise smooth boundaries in [3]. This finite volume scheme has been combined with operator splitting and semi-Lagrangian methods in [2] for attacking advection-diffusion problems with Robin boundary conditions in moving domains. Recently, second-order treatments of Robin conditions or mixed Dirichlet-Neumann-Robin boundary conditions in GFM-involved finite difference methods have been introduced in [7,8]. The ghost values in the normal direction are obtained by constructing a linear polynomial and successively extrapolating the normal derivative using a PDE approach.

The development of high-order (at least third-order) methods for irregular domain parabolic IBVPs has attracted some attention in the literature. Such methods are ideal for problems associated with high-frequency solutions and are usually more cost-efficient than second-order methods. The fourth-order GFM for Dirichlet problems introduced by Gibou and Fedkiw [23] is one of the pioneer studies in this direction. The existing high-order methods could be clarified into two general categories.

In the first category, certain smooth extensions of physical solutions beyond the irregular domain are carried out so that high-order discretization can be established for the entire computational domain. For example, the Fourier continuation (FC) method introduced in [4,5] is able to construct a smooth periodic function extension for irregular domain problems and can overcome the Gibbs phenomenon. Based on it, an alternating-direction fast solver (FC-AD) has been developed in [6,31] to provide spectral accuracy in solving parabolic, elliptic, and hyperbolic PDEs with Dirichlet boundary conditions. In [37], a different high-order extension method has been proposed by using a flexible immersed boundary smooth extension (IBSE). In combination with the Fourier spectral method, the IBSE attains fourth-order convergence for Dirichlet problems and third-order convergence for Neumann problems, in solving several different PDEs. There exist other high-order smooth extension methods for PDE interface problems, such as the correction function method [32] and kernel-free boundary integral method [41].

In the second category, several layers of ghost cells or fictitious points are generated outside the irregular domain Ω, so that high-order finite difference approximations can be carried out for all mesh points inside Ω. The fourth-order GFM [23] is one of such methods for treating Dirichlet boundary conditions. An arbitrary-order reconstruction off-site data (ROD) method has been introduced in [21,11] for enforcing Dirichlet, Neumann or Robin boundary conditions over irregular domains. By appropriately choosing collar points on Γ and enough supporting nodes inside Ω, a linear squares fitting is conducted in the ROD to generate 2D high-order polynomials, which can then produce

the desired ghost cell values. Up to sixth-order finite difference methods have been reported for solving several different PDEs [21,11]. A matched interface and boundary (MIB) method has been developed in [44] for treating perfectly electric conducting (PEC) boundary conditions over complex domains in solving Helmholtz eigenvalue problems. Based on it, a ray-casting MIB method has recently been proposed for solving elliptic PDEs with Dirichlet, Neumann, or Robin boundary conditions and their mixed combinations [35]. The ray-casting MIB scheme can achieve arbitrarily high order in handling irregular domains, and up to eighth order is reported in [35]. Moreover, generalization to 3D PDE problems has been demonstrated. We note that a potential advantage of the methods in this category is that the generation of ghost cells or fictitious points is a process independent from the discretization inside Ω. This is essentially why such methods can be applied to different PDEs, and could be arbitrarily high order in principal.

Besides spatial discretization, time integration is also crucial when solving parabolic problems. Explicit time schemes are simple and cheap in each time step, but a small time step size Δt has to be used due to severe stability conditions. Implicit time schemes could be unconditionally stable and thus allow the use of a large Δt for long-time integration or steady-state solution problems. Nevertheless, in each implicit time step, one has to solve a linear system, which involves all spatial degrees of freedom N. With a generic iterative solver, the algebraic solution could require a computational complexity around $O(N^2)$ in each time step. For a large N, particularly in multidimensional problems, this leads to an intractable demand for computational time.

Much research interest has been devoted to developing fast algebraic solvers for large linear systems, so that the implicit time schemes for parabolic PDEs could be as efficient as explicit schemes while enjoying the unconditional stability. The popular fast algorithms in scientific computing include fast Fourier transform (FFT), multigrid, or alternating directional implicit (ADI) methods, which have a complexity on the order of $O(N)$ or $O(N \log N)$. The FFT algorithm will be employed in the present study for solving the immersed parabolic problems in $O(N \log N)$ speed. However, we note that the FFT could not be extended to parabolic interface problems, due to the non-constant diffusion coefficient. An overview of ADI and multigrid techniques is offered below for parabolic interface problems. These fast algorithms can be applied to immersed parabolic problems in principle.

The classical ADI method [14] introduced in the 1950s is well known for its $O(N \log N)$ efficiency in solving parabolic PDEs, because one can reduce a multidimensional linear system into a sequence of independent one-dimensional (1D) subsystems of tridiagonal structure and then solve them efficiently by the Thomas algorithm. It is noted that the state-of-the-art ADI methods are no longer restricted to rectangular domains. In fact, many recent ADI methods can handle curved interfaces and/or irregular boundaries while maintaining $O(N \log N)$ efficiency. For example, Li and Mayo [29] introduced an immersed interface method (IIM) based ADI algorithm, which is the first of its kind for the parabolic interface problem in a simplified setting with the dif-

fusion coefficient being a constant throughout. The first ADI algorithm for the standard parabolic interface problem involving both function and flux jumps was presented in [45], in which the MIB method [46,49] is adopted for handling curved interfaces. The MIB-ADI method maintains second-order accuracy in space for complex geometries in two [45] and three [39] dimensions, including for variable coefficients [25], but can only attain first-order accuracy in time. A multiscale IIM-ADI method has been introduced in [28] for general interface problems, which is able to fulfill second-order accuracy for both spatial and temporal discretization. The GFM has also been combined with the ADI to solve parabolic interface problems [26].

Geometric multigrid methods have also been explored for solving PDEs with complicated geometries. Typically, multigrid interpolation and restriction operators need to be modified near the interfaces and boundaries, and such modifications have to be specially designed based on the particular numerical procedure for treating the boundary or interface conditions. In particular, a new multigrid method has to be developed. Nevertheless, such an effort is worthwhile, because the geometric multigrid methods can guarantee an $O(N)$ efficiency in real computations. The multigrid method for the IIM has been constructed by Adams and Li in [1], which preserves the maximum principle. In [38], the multigrid method has been applied to the boundary-condition-capturing method for irregular boundary problems. The use of a multigrid method in combination with ghost values for interface and boundary treatments has been studied in [12] and [13], respectively, for elliptic and nonlinear diffusion problems. A new multigrid solver has been introduced in [33] for cell-centered finite volume discretizations of 3D anisotropic diffusion equations with discontinuous coefficients. In [20], a multigrid scheme has been constructed for an augmented MIB (AMIB) method to solve parabolic interface problems. In the augmented formulation, one just needs to invert a usual finite difference matrix so that the standard multigrid procedure can be utilized in the AMIB method. This considerably simplifies the interpolation and restriction process for the coarse grid operators in the multigrid method [20]. We finally note that the aforementioned ADI and multigrid methods can at most deliver second order accuracy in space.

The numerical method to be developed in this study is based on the matched interface and boundary (MIB) method, which was originally developed as a high-order method for solving Maxwell interface [46] and elliptic interface [49] problems. In [48,43,47], the MIB scheme has been developed into a systematic approach for handling various general boundary conditions in arbitrarily high-order central schemes over cubic domains. Two recent advancements of the MIB method will be taken advantage of. First, an AMIB method has been formulated in [16], which introduces Cartesian derivative jumps as additional unknowns to correct the central difference approximation. With an enlarged linear system, the discrete Laplacian is the same as that obtained by usual finite difference discretization, and thus can be inverted by the FFT algorithm. Consequently, the AMIB method can not only achieve fourth-order accuracy in solving curved elliptic interface problems, but also attains the FFT

$O(N \log N)$ efficiency [19]. In solving elliptic boundary value problems over a cubic domain, the AMIB method [18,17] can handle any boundary condition, is at least fourth-order accurate in two or three dimensions, and enjoys the FFT efficiency. A multigrid based AMIB method has also been developed for solving parabolic interface problem [20]. Second, a ray-casting AMIB method has been constructed in [35] for solving elliptic boundary value problems over irregular domains. By generating fictitious values in the normal direction or ray-casting direction, the AMIB method guarantees at least a fourth order of accuracy in treating Dirichlet, Neumann, and Robin conditions and their mixed combinations while maintaining the FFT efficiency.

In this work, a new ray-casting AMIB method will be developed for solving the immersed parabolic problem, which consists of the heat equation (1) and initial and boundary conditions given in Eqs. (5), (3), and (6). In combination with the Crank-Nicolson time integration, the new method will be second-order accurate in time and unconditionally stable. Besides the FFT algorithm, further acceleration using the LU decomposition algorithm will be explored for long time integration. The AMIB method can achieve a fourth order of accuracy in space and is flexible in handling a variety of boundary conditions imposed on irregularly-shaped Γ.

3 Theory and Algorithm

The proposed AMIB method will be presented in detail for solving immersed parabolic problems in two dimensions in subsection 3.1 - 3.4. Its extension to three dimensions can be easily obtained by a tensor product and is briefly mentioned in subsection 3.5. A computational complexity analysis of the AMIB method is then conducted in subsection 3.6.

In two dimensions, a uniform mesh spacing, h, is assumed to partition the rectangular domain $D = [a, b] \times [c, d]$ into n_x and n_y equally spaced intervals in the x- and y- directions, respectively. That is, $n_x = (b - a)/h$ and $n_y = (d - c)/h$. In the time direction, a constant time step, Δt, is used to partition a time interval $[0, T]$ into n_t equally spaced intervals so that $n_t = T/\Delta t$. The numerical approximation to the exact solution $u(t_n, x_i, y_j)$ is denoted by $u^n_{i,j}$ with $t_n = n\Delta t$ for $0 = t_0 < t_1 < ... < t_{n_t-1} < t_{n_t} = T$, $x_i = a + ih$ for $a = x_0 < x_1 < ... < x_{n_x-1} < x_{n_x} = b$, and $y_j = c + jh$ for $c = y_0 < y_1 < ... < y_{n_y-1} < y_{n_y} = d$. Similar notations are used for the solution $u(t_n, x_i, y_j, z_k)$ in three dimensions where $z_k = e + kh$ for $e = z_0 < z_1 < ... < z_{n_z-1} < z_{n_z} = f$.

3.1 Temporal Discretization

In the time direction, the standard Crank-Nicolson (CN) scheme is adopted to discretize eqn. (1), yielding a semi-discretized finite difference (FD) equation

$$\frac{u^{n+1}_{i,j} - u^n_{i,j}}{\Delta t} = \frac{\beta}{2}\left(\Delta u^n_{i,j} + \Delta u^{n+1}_{i,j}\right) + \frac{1}{2}\left(f^n_{i,j} + f^{n+1}_{i,j}\right). \tag{7}$$

Reorganizing the terms of eqn. (7) results in

$$(\Delta - \kappa I)\, u_{i,j}^{n+1} = -(\Delta + \kappa I)\, u_{i,j}^{n} - \frac{1}{\beta}\left(f_{i,j}^{n} + f_{i,j}^{n+1}\right), \tag{8}$$

where $\kappa = \frac{2}{\beta \Delta t}$ and I is an identity operator. The two differential operators in eqn. (8), $\Delta - \kappa I$ and $\Delta + \kappa I$, need to be approximated by appropriate FDs.

In order to obtain fourth-order convergence in space, a 5-point central FD and its corrections are proposed to approximate the two differential operators. This central FD and its corrections are applied to different types of mesh points depending on their relative positions to Γ. As a consequence, the resulting AMIB method is a high-order (second-order in time and fourth-order in space, i.e., $O(\Delta t^2 + h^4)$) method for solving immersed parabolic problems in two and three dimensions.

3.2 A Fourth-Order FD and Its Corrections in Spatial Discretization

In this subsection, a fourth-order FD and its corrections are utilized to approximate the two differential operators, $\Delta - \kappa I$ and $\Delta + \kappa I$, at different types of mesh points in each time step. Without causing ambiguity, the superscript n for time steps is suppressed from notations in this subsection. It will be placed back in subsection 3.3 when the fully discretized method is described.

The ray-casting AMIB method for irregular boundary problems consists of three steps [35]. First, some fictitious values will be generated by satisfying the Robin boundary condition. Second, the fourth order central differences will be corrected by using some jump values on Γ. Finally, the jump values are reconstructed based on fictitious values. The ray-casting AMIB method is further improved in this study. In particular, the finite difference corrections for various corner point cases will be studied, so that the present ray-casting AMIB method becomes more robust.

3.2.1 Fictitious Values by the Ray-Casting Scheme and Classification of Interface Points

Consider an irregular boundary or immersed interface Γ, which can be prescribed in parametric form or is determined by a zero level set $\varphi(x, y) = 0$. Here $\varphi(x, y) < 0$ indicates the point inside the domain Ω, while $\varphi(x, y) > 0$ indicates the point outside Ω.

The ray-casting MIB scheme proposed in [35] is adopted to construct *fictitious values* at mesh points which are close to Γ and positioned in the outer subdomain Ω^+. Such fictitious values will be used to correct the standard FD at mesh points near the interface. One can view such fictitious values as the extension of the solution in the inner Ω^- across the interface Γ. The extended values are rigorously determined by the Robin boundary condition (3) imposed at a point on Γ [35]. In addition, two layers of such mesh points are required, as shown by red filled circles in Fig. 2, because we are seeking a FD formula with

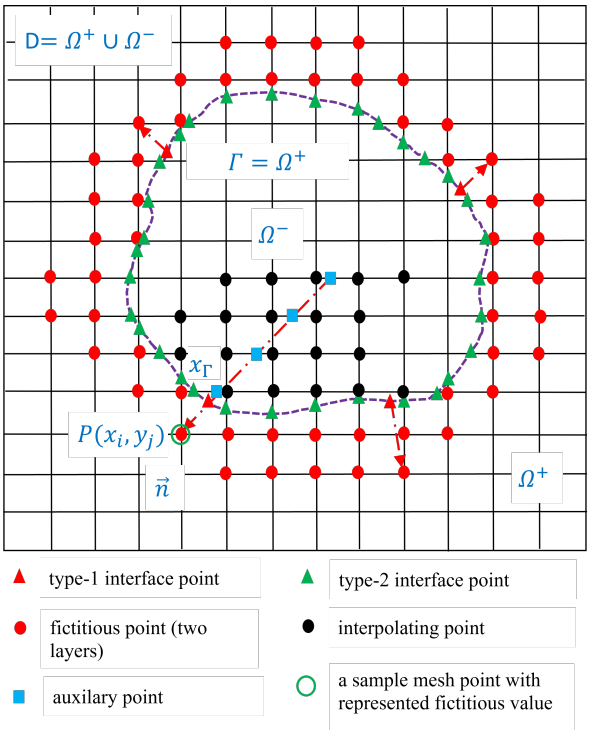


Fig. 2: A demonstration of fictitious values, ray-casting scheme, and classification of interface points. Fictitious values are constructed at two-layer mesh points (red filled circles) outside the boundary Γ. For each of such mesh points, a type-1 interface point (red triangle) is found so that the outer normal direction (red arrow) at this interface point passes through this mesh point. The fictitious value at this mesh point is then represented by values at a set of 20 mesh points (black filled circles) in Ω^- and the interface condition (3) at one type-1 interface point. A second type of interface points at which grid lines cut Γ are also demonstrated by green triangles.

fourth-order accuracy in space. A brief description of the fictitious values and ray-casting scheme is given as follows, while interested readers are directed to subsection 2.2 of Ref. [35] for a detailed description of the ray-casting scheme.

A sample mesh point $P(x_i, y_j)$ (green circle) is shown in Fig. 2 to demonstrate the construction of fictitious value. An interface point (red triangle), x_Γ, on Γ is found so that P is on the normal direction ($\overrightarrow{n}$) of Γ at x_Γ. Extension of $\overrightarrow{n}$ cuts four consecutive grid lines at four *auxiliary points* (blue filled squares) within Ω^-. The interface condition (3) at x_Γ can then be discretized by the desired fictitious value and values at the four auxiliary points (blue filled squares). Due to the fact that these auxiliary points are usually off-grid, their values are normally not readily available and have to be interpolated again by

values at nearby mesh points (black filled circles) within Ω^-. In total, fictitious value at P can then be solved and represented in the form of

$$\hat{u}_{i,j} = \sum_{(x_I, y_J) \in S_{i,j}} W_{I,J} u_{i,j} + W_\Gamma \phi_\Gamma, \tag{9}$$

where $S_{i,j}$ is a set of 20 chosen mesh points within the subdomain Ω^-, and ϕ_Γ represents the boundary data of eqn. (3) evaluated at x_Γ. In fact, eqn. (9) can be obtained for all three (Dirichlet, Neumann, and Robin) boundary conditions and even other mixed forms of boundary conditions. All FD weights involved in eqn. (9) can be calculated via, for example, Fornberg's method [22]. These weights are calculated once, and saved so that updating fictitious values can be effectively accomplished in every time step.

Representing fictitious values at all mesh points demonstrated by red filled circles in Fig. 2 requires eqn. (3) to be evaluated at a set of interface points on Γ (red triangles). These interface points are named *type-1 interface points* in order to distinguish them from another set of interface points described below.

The other set of interface points are those at which Cartesian grid lines cut the interface Γ. They are denoted by green filled triangles in Fig. 2 and named *type-2 interface points*. At type-2 interface points, condition (3) is not used explicitly, while jump-corrected Taylor expansions suggested in [40] are adopted to correct the standard FD when it is applied to approximate the Laplacian operator at mesh points close to Γ so that the desired high order of accuracy can be well maintained. Since these jump corrections are usually unknown, they will be treated as *additional unknowns* and solved together with unknown function values in an augmented system in each time step. More detail will be provided in the next subsection.

One constraint we require for the interface points and their closest mesh points is that, on any grid line, there must exist at least one mesh point between any two consecutive type-2 interface points so that the local change of the interface shape can be captured by nearby mesh points according to their positions inside or outside Γ. Otherwise, a mesh refinement must be conducted.

3.2.2 A Fourth Order FD and Its Corrections

Interior mesh points, (x_i, y_j) for $i = 1, ..., n_x - 1$ and $j = 1, ..., n_y - 1$, will be classified into different categories so that either a standard FD or its corrections will be applied at different mesh points to approximate derivatives involved in the Laplacian operator. Due to the fact that the two second partial derivatives in the Laplacian operator are treated independently in FD formulations, FD approximation to the second order partial derivative of x is chosen to be demonstrated here, while one can follow the same arguments to handle the derivative of y.

Let (x_i, y_j) be an interior mesh point on line $y = y_j$ and consider the construction of the FD to approximate the second derivative of x at this mesh

point. That is, $\frac{\partial^2}{\partial x^2}u(x_i, y_j) \approx \delta_{xx}u_{i,j}$. The mesh point (x_i, y_j) is named a *regular point* when all 5 mesh points involved in the stencil used by the 5-point central FD

$$\delta_{xx}u_{i,j} = \frac{1}{h^2}\left(-\frac{1}{12}u_{i-2,j} + \frac{4}{3}u_{i-1,j} - \frac{5}{2}u_{i,j} + \frac{4}{3}u_{i+1,j} - \frac{1}{12}u_{i+2,j}\right), \quad (10)$$

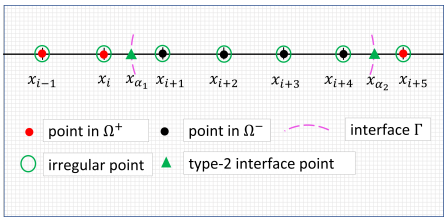
are all positioned on the same side of Γ. In this case, eqn. (10) guarantees a fourth order of accuracy, $O(h^4)$, to approximate the second derivative of x at (x_i, y_j).

In contrast, it is possible that there might be one or multiple mesh points in the stencil are positioned on the opposite of an (type-2) interface point when (x_i, y_j) is close to the interface Γ. In this case, eqn. (10) needs to be corrected by taking into account of the jump values at this interface point, in order to maintain the desired order of accuracy. This mesh point (x_i, y_j) is thereby named an *irregular point*. Furthermore, an extreme case can occur when two *type-2 interface points* are positioned close to each other on the same line such that there exists only one mesh point between them. When this happens, the mesh point in between is named a *corner point* and jump values occurring at both interface points must be taken into account at the same time to correct the FD at this mesh point. Such corrections were first suggested in [40] for central differences. In this study, we will examine all possible cases for the fourth order 5-point FD stencil.

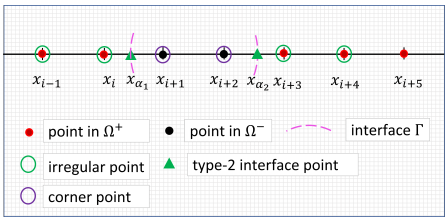
Corrections of the fourth-order FD (10) at irregular and corner points are graphically demonstrated in Fig. 3 for different situations. In all subfigures, x_{α_1} and x_{α_2} are two (type-2) interface points on a grid line $y = y_j$ and denoted by green filled triangles. Regular points are denoted by red and green filled dots, depending on their positions in either outside or inside Γ, respectively. Irregular points are demonstrated by green circles, and corner points are demonstrated by purple circles. In what follows, corrections of the FD (10) are considered in Theorem 1-4 for various scenarios, while one can obtain FDs to approximate the second partial derivative of y in a similar manner.

In Fig. 3a, four mesh points are positioned between the two interface points. In this case, all 7 mesh points are marked as irregular points due to the fact that the 5 points in the stencil of eqn. (10) scatter precisely across one of the two interface points. Correction of eqn. (10) at the 4 left-most irregular points can be obtained by a theorem stated in [40,18,35] using jump values at the left interface point x_{α_1}. This theorem is restated in Theorem 1 using the notations shown in Fig. 3a for easy reference. Correction of eqn. (10) at right 3 mesh points can be obtained in a similar manner by using jump values at the right interface point x_{α_2}.

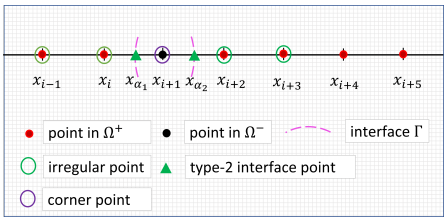
Theorem 1 (A corrected fourth order FD at irregular points [40,18, 35]) Let $x_i < x_{\alpha_1} < x_{i+1}$, $h^- = x_i - x_{\alpha_1}$, and $h^+ = x_{i+1} - x_{\alpha_1}$. Suppose $u \in C^6[x_{i-1}, x_{\alpha_1}] \cap C^6(x_{\alpha_1}, x_{i+2}]$, with derivative extending continuously up to the interface point x_{α_1}. Then the following approximations hold to $O(h^4)$



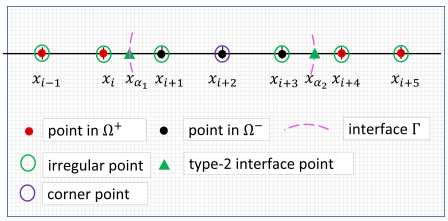
(a) A pair of type-2 interface points with four mesh points in between.



(b) A pair of type-2 interface points with two mesh points in between.



(c) A pair of type-2 interface points with one mesh point in between.



(d) A pair of type-2 interface points with three mesh points in between.

Fig. 3: A demonstration of a pair of type 2 interface points and various number of mesh points in between them, on a horizontal grid line. Mesh points in Ω^+ are denoted by red filled circles, mesh points in Ω^- are denoted by black filled circles, and the two interface points are denoted by green triangles. Irregular points are denoted by green open circles, corner points are denoted by purple open circles, and regular points are those without green or purple open circles.

when $K = 4$:

$$u_{xx}(x_{i-1}) \approx \frac{1}{h^2}\left(-\frac{1}{12}u(x_{i-3}) + \frac{4}{3}u(x_{i-2}) - \frac{5}{2}u(x_{i-1}) + \frac{4}{3}u(x_i) - \frac{1}{12}u(x_{i+1})\right) + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}}, \tag{11}$$

$$u_{xx}(x_i) \approx \frac{1}{h^2}\left(-\frac{1}{12}u(x_{i-2}) + \frac{4}{3}u(x_{i-1}) - \frac{5}{2}u(x_i) + \frac{4}{3}u(x_{i+1}) - \frac{1}{12}u(x_{i+2})\right) - \frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h+h^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}}, \tag{12}$$

$$u_{xx}(x_{i+1}) \approx \frac{1}{h^2}\left(-\frac{1}{12}u(x_{i-1}) + \frac{4}{3}u(x_i) - \frac{5}{2}u(x_{i+1}) + \frac{4}{3}u(x_{i+2}) - \frac{1}{12}u(x_{i+3})\right) + \frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h^-)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} - \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h^- - h)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}}, \tag{13}$$

$$u_{xx}(x_{i+2}) \approx \frac{1}{h^2}\left(-\frac{1}{12}u(x_i) + \frac{4}{3}u(x_{i+1}) - \frac{5}{2}u(x_{i+2}) + \frac{4}{3}u(x_{i+3}) - \frac{1}{12}u(x_{i+4})\right)$$

$$-\frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h^-)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}}, \tag{14}$$

where $\left[u^{(k)}\right]_{x_{\alpha_1}} = \lim_{x\to x_{\alpha_1}^+} u^{(k)}(x) - \lim_{x\to x_{\alpha_1}^-} u^{(k)}(x)$ for $k = 0, ..., 4$ denote the function jump ($k = 0$) or kth derivative jump ($k > 0$), at the interface point x_{α_1}.

Another case which has been studied in [40,18,35] is when two mesh points are positioned between the two interface points, as shown in Fig. 3b. In this case, the 4 red mesh points (in Ω^+) next to the two interface points are all irregular points so that Theorem 1 can be used to correct eqn. (10) at them, while the two mesh points in the middle are corner points, due to the fact that the 5 points in the stencil of eqn. (10) scatter on two sides of both interface points. Therefore, jump values at both interface points must be taken into consideration simultaneously to correct eqn. (10) at these two corner points. The following theorem provides the corrected FDs at these two corner points.

Theorem 2 (A corrected fourth order FD at two corner points shown in Fig. 3b [40,18,35]) Let $x_i < x_{\alpha_1} < x_{i+1} < x_{i+2} < x_{\alpha_2} < x_{i+3}$, $h_1^- = x_i - x_{\alpha_1}$, and $h_2^+ = x_{i+3} - x_{\alpha_2}$. Suppose $u \in C^6[x_{i-1}, x_{\alpha_1}) \cap C^6(x_{\alpha_1}, x_{\alpha_2}) \cap C^6(x_{\alpha_2}, x_{i+4}]$, with derivative extending continuously up to the interface points x_{α_1} and x_{α_2}. Then the following approximations hold to $O(h^4)$ when $K = 4$:

$$\begin{aligned}
u_{xx}(x_{i+1}) \approx\ & \frac{1}{h^2}\left(-\frac{1}{12}u(x_{i-1}) + \frac{4}{3}u(x_i) - \frac{5}{2}u(x_{i+1}) + \frac{4}{3}u(x_{i+2}) - \frac{1}{12}u(x_{i+3})\right) \\
& + \frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} - \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h_1^- - h)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} \\
& + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_2}}, \qquad (15) \\
u_{xx}(x_{i+2}) \approx\ & \frac{1}{h^2}\left(-\frac{1}{12}u(x_i) + \frac{4}{3}u(x_{i+1}) - \frac{5}{2}u(x_{i+2}) + \frac{4}{3}u(x_{i+3}) - \frac{1}{12}u(x_{i+4})\right) \\
& - \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} \\
& - \frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_2}} + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h + h_2^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_2}}. \qquad (16)
\end{aligned}$$

In this work, we continue to consider another extreme case when there exists only one mesh point between the two interface points, as shown in Fig. 3c. The middle mesh point, x_{i+1}, is obviously a corner point, while other mesh points are either irregular or regular. In particular, it should be pointed out that the two mesh points, x_i and x_{i+2}, are actually irregular points due to the fact that, when correcting the FD applied to these two mesh points, only the

middle mesh point, x_{i+1}, is positioned on the Ω^--side of exactly one interface point (either x_{α_1} or x_{α_2}), while all other mesh points are positioned on the Ω^+-side. We thereby propose the following theorem to correct eqn. (10) at the only corner point, x_{i+1}, as follows in this work.

Theorem 3 (A corrected fourth order FD at the corner point shown in Fig. 3c) Let $x_i < x_{\alpha_1} < x_{i+1} < x_{\alpha_2} < x_{i+2}$, $h_1^- = x_i - x_{\alpha_1}$, and $h_2^+ = x_{i+2} - x_{\alpha_2}$. Suppose $u \in C^6[x_{i-2}, x_{\alpha_1}] \cap C^6(x_{\alpha_1}, x_{\alpha_2}] \cap C^6(x_{\alpha_2}, x_{i+4}]$, with derivative extending continuously up to the interface points x_{α_1} and x_{α_2}. Then the following approximations hold to $O(h^4)$ when $K = 4$:

$$\begin{aligned} u_{xx}(x_{i+1}) \approx \frac{1}{h^2}\left(-\frac{1}{12}u(x_{i-1}) + \frac{4}{3}u(x_i) - \frac{5}{2}u(x_{i+1}) + \frac{4}{3}u(x_{i+2}) - \frac{1}{12}u(x_{i+3})\right) \\ -\frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h_1^- - h)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} + \frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} \\ -\frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_2}} + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h_2^+ + h)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_2}}. \end{aligned} \quad (17)$$

Theorems 1-3 provide corrected FDs when either irregular or corner points occur between two consecutive (type-2) interface points on one grid line. The last case taken into consideration is when there are 3 mesh points between the two interface points. It is considered the last because it is actually a mixed case when both irregular and corner points occur between the two interface points, as shown in Fig. 3d. In this case, all other mesh points are irregular so that Theorem 1 can be applied to correct eqn. (10) at them, while the only corner point is x_{i+2}, at which the following theorem is proposed in this work to correct eqn. (10).

Theorem 4 (A corrected fourth order FD at the corner point shown in Fig. 3d) Let $x_i < x_{\alpha_1} < x_{i+1} < x_{i+2} < x_{i+3} < x_{\alpha_2} < x_{i+4}$, $h_1^- = x_i - x_{\alpha_1}$, $h_1^+ = x_{i+1} - x_{\alpha_1}$, $h_2^- = x_{i+3} - x_{\alpha_2}$, $h_2^+ = x_{i+4} - x_{\alpha_2}$. Suppose $u \in C^6[x_{i-2}, x_{\alpha_1}] \cap C^6(x_{\alpha_1}, x_{\alpha_2}] \cap C^6(x_{\alpha_2}, x_{i+4}]$, with derivative extending continuously up to the interface points x_{α_1} and x_{α_2}. Then the following approximations hold to $O(h^4)$ when $K = 4$:

$$\begin{aligned} u_{xx}(x_{i+2}) \approx \frac{1}{h^2}\left(-\frac{1}{12}u(x_i) + \frac{4}{3}u(x_{i+1}) - \frac{5}{2}u(x_{i+2}) + \frac{4}{3}u(x_{i+3}) - \frac{1}{12}u(x_{i+4})\right) \\ -\frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_1}} + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}\left[u^{(k)}\right]_{x_{\alpha_2}}. \end{aligned} \quad (18)$$

3.2.3 Jump Values Reconstruction

Jump values involved in Theorem 1 - 4 usually cannot be obtained directly from the interface condition (3) when, for example, Neumann and Robin

boundary conditions are imposed on Γ. These jump values are treated as *auxiliary variables* and will be solved together with unknown function values, as suggested in [35]. To be more specific, we consider to construct jump values at the left interface point x_{α_1} in Fig. 3a. One can follow the same arguments to construct jump values at other interface points shown in all Fig. 3a - 3d.

Noticing the two left-most mesh points in Fig. 3a are in subdomain Ω^+, fictitious values at them can be obtained by the ray-casting scheme. Utilizing their fictitious values and true values of functions at the other three mesh points in subdomain Ω^-, one can interpolate jump values at x_{α_1} as

$$\left[u^{(k)}\right]_{x_{\alpha_1}} \approx \left(\sum_{l=-1}^{0} W_{i+l}^k \hat{u}_{i+l,j} + \sum_{l=1}^{3} W_{i+l}^k u_{i+l,j}\right) - 0, \tag{19}$$

where zero on the right hand side of eqn. (19) is due to the fact that the function values beyond Ω^- are all assumed to be zeros. In light of the two fictitious values in eqn. (19) can be rewritten in the form of eqn. (9) individually, and substituting eqn. (9) into eqn. (19) yields an equation,

$$\sum_{(x_I y_J)\in S_{i,j}} C_{I,J} u_{I,J} + \left[u^{(k)}\right]_{x_{\alpha_1}} = C_0 \phi_\Gamma, \tag{20}$$

where $C_{I,J}$ is the corresponding weights of function value $u_{I,J}$ in the approximation to jump quantities $\left[u^{(k)}\right]_{x_{\alpha_1}}$ for $k = 0, ..., 4$, and ϕ_Γ is the known boundary data at a type-2 interface point. As the matter of fact, one can obtain formulas similar to eqn. (20) for all type-2 interface points in both x- and y- directions. Moreover, $C_{I,J}$'s are all time invariant so that they are only calculated once and then reused in every time step.

3.3 An Augmented System and Its Solution Accelerated by A Fast Sine Transform

Eqn. (10) and its corrections introduced in Theorems 1-4 can be used to discretize the two differential operators ($\Delta - \kappa I$ and $\Delta + \kappa I$) in eqn. (7) at various types of mesh points. The resulting fully-discretized method yields an *augmented system of equations* to be solved for the unknown function values at all interior mesh points and auxiliary variables for irregular and corner points in every time step. Construction of the augmented system in matrix form will be introduced first, followed by the solution to the system.

We first generalize eqn. (20) for all type-2 interface points in both the x- and y- directions and rewrite it in an equivalent matrix form. Let $N_1 = (n_x - 1) \times (n_y - 1)$ be the total number of interior mesh points and N_2 be the total number of type-2 interface points in both the x- and y- directions. In time step t_n, (unknown) function values at N_1 interior mesh points in the next time step t_{n+1} are formed in a 1D column vector U^{n+1} of dimension $N_1 \times 1$, and the corresponding auxiliary variables at all N_2 type-2 interface

points in the next time step t_{n+1} are formed in another 1D column vector Q^{n+1} of dimension $5N_2 \times 1$ (5 jump values at each type-2 interface point). The matrix form of eqn. (20) at all type-2 interface points is then written as

$$CU^{n+1} + IQ^{n+1} = \Phi^{n+1}, \tag{21}$$

where C is a sparse matrix of dimension $5N_2 \times N_1$ consisting of FD weights for U^{n+1}, I is an identity matrix of dimension $5N_2 \times 5N_2$, and Φ^{n+1} is a column vector of dimension $5N_2 \times 1$ composed of terms after moving the known quantities in time step t_{n+1} to the right-hand side.

The other FD equation is obtained by applying Theorem 1-4 at all interior mesh points, depending on their types, to discretize the two (spatial) differential operators in the semi-discretized eqn. (8). The resulting equation in the matrix form is given by

$$A^- U^{n+1} + BQ^{n+1} = -(A^+ U^n + BQ^n) - F^{n+\frac{1}{2}}, \tag{22}$$

where matrix B is a sparse matrix of dimension $N_1 \times 5N_2$ composed of coefficients from correction terms, and $F^{n+\frac{1}{2}}$ is a column vector of dimension $N_1 \times 1$ composed of source terms on the right-hand side of eqn. (8). The FD matrices A^- and A^+ consist of coefficients from discretizing the differential operators $\Delta - \kappa$ and $\Delta + \kappa$ using the standard fourth order central FD (10), as if no interfaces or boundaries were involved. Note that a zero Dirichlet boundary condition (6) is used on ∂D, and the solution values near ∂D can all be assumed to be zero. Thus, along each x or y grid line, a symmetric, diagonally dominant, and pentadiagonal matrix can be formed for the FD discretization. For the 2D operators $\Delta - \kappa$ and $\Delta + \kappa$, the corresponding matrices A^- and A^+ are generated using the tensor product, so that these block-structured matrices are symmetric and diagonally dominant.

Coupling eqn. (21) - (22) and denoting $R^n = -(A^+ U^n + BQ^n) - F^{n+\frac{1}{2}}$ yields the desired augmented system,

$$\begin{pmatrix} A^- & B \\ C & I \end{pmatrix} \begin{pmatrix} U^{n+1} \\ Q^{n+1} \end{pmatrix} = \begin{pmatrix} R^n \\ \Phi^{n+1} \end{pmatrix}, \tag{23}$$

to be solved in time step t_n. In eqn. (23), matrix B and C are both sparse and time invariant. They are constructed in sparse forms once, and then reused in all time steps. On the other hand, matrix A^- and A^+ are also sparse and time invariant but they will not be constructed explicitly in our computation. Taking advantage of the fact that the diffusion coefficient, β, is a constant and the anti-symmetric property can be trivially achieved near ∂D due to zero Dirichlet boundary condition on ∂D and zero function values in Ω^+, inversion of matrix A^- and multiplication by A^+ can alternatively be carried out by 2D Fast Sine Transform (FST) very effectively. The following two algorithms are modified from the procedures introduced in [16,19,35] to solve the augmented system (23) as follows.

Matrix-vector multiplication of $A^+ U^n$ on the right-hand side of eqn. (23) is obtained by substituting $\mathbf{U} = U^n$ and $\kappa = \frac{2}{\beta \Delta t}$ in Algorithm 1. Given obtained

A^+U^n, R^n can be easily calculated via one direct matrix-vector multiplication (BQ^n) and two vector-vector additions.

Algorithm 1 A 2D FST for the matrix-vector product of $(A+\kappa I)\mathbf{U}$

Input. A constant matrix A resulting from applying the fourth order central FD (10) at $(I-1)\times(J-1)$ interior mesh points, a constant κ, and a known 1D vector $\mathbf{U}$ of dimension $(I-1)(J-1)\times 1$.

Output. The desired product is saved in a 1D vector $\mathbf{V}$ of dimension $(I-1)(J-1)\times 1$.

Step 1. Reshape $\mathbf{U}$ to an equivalent 2D array $\left[u_{l,m}\right]$ for $l=1,...,I-1$ and $m=1,...,J-1$. Compute the Sine transform of $u_{l,m}$ via the 2D fast Sine transform

$$\hat{u}_{p,q}=\frac{4}{IJ}\sum_{l=1}^{I-1}\sum_{m=1}^{J-1}u_{l,m}\sin\left(\frac{pl\pi}{I}\right)\sin\left(\frac{qm\pi}{J}\right),$$

for $p=1,...,I-1,\ q=1,...,J-1$.

Step 2. Compute

$$\lambda_p^x=-\frac{1}{3h_x^2}\left(\cos\left(\frac{p\pi}{I}\right)-1\right)\left(\cos\left(\frac{p\pi}{I}\right)-7\right),$$
$$\lambda_q^y=-\frac{1}{3h_y^2}\left(\cos\left(\frac{q\pi}{J}\right)-1\right)\left(\cos\left(\frac{q\pi}{J}\right)-7\right),$$

and

$$\hat{v}_{p,q}=(\lambda_p^x+\lambda_q^y+\kappa)\hat{u}_{p,q},$$

for $p=1,...,I-1,\ q=1,...,J-1$.

Step 3. Compute $v_{l,m}$ via the 2D inverse fast Sine transform (IFST)

$$v_{l,m}=\sum_{p=1}^{I-1}\sum_{q=1}^{J-1}\hat{v}_{p,q}\sin\left(\frac{pl\pi}{I}\right)\sin\left(\frac{qm\pi}{J}\right),$$

for $l=1,...,I-1,\ m=1,...,J-1$. The resulting 2D array $\left[v_{l,m}\right]$ is then packed to a 1D vector $\mathbf{V}$ for the product of $(A+\kappa I)\mathbf{U}$.

After R^n is obtained, a Schur complement method [16,19,35] is utilized to eliminate U^{n+1} from the augmented system (23), yielding a linear system of dimension $5N_2\times 5N_2$ for Q^{n+1},

$$\left(I-C\left(A^-\right)^{-1}B\right)Q^{n+1}=\Phi^{n+1}-C\left(A^-\right)^{-1}R^n. \tag{24}$$

to be solved for auxiliary variables in vector Q^{n+1} first. To this end, matrix-vector multiplication of $\left(A^-\right)^{-1}R^n$ on the right-hand side of eqn. (24) is obtained by solving the equivalent linear system of equations, $A^-\mathbf{x}=R^n$, using Algorithm 2 with $\kappa=\frac{2}{\beta\Delta t}$ and $\mathbf{b}=R^n$. Then the right-hand side of eqn. (24)

Algorithm 2 A 2D FST for solving the linear system of $(A - \kappa I)\mathbf{x} = \mathbf{b}$

Input. A constant matrix A resulting from applying the fourth order central FD (10) at $(I-1) \times (J-1)$ interior mesh points, a constant κ, and a known 1D vector $\mathbf{b}$ of dimension $(I-1)(J-1) \times 1$.

Output. The unknown vector $\mathbf{x}$ of dimension $(I-1)(J-1) \times 1$.

Step 1. Reshape $\mathbf{b}$ to an equivalent 2D array $\left[b_{l,m}\right]$ for $l = 1, ..., I-1$ and $m = 1, ..., J-1$. Compute the Sine transform of $b_{l,m}$ via the 2D fast Sine transform

$$\hat{b}_{p,q} = \frac{4}{IJ} \sum_{l=1}^{I-1} \sum_{m=1}^{J-1} b_{l,m} \sin\left(\frac{pl\pi}{I}\right) \sin\left(\frac{qm\pi}{J}\right),$$

for $p = 1, ..., I-1,\ q = 1, ..., J-1$.

Step 2. Compute

$$\lambda_p^x = -\frac{1}{3h_x^2}\left(\cos\left(\frac{p\pi}{I}\right) - 1\right)\left(\cos\left(\frac{p\pi}{I}\right) - 7\right),$$
$$\lambda_q^y = -\frac{1}{3h_y^2}\left(\cos\left(\frac{q\pi}{J}\right) - 1\right)\left(\cos\left(\frac{q\pi}{J}\right) - 7\right),$$

and

$$\hat{x}_{p,q} = \frac{\hat{b}_{p,q}}{\lambda_p^x + \lambda_q^y - \kappa},$$

for $p = 1, ..., I-1,\ q = 1, ..., J-1$.

Step 3. Compute $x_{l,m}$ via the 2D inverse fast Sine transform (IFST)

$$x_{l,m} = \sum_{p=1}^{I-1} \sum_{q=1}^{J-1} \hat{x}_{p,q} \sin\left(\frac{pl\pi}{I}\right) \sin\left(\frac{qm\pi}{J}\right),$$

for $l = 1, ..., I-1,\ m = 1, ..., J-1$. The resulting 2D array $\left[x_{l,m}\right]$ is then packed to a 1D vector $\mathbf{x}$ for the solution of the linear system.

can be obtained by one additional matrix-vector multiplication ($C\mathbf{x}$) and one additional vector-vector subtraction.

To further improve the efficiency of the proposed AMIB method, the following two options will be explored to solve eqn. (24) for Q^{n+1}:

Option 1. (LU Decomposition approach (**LU**)) $(A^-)^{-1} B$ on the left-hand side of eqn. (24) can be obtained by repeatedly using Algorithm 2 for $\kappa = \frac{2}{\beta \Delta t}$ and $\mathbf{b}$ = each column of matrix B. The coefficient matrix, $I - C\,(A^-)^{-1} B$, is then constructed by one more direct matrix-vector multiplication and one more matrix-matrix subtraction. After that, a procedure of LU decomposition is conducted to factor the resulting coefficient matrix into two triangular matrices, L and U. The two matrices, L and U, are saved and reused so that eqn.

(24) can be easily solved by a procedure of backward and forward substitutions in every time step.

Option 2. (Biconjugate Gradient approach (**BCG**)) The second option is the same as that proposed in [35]. In this option, a BCG iteration is conducted in each time step for solving Q^{n+1} based on known Q^n and U^n values. Without generating the matrix explicitly, the matrix vector multiplication involved in the left-hand side of eqn. (24) needs to be carried out in each BCG iteration for a vector Q. In particular, the left-hand side of eqn. (24) is rewritten as $IQ - C\left(A^-\right)^{-1} BQ$, in which BQ is obtained by direct matrix-vector multiplication, $\left(A^-\right)^{-1}(BQ)$ is again carried out by using Algorithm 2 for $\kappa = \frac{2}{\beta \Delta t}$ and $\mathbf{b} = BQ$, and $IQ - C\left(A^-\right)^{-1} BQ$ is obtained by additional direct matrix-vector manipulation and vector-vector subtraction. The transpose, $\left(I - C\left(A^-\right)^{-1} B\right)^T Q$, can be completed by following the same strategy on $IQ - B^T\left(A^-\right)^{-1} C^T Q$. A BCG iteration can then be conducted by using Q^n as the initial guess for Q^{n+1} until either a maximal iteration number (for instance, 500) or an error tolerance (for instance, 10^{-13}) is achieved.

Two options are considered to solve eqn. (24) because we anticipate they are suitable to be used in different scenarios. In long-term simulations when numerous time steps are needed to evolve from the initial time to the final time, we expect the LU approach to be more efficient due to the fact that the most time-consuming calculations, factoring the coefficient matrix into L and U, only takes place once, while the procedure of backward and forward substitutions is far more efficient when compared to the BCG approach in each time step. On the other hand, the BCG approach is believed to be more efficient for short-term simulations when not many time steps are required. Numerical examples are used to verify our predictions in section 4.

After the auxiliary vector Q^{n+1} is solved from eqn. (24), U^{n+1} is then solved from

$$A^- U^{n+1} = R^n - BQ^{n+1} \tag{25}$$

by using Algorithm 2 again. At the end of time step t_n, U^{n+1} returned by above procedures can be contaminated by round-off errors. In order to prevent these errors from being carried over to the next time step, a sweep of values, $u_{i,j} = 0$ for all $(x_i, y_j) \in \Omega^+$ is carried out at the end of each time step.

3.4 Additional Numerical Treatments, Limitation, and Future Improvements

A couple of additional numerical treatments were also taken into consideration, implemented and tested in this work. Even though we did not observe notable differences in the results of tested examples after applying these treatments, we expect they could be useful when more complicated examples are considered. These treatments are listed in below to address possible questions from readers.

One such treatment is how to set the initial values for the auxiliary vector Q^0 at the initial time $t = 0$. The most straightforward way is obviously by using the initial condition $u_0(\mathbf{x})$. This assumes that the initial function is given and one can take its derivatives to calculate derivative jumps at various type-2 interface points for Q^0. In contrast, another way to obtain Q^0 is to use eqn. (21). That is, $Q^0 = \Phi^0 - CU^0$. In this second way, function expression and analytical differentiation are not needed. One can calculate Q^0 simply based on discrete initial values U^0. In fact, we have tested setting Q^0 in both ways for all 2D examples in section 4. Obtained results agree up to the seventh decimal place (single precision) in all tested 2D examples.

The other additional treatment is for calculating $\hat{v}_{k,i}$ and $\hat{x}_{k,i}$ in Step 2 of Algorithm 1 - 2. One can see that the three values, $\kappa = \frac{2}{\beta\Delta t}$ and the two λ's, can differ by an order of several magnitudes when, for instance, $\Delta t \gg h$. It is possible that significant numerical cancellation could occur in the sum of these three values in this case. Thus, the *Kahan summation algorithm* (https://en.wikipedia.org/wiki/Kahan_summation_algorithm) was implemented in both algorithms to reduce possible numerical errors involved in the summation. However, after carefully comparing the results obtained with and without the Kahan summation algorithm, no notable differences were actually observed in all tested 2D examples.

The major limitation in the current version of the AMIB method when solving 2D problems is the fact that the involved ray-casting scheme requires a set of 20 mesh points to be found in Ω^- to represent the fictitious value at one mesh point. Even though one can follow the procedure suggested in [35] to have a better chance to find sufficient mesh points in Ω^-, it is still possible that no sufficient mesh points can be found in Ω^- when Γ's shape changes dramatically. However, this limitation may be lifted, for example, by allowing fictitious values to be calculated in a nested manner so that one fictitious value can be obtained by repeatedly reusing fictitious values at other nearby mesh points. It requires a more complicated procedure to implement the ray-casting scheme, which is beyond the scope of this paper. Work in this direction will be explored in the future.

3.5 Extension of the AMIB method to Three Dimensions

The AMIB method introduced in proceeding subsection 3.1 - 3.4 can be extended to solve 3D immersed parabolic problems by two major modifications. One modification is to extend the ray-casting scheme to three dimensions, which has been considered in the original work [35]. In three dimensions, to determine one fictitious value, the ray-casting approximation is again conducted along the normal direction. This normal line will intersect with four 2D planes at four auxiliary points. Each auxiliary point is interpolated by 16 mesh points on the corresponding 2D plane, so that each fictitious value is eventually represented by a linear combination of function values at 64 mesh points and one boundary data at a type-1 interface point. The other major modification

is to correct Algorithm 1 - 2 to use the 3D FST. It is fairly straightforward, because the original algorithms are formulated using the tensor product. The two algorithms for solving 3D problems are given in Algorithm 3 - 4.

Algorithm 3 A 3D FST for the matrix-vector product of $(A + \kappa I)\mathbf{U}$

Input. A constant matrix A resulting from applying the fourth order central FD (10) at $(I-1) \times (J-1) \times (K-1)$ interior mesh points, a constant κ, and a known 1D vector $\mathbf{U}$ of dimension $(I-1)(J-1)(K-1) \times 1$.

Output. The desired product is saved in a 1D vector $\mathbf{V}$ of dimension $(I-1)(J-1)(K-1) \times 1$.

Step 1. Reshape $\mathbf{U}$ to an equivalent 3D array $\left[u_{l,m,n}\right]$ for $l = 1, ..., I-1$, $m = 1, ..., J-1$, and $n = 1, ..., K-1$. Compute the Sine transform of $u_{l,m}$ via the 3D fast Sine transform

$$\hat{u}_{p,q,r} = \frac{8}{IJK} \sum_{l=1}^{I-1} \sum_{m=1}^{J-1} \sum_{n=1}^{K-1} u_{l,m,n} \sin\left(\frac{pl\pi}{I}\right) \sin\left(\frac{qm\pi}{J}\right) \sin\left(\frac{rn\pi}{K}\right),$$

for $p = 1, ..., I-1$, $q = 1, ..., J-1$, and $r = 1, ..., K-1$.

Step 2. Compute

$$\lambda_p^x = -\frac{1}{3h_x^2}\left(\cos\left(\frac{p\pi}{I}\right) - 1\right)\left(\cos\left(\frac{p\pi}{I}\right) - 7\right),$$
$$\lambda_q^y = -\frac{1}{3h_y^2}\left(\cos\left(\frac{q\pi}{J}\right) - 1\right)\left(\cos\left(\frac{q\pi}{J}\right) - 7\right),$$
$$\lambda_r^z = -\frac{1}{3h_z^2}\left(\cos\left(\frac{r\pi}{K}\right) - 1\right)\left(\cos\left(\frac{r\pi}{K}\right) - 7\right),$$

and

$$\hat{v}_{p,q,r} = (\lambda_p^x + \lambda_q^y + \lambda_r^z + \kappa)\hat{u}_{p,q,r},$$

for $p = 1, ..., I-1$, $q = 1, ..., J-1$, and $r = 1, ..., K-1$.

Step 3. Compute $v_{l,m,n}$ via the 3D inverse fast Sine transform (IFST)

$$v_{l,m,n} = \sum_{p=1}^{I-1} \sum_{q=1}^{J-1} \sum_{r=1}^{K-1} \hat{v}_{p,q,r} \sin\left(\frac{pl\pi}{I}\right) \sin\left(\frac{qm\pi}{J}\right) \sin\left(\frac{rn\pi}{K}\right)$$

for $l = 1, ..., I-1$, $m = 1, ..., J-1$, and $n = 1, ..., K-1$. The resulting 3D array $\left[v_{l,m,n}\right]$ is then packed to a 1D vector $\mathbf{V}$ for the product of $(A + \kappa I)\mathbf{U}$.

3.6 Computational Complexity Analysis

This last subsection is dedicated to analyze computational complexity of the proposed method. To this end, floating point operations (FLOPS) are used

Algorithm 4 A 3D FST for solving the linear system of $(A-\kappa I)\mathbf{x}=\mathbf{b}$

Input. A constant matrix A resulting from applying the fourth order central FD (10) at $(I-1)\times(J-1)\times(K-1)$ interior mesh points, a constant κ, and a known 1D vector $\mathbf{b}$ of dimension $(I-1)(J-1)(K-1)\times 1$.

Output. The unknown vector $\mathbf{x}$ of dimension $(I-1)(J-1)(K-1)\times 1$.

Step 1. Reshape $\mathbf{b}$ to an equivalent 3D array $\left[b_{l,m,n}\right]$ for $l=1,...,I-1$, $m=1,...,J-1$, and $n=1,...,K-1$. Compute the Sine transform of $b_{l,m,n}$ via the 3D fast Sine transform

$$\hat{b}_{p,q,r}=\frac{8}{IJK}\sum_{l=1}^{I-1}\sum_{m=1}^{J-1}\sum_{n=1}^{K-1}b_{l,m,n}\sin\left(\frac{pl\pi}{I}\right)\sin\left(\frac{qm\pi}{J}\right)\sin\left(\frac{rn\pi}{K}\right),$$

for $p=1,...,I-1$, $q=1,...,J-1$, and $r=1,...,K-1$.

Step 2. Compute

$$\lambda_p^x=-\frac{1}{3h_x^2}\left(\cos\left(\frac{p\pi}{I}\right)-1\right)\left(\cos\left(\frac{p\pi}{I}\right)-7\right),$$
$$\lambda_q^y=-\frac{1}{3h_y^2}\left(\cos\left(\frac{q\pi}{J}\right)-1\right)\left(\cos\left(\frac{q\pi}{J}\right)-7\right),$$
$$\lambda_r^z=-\frac{1}{3h_z^2}\left(\cos\left(\frac{r\pi}{K}\right)-1\right)\left(\cos\left(\frac{r\pi}{K}\right)-7\right),$$

and

$$\hat{x}_{p,q,r}=\frac{\hat{b}_{p,q,r}}{\lambda_p^x+\lambda_q^y+\lambda_r^z-\kappa},$$

for $p=1,...,I-1$, $q=1,...,J-1$, and $r=1,...,K-1$.

Step 3. Compute $x_{l,m}$ via the 3D inverse fast Sine transform (IFST)

$$x_{l,m,n}=\sum_{p=1}^{I-1}\sum_{q=1}^{J-1}\sum_{r=1}^{K-1}\hat{x}_{p,q,r}\sin\left(\frac{pl\pi}{I}\right)\sin\left(\frac{qm\pi}{J}\right)\sin\left(\frac{rn\pi}{K}\right),$$

for $l=1,...,I-1$, $m=1,...,J-1$, and $n=1,...,K-1$. The resulting 3D array $\left[x_{l,m,n}\right]$ is then packed to a 1D vector $\mathbf{x}$ for the solution of the linear system.

to facilitate the general discussion in this subsection. We will consider the full details for 2D problems only. For the sake of simplicity, we assume the number of mesh points per direction is the same and denote it by N. That is, $N=n_x(=n_y)$ in two dimensions so that $N_1=O(N^2)$. Assume the dimension of Q^n to be $M=5N_2$. Recall that N_2 represents the total number of type-2 interface points. So, N_2 is one-dimensionally smaller than N_1. Therefore, we have $M=O(N)$. We also denote the number of involved time steps by N_t.

We first discuss the number of FLOPS for the LU approach. The computation cost comes from two parts, i.e., solving linear systems in implicit time integration and the pre-computation of L and U. In each time step, eqn. (24) is

solved by backward and forward substitutions with cost about $O(M) = O(N)$ FLOPS. On the other hand, calling either of the two algorithms, Algorithm 1 - 2, costs $O(N^2 \log N)$ FLOPS. In fact, Algorithm 1 is called once and Algorithm 2 is called twice in each time step when the LU approach is used. Other direct algebraic operations, such as matrix-vector multiplications by matrix B and C, are negligible due to the fact that they are both sparse with much smaller dimension than N^2. Since the cost of backward and forward substitutions is one order of magnitude smaller, it will be dropped in our analysis. In total, we conclude the order of FLOPS in each time step is $O(N^2 \log N)$. Thus, the total implicit time integration needs $O(N_t N^2 \log N)$ FLOPS when the LU approach is used. For the pre-computation of L and U, one needs to first generate the matrix $(I - C(A^-)^{-1}B)$ in eqn. (24). Algorithm 2 will be called for M times to compute $(A^-)^{-1}B$ with a cost about $O(MN^2 \log N) = O(N^3 \log N)$. The LU decomposition needs $O(N^3)$ FLOPS, while costs of other minor operations are neglected. Thus, the total pre-computation takes $O(N^3 \log N)$ FLOPS. Therefore, the overall computation of the LU approach would require a complexity on the order of $O(N^3 \log N) + O(N_t N^2 \log N)$. This complexity obviously depends how large N_t is, relative to N. For short-term simulations with small N_t, the first term dominates so that the CPU time will increase slightly when N_t is increased. For a long-term simulation with a much large N_t, the second term dominates. In this case, we expect the counts of FLOPS to increase linearly proportional to N_t.

The BCG approach does not involve much pre-computation. In each time step, the major cost is due to the iterative solution of (24) by the BCG approach. In each BCG iteration, Algorithm 2 will be called a couple of times. The total calls of Algorithm 2 depend on the iteration number consumed in the BCG algorithm. For solving the elliptic boundary value problems over irregular domains, the iteration numbers of the ray-casting AMIB method have been reported for various examples in [35]. In general, the iteration number varies for different problems and boundary conditions. Nevertheless, the iteration number is known to grow slowly with respect to the increment of N. For this reason, the BCG iteration in each time step can be assumed to have a complexity $O(N^2 \log N)$, after ignoring FLOPS costed by minor operations. Considering all time steps, the counts of FLOPS increase linearly proportional to N_t because a similar number of iterations can be assumed in each time step. Therefore, the overall computation of the BCG approach would require a complexity on the order of $O(N_t N^2 \log N)$.

One can follow similar discussion to analyze the complexity in three dimensions. For example, the estimated counts of FLOPS for the BCG approach is $O(N_t N^3 \log N)$. However, the complexity of the LU approach is estimated to be $O(N^6) + (N_t N^3 \log N)$, which could be significantly more expensive than the BCG approach. We thus did not implement the LU approach in 3D. Numerical verification of complexity for some 2D and 3D problems is given in subsection 4.2.

4 Numerical Experiments

In this section, a variety of 2D and 3D numerical examples are studied to validate the proposed ray-casting AMIB method. In order to test the order of convergence, the following two error norms

$$L^2 = \sqrt{\frac{1}{N_{\Omega^-}} \sum_{\mathbf{x}\in\Omega^-} |u(T,\mathbf{x}) - u_h(T,\mathbf{x})|^2}$$
$$L^\infty = \max_{\mathbf{x}\in\Omega^-} |u(T,\mathbf{x}) - u_h(T,\mathbf{x})|$$

are reported at the final time T, where N_{Ω^-} is the number of mesh points inside Γ, u is the exact solution, and u_h is the numerical solution obtained with spatial mesh size h utilized in all Cartesian directions. The order of convergence is calculated by

$$\text{order } = \frac{\log(||E_1||/||E_2||)}{\log(h_1/h_2)}$$

where E_i for $i = 1, 2$ denotes the numerical error obtained on all mesh points after partitioning the computational domain D with uniform mesh size h_i.

Numerical errors, convergence rates and stability of the proposed method are tested on a variety of 2D and 3D examples and reported in subsection 4.1, while computational complexity is discussed using the obtained wall clock times in these examples in subsection 4.2. All tests reported in this section are conducted on a high performance computing cluster equipped with 24 Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz (each CPU is hyper-threaded into 2 cores resulting in 48 processors in total), 128 GB memory, and CentOS 9 operating system. The FORTRAN 90/95 code is compiled with GCC version 4.8.5.

4.1 Numerical Errors, Convergence Rates, and Stability

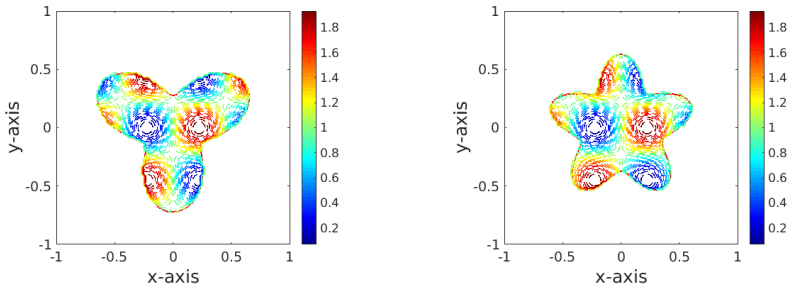
Example 1. The first example is constructed by assuming the exact solution

$$u(t,\mathbf{x}) = \sin(7x)\cos(7y) + \cos(t) \tag{26}$$

defined inside an interface Γ, which is governed by a parametric equation

$$\Gamma : r = 0.5 + 0.22\sin(3\theta) \tag{27}$$

for $0 \le \theta \le 2\pi$. The interface Γ is shaped like a 3-point star, which is plotted together with the exact solution (26) at the initial time $t = 0$ in Fig. 4a. Both the Dirichlet boundary condition employed on Γ and the initial condition of u are obtained by the exact solution (26). The computational domain D is set to be $[-\frac{1}{3}\pi, \frac{1}{3}\pi] \times [-\frac{1}{3}\pi, \frac{1}{3}\pi]$. The diffusion coefficient is fixed, $\beta = 1$, in all spatial and temporal convergence tests except those demonstrated in Table 3



(a) Example 1. (b) Example 2.

Fig. 4: Demonstration of the exact solution and interfaces used in Example 1 - 2.

for studying the impact of β values on the convergence and stability of the method.

The order of spatial convergence is tested first. In these tests, a small enough time increment $\Delta t = 10^{-5}$ is fixed, so that $\Delta t \ll h$. The final time is also fixed, $T = 10^{-1}$, so that the problem is solved in sufficiently many $(10,000)$ time steps. The order of convergence with respect to h is then determined by varying the number of mesh points per direction ranging from 65 to 1025 (h ranges from 3.27E-02 to 2.05E-03). Errors in the final time step and numerically calculated orders are listed in Table 1.

Table 1: Spatial Convergence Tests, Example 1

BCG	L^∞		L^2		Wall Clock (s)
$[N_x,N_y]$	error	order	error	order	
$[65,65]$	6.70872E-05		1.24562E-05		186
$[129,129]$	3.89483E-06	4.10641	5.86270E-07	4.40915	789
$[257,257]$	1.19083E-07	5.03241	3.73684E-08	3.97238	4,589
$[513,513]$	6.53282E-09	4.18886	2.35052E-09	3.99147	28,021
$[1025,1025]$	5.34299E-10	3.61071	1.65825E-10	3.82390	143,816
LU	L^∞		L^2		Wall Clock (s)
$[N_x,N_y]$	error	order	error	order	
$[65,65]$	6.70872E-05		1.24562E-05		83
$[129,129]$	3.89483E-06	4.10641	5.86270E-07	4.40915	433
$[257,257]$	1.19083E-07	5.03241	3.73684E-08	3.97238	1,581
$[513,513]$	6.53271E-09	4.18888	2.35052E-09	3.99147	7,037
$[1025,1025]$	5.34350E-10	3.61055	1.65846E-10	3.82372	34,369

In Table 1, L^∞ and L^2 errors in the final time obtained by the LU approach and the BCG approach with tolerance $= 10^{-13}$ are shown in column two and four, respectively. Their orders are shown in column three and five. One can see that both L^∞ and L^2 errors are close to the fourth order, with only minor perturbations caused by round offs and time step Δt. Errors and orders are shown to five decimal places. One can see that both approaches obtain almost the same (close to double-precision) errors and orders with only very slight differences occurring when $N_x = N_y = 513$ and 1025. As a matter of fact, both approaches yield very close errors and orders in all tested cases conducted in this paper. For simplicity, we thus will only report the errors and orders of the BCG approach from now on. Wall clock time shown in Table 1 clearly reveals that the LU approach is more efficient than the BCG approach in all spatial convergence tests. It is because a sufficiently large number of time steps, 10,000, is used in all spatial convergence tests.

The order of temporal convergence is tested next. In this series of tests, $h \ll \Delta t$ is required to reduce the impact of errors introduced by spatial discretization. To this end, $N_x = N_y = 1025$ ($h \approx$ 2.05E-03) and the final time $T = 1.0$ are fixed, while the number of time advance steps, N_t, varies from 2 to 128 (Δt ranges from 5.0E-01 to 7.81E-03). The BCG approach is utilized only since the number of time steps is relatively small. Obtained results are shown in Table 2.

Table 2: Temporal Convergence Tests, Example 1

N_t	$\frac{\beta \Delta t}{h^2}$	L^∞		L^2		Wall Clock (s)
		error	order	error	order	
2	119523	8.19E-04		4.44E-04		229
4	59762	1.69E-04	2.28	9.18E-05	2.27	349
8	29881	4.18E-05	2.01	2.27E-05	2.01	658
16	14940	1.04E-05	2.00	5.67E-06	2.00	1,432
32	7470	2.61E-06	2.00	1.42E-06	2.00	2,434
64	3735	6.52E-07	2.00	3.54E-07	2.00	4,290
128	1868	1.63E-07	2.00	8.86E-08	2.00	6,935

One can see that the second order of convergence in time is well maintained in all tested cases. In addition, the ratio, $\frac{\beta \Delta t}{h^2}$, is also demonstrated in column two to emphasize the unconditional stability of the AMIB method. This ratio is known as the stability constraint, which must be strictly less than one for any explicit method to converge. In our tests, it is far larger than one in all temporal convergence tests when the AMIB method well maintains convergence at the final time. It clearly demonstrates the unconditional stability of the AMIB method.

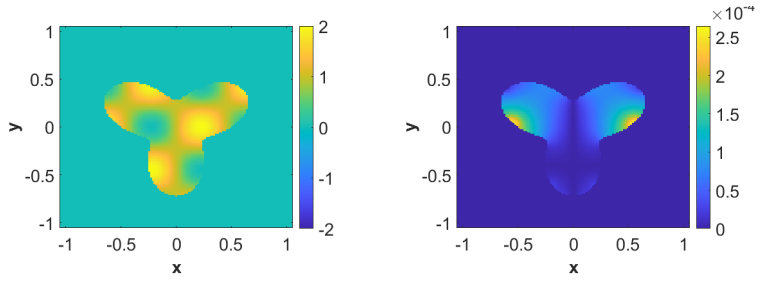
In our previous experiments, the value of β can also affect convergence and stability of the method. To test its impact on the AMIB method, a series of tests was conducted by fixing final time $T = 1.0$, time step $\Delta t = 10^{-3}$, and the number of mesh points per direction $N_x = N_y = 257$, and varying β values ranging from 10^{-2} to 10^{3}. Obtained L^{∞} and L^{2} errors in the final time are shown in Table 3.

Table 3: Tests of the impact of β, Example 1

β	L^{∞}	L^{2}
10^{-2}	7.74E-06	9.84E-07
10^{-1}	1.23E-05	2.92E-06
10^{0}	1.38E-05	3.83E-06
10^{1}	1.37E-05	3.83E-06
10^{2}	1.36E-05	3.83E-06
10^{3}	1.29E-05	3.83E-06

It is found β values do not impact the resulting errors significantly. Errors increase slowly and plateau quickly as β increases. The largest errors are just about 1.8 and 3.9 times of the smallest ones while β changes from 10^{-2} to 10^{5} (7 orders of magnitude). Given the results shown in Table 3, we believe that the performance of the AMIB method is not impacted by β values significantly.

At the end of this example, the numerical solution and corresponding absolute errors at a final time $T = 2\pi(\approx 6.284)$ are shown in Fig. 5 for visualizing the dissipation of the solution and error in subdomain Ω^{-}. In this demo, $\Delta t = 10^{-3}$ and $N_x = N_y = 129$ are used.



(a) Solution (b) Error

Fig. 5: Demonstration of numerical solution and absolute errors at a final time $T = 6.284$ in Example 1.

Example 2. The second example reuses the same exact solution (26) but reconstructs the interface by

$$\Gamma : r = 0.5 + 0.13\sin(5\theta), \tag{28}$$

resulting in a 5-point star interface Γ as shown in Fig. 4b. A more generic Robin boundary condition (3) with $\alpha_\Gamma = \beta_\Gamma = 1.0$ is imposed on Γ, where $\phi(t, \mathbf{x})$ is determined by the exact solution (26) again. Given the changed interface shape and boundary condition, we would like to see whether the orders of convergence are still maintained in this example.

Spatial convergence is tested first again. The same numerical setup used in Example 1 is reused here and the obtained errors and associated spatial convergence order are reported in Table 4. One can see that both L^∞ and L^2 errors are about 1-2 magnitudes larger than those obtained in Example 1. All numerically calculated orders are still reasonably close to the desired order four with only one exception that the order of L^∞ error is 2.97 when $N_x = N_y = 129$. This suggests that the boundary condition imposed on Γ and the shape of Γ indeed have some impacts on the accuracy, and this motivates us to test additional generic boundary conditions and interfaces in the following examples. On the other hand, no significant differences are observed when comparing wall clock time shown in Table 1 and 4 so that it is strongly believed that this method is equally efficient when solving these two examples.

Table 4: Spatial Convergence Tests, Example 2

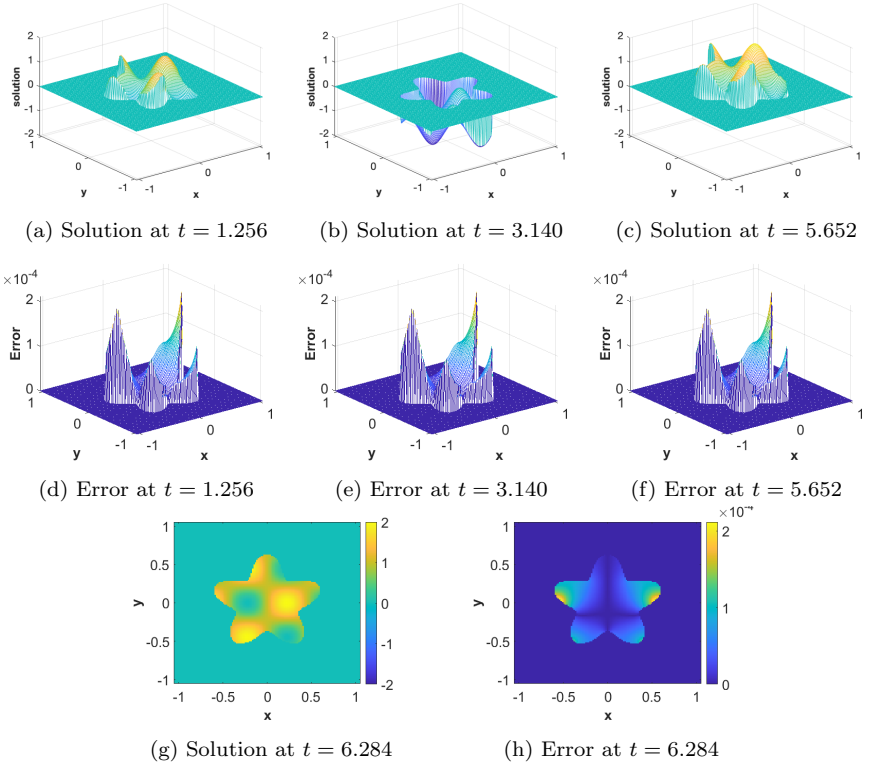
$[N_x, N_y]$	L^∞		L^2		Wall Clock (s)	
	error	order	error	order	BCG	LU
$[65, 65]$	1.56E-03		5.76E-04		161	66
$[129, 129]$	1.99E-04	2.97	5.45E-05	3.40	852	385
$[257, 257]$	8.26E-06	4.59	2.40E-06	4.51	4,244	2,063
$[513, 513]$	3.66E-07	4.50	1.22E-07	4.30	22,036	8,063
$[1025, 1025]$	4.00E-08	3.19	1.04E-08	3.56	102,728	36,506

Temporal convergence tests are conducted in a similar manner and the results are presented in Table 5. The utilized time steps are shown in the second column. They are identical to those used in temporal convergence tests in Example 1. Values of $\frac{\beta \Delta t}{h^2}$ are identical to those shown in Table 2 so that they are removed from Table 5. Noticing all errors are comparable to those shown in Table 2 and all calculated orders of convergence in time are close to two, we conclude that the AMIB method well maintains its unconditional stability and the order of convergence in time on this example as well.

Lastly, numerical solutions and corresponding absolute errors at various time steps obtained by using $\Delta t = 10^{-3}$ and $N_x = N_y = 129$ are shown in

Table 5: Temporal Convergence Tests, Example 2

N_t	Δt	L^∞		L^2		Wall Clock (s)
		error	order	error	order	
2	5.00E-01	4.06E-04		2.00E-04		68
4	2.50E-01	9.86E-05	2.04	4.85E-05	2.05	129
8	1.25E-01	2.45E-05	2.01	1.20E-05	2.01	253
16	6.25E-02	6.14E-06	2.00	3.00E-06	2.00	506
32	3.13E-02	1.56E-06	1.98	7.47E-07	2.01	966
64	1.56E-02	4.13E-07	1.92	1.84E-07	2.02	1,768
128	7.81E-03	1.27E-07	1.70	4.69E-08	1.97	3,094



(a) Solution at $t = 1.256$ (b) Solution at $t = 3.140$ (c) Solution at $t = 5.652$

(d) Error at $t = 1.256$ (e) Error at $t = 3.140$ (f) Error at $t = 5.652$

(g) Solution at $t = 6.284$ (h) Error at $t = 6.284$

Fig. 6: Demonstration of the numerical solution and errors in Example 2.

Fig. 6a - 6f, and those obtained at the final time $T = 2\pi$ are shown in Fig. 6g - 6h. Different plots are used so that one can see how solution and error change over time.

Example 3. We continue to test the performance of the AMIB method on another example with a new exact solution, a differently-shaped Γ, and more generic boundary conditions. In this example, the exact solution is defined by

$$u(t, \mathbf{x}) = \sin(7x)\cos(7y)\cos(t), \tag{29}$$

and $\Gamma = \Gamma_1 \cup \Gamma_2$ is constructed by the union of two circles

$$\begin{aligned} \Gamma_1 &= \{(x, y) | x < 0, (x + 0.2)^2 + y^2 = 0.5^2\}, \\ \Gamma_2 &= \{(x, y) | x > 0, (x - 0.2)^2 + y^2 = 0.5^2\}. \end{aligned} \tag{30}$$

A more complicated mixed (Robin and Neumann) boundary condition is imposed on Γ by

$$\begin{aligned} u + \frac{\partial u}{\partial n} &= (\sin(7x)\cos(7y) + 7\cos(7x)\sin(7y)\hat{n}_x - 7\sin(7x)\cos(7y)\hat{n}_y)\cos(t), \\ &\quad \text{on } \Gamma_1 \\ \frac{\partial u}{\partial n} &= (7\cos(7x)\cos(7y)\hat{n}_x - 7\sin(7x)\sin(7y)\hat{n}_x)\cos(t), \text{ on } \Gamma_2 \end{aligned} \tag{31}$$

where $(\hat{n}_x, \hat{n}_y)$ determines the normal direction. Besides the change of the shape of Γ, another significance of this example is that the boundary condition (31) is time-and-space dependent.

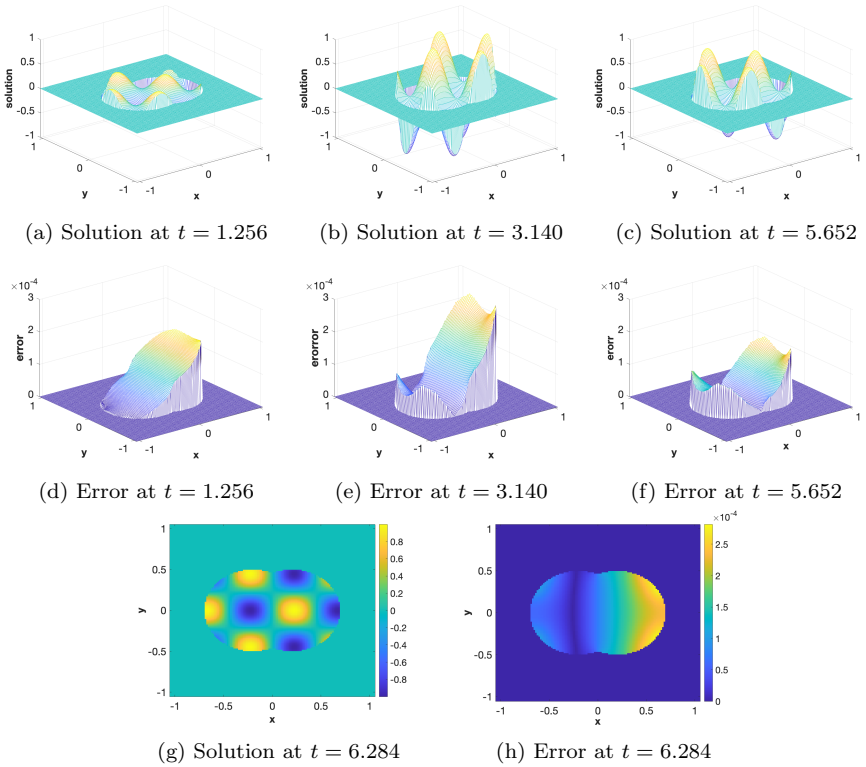
Table 6: Spatial Convergence Tests, Example 3

$[N_x, N_y]$	L^∞		L^2		Wall Clock (s)	
	error	order	error	order	BCG	LU
$[65, 65]$	2.62E-03		9.19E-04		161	66
$[129, 129]$	1.37E-04	4.26	5.45E-05	4.08	903	356
$[257, 257]$	1.80E-05	2.92	5.97E-06	3.19	4,272	1,799
$[513, 513]$	8.57E-07	4.39	2.98E-07	4.33	21,998	6,737
$[1025, 1025]$	4.68E-08	4.19	1.79E-08	4.06	101,497	31,961

We continue using the same numerical setup to test the orders of accuracy in space and time. The resulting errors and orders of convergence are shown in Table 6 and 7, respectively. Once again, one can see that the proposed method is still capable of achieving the fourth order of convergence in space and the second order of convergence in time, despite the fact that the shape of Γ and the boundary condition are quite different from those used in Example 1-2. All tested cases in this example return normally. No divergence was observed. This strongly suggests that the proposed method is indeed unconditionally stable, reliably producing close approximations to the exact solution of the

Table 7: Temporal Convergence Tests, Example 3

N_t	L^∞		L^2		Wall Clock (s)
	error	order	error	order	
2	6.88E-04		3.07E-04		64
4	1.69E-04	2.02	7.57E-05	2.02	131
8	4.22E-05	2.01	1.89E-05	2.00	247
16	1.05E-05	2.01	4.69E-06	2.01	474
32	2.58E-06	2.03	1.15E-06	2.03	920
64	5.98E-07	2.11	2.63E-07	2.13	1,695
128	1.03E-07	2.53	4.32E-08	2.61	3,043



(a) Solution at $t = 1.256$ (b) Solution at $t = 3.140$ (c) Solution at $t = 5.652$

(d) Error at $t = 1.256$ (e) Error at $t = 3.140$ (f) Error at $t = 5.652$

(g) Solution at $t = 6.284$ (h) Error at $t = 6.284$

Fig. 7: Demonstration of the numerical solution and errors in Example 3.

tested problem. Similar plots as those shown in Example 2 for visualizing the solutions and errors at various time steps are given in Fig. 7.

Example 4. One more 2D example is given before our tests in two dimensions are concluded. In this example, exact solution (29) is reused, while an arch-like interface Γ is constructed by two half-circles and two rectangles

$$\begin{aligned} \text{circle 1}: \ & \Gamma_1 = \{(x,y)|x<0, y=\sqrt{0.3^2-x^2}\}, \\ \text{circle 2}: \ & \Gamma_2 = \{(x,y)|x>0, y=\sqrt{0.8^2-x^2}\}, \\ \text{rectangles}: \ & [-0.8,-0.3]\times[-0.8,0] \text{ and } [0.3,0.8]\times[-0.8,0]. \end{aligned} \tag{32}$$

One can refer to Fig 8f - 8g for the shape of Γ used in this example. A mixed boundary condition is again imposed on Γ as

$$\begin{aligned} u+\frac{\partial u}{\partial n} &= (\sin(7x)\cos(7y)+7\cos(7x)\sin(7y)\hat{n}_x - 7\sin(7x)\cos(7y)\hat{n}_y)\cos(t), \\ &\quad \text{on } \Gamma\cap\{(x,y)|x>0\}, \\ \frac{\partial u}{\partial n} &= (7\cos(7x)\cos(7y)\hat{n}_x - 7\sin(7x)\sin(7y)\hat{n}_x)\cos(t), \text{ on } \Gamma_2 \\ &\quad \text{on } \Gamma\cap\{(x,y)|x<0\}. \end{aligned} \tag{33}$$

Table 8: Spatial Convergence Tests, Example 4

$[N_x,N_y]$	L^∞		L^2		Wall Clock (s)	
	error	order	error	order	BCG	LU
$[65,65]$	1.61E-03		5.00E-04		217	159
$[129,129]$	7.53E-05	4.41	2.81E-05	4.16	1,031	1,284
$[257,257]$	4.24E-06	4.15	1.60E-06	4.14	4,359	3,338
$[513,513]$	3.53E-07	3.59	9.77E-08	4.03	23,029	19,041
$[1025,1025]$	3.14E-08	3.49	6.90E-09	3.82	101,030	69,224

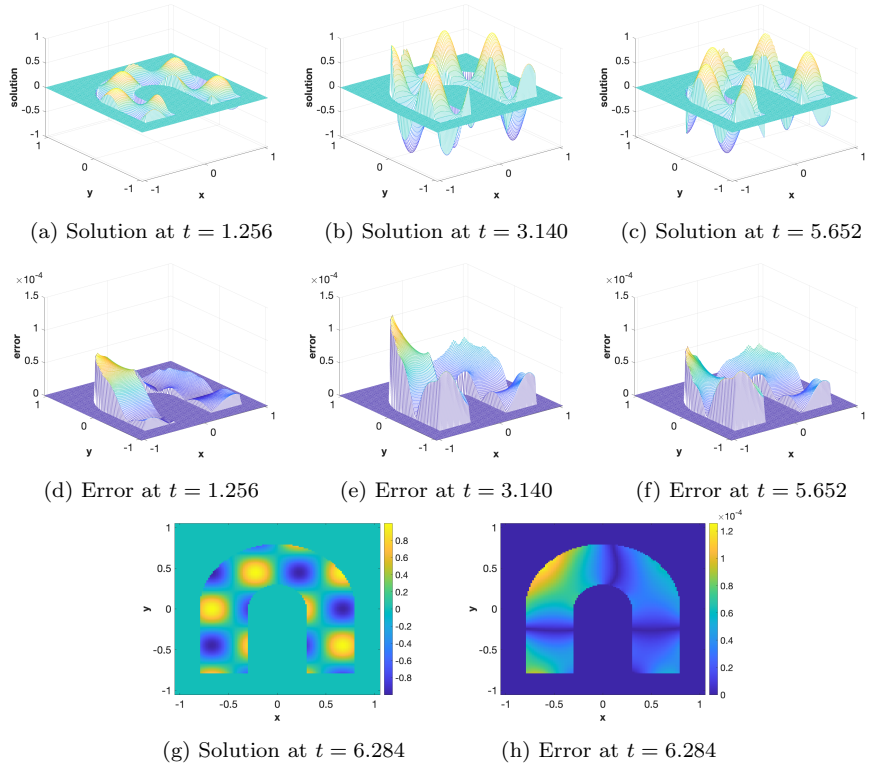
Spatial convergence tests, temporal convergence tests, and visualization of solutions and errors are shown in Table 8, Table 9, and Fig. 8, respectively. The proposed method, as expected, robustly maintains desired convergence orders on this example again. Given our experience on these four examples in two dimensions, we are confident to claim that the first three goals we set in section 1 are attained by the developed AMIB method in two dimensions. We then move forward to test its performance to solve a problem in three dimensions.

Example 5. In this 3D example, the exact solution defined by

$$u(t,x,y,z) = \sin(2x)\cos(2y)\sin(2z)\cos(2t)$$

Table 9: Temporal Convergence Tests, Example 4

N_t	L^∞		L^2		Wall Clock (s)
	error	order	error	order	
2	4.06E-04		2.00E-04		78
4	9.86E-05	2.04	4.85E-05	2.05	141
8	2.45E-05	2.01	1.20E-05	2.01	262
16	6.14E-06	2.00	3.00E-06	2.00	528
32	1.56E-06	1.98	7.47E-07	2.01	955
64	4.13E-07	1.92	1.84E-07	2.02	1,739
128	1.27E-07	1.70	4.69E-08	1.97	3,073



(a) Solution at $t = 1.256$ (b) Solution at $t = 3.140$ (c) Solution at $t = 5.652$

(d) Error at $t = 1.256$ (e) Error at $t = 3.140$ (f) Error at $t = 5.652$

(g) Solution at $t = 6.284$ (h) Error at $t = 6.284$

Fig. 8: Demonstration of the numerical solution and errors in Example 4.

inside a 3D spherical domain with boundary Γ governed by

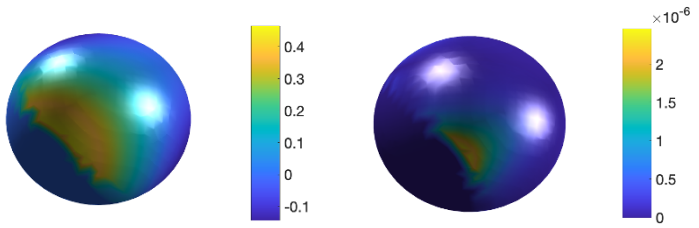
$$\Gamma : x^2 + y^2 + z^2 = 0.6^2.$$

A Robin boundary condition with $\alpha_\Gamma = \beta_\Gamma = 1.0$ is employed, and the computational domain D is set to be $\left[-\frac{2}{3}\pi, \frac{2}{3}\pi\right] \times \left[-\frac{2}{3}\pi, \frac{2}{3}\pi\right] \times \left[-\frac{2}{3}\pi, \frac{2}{3}\pi\right]$.

Table 10: Spatial Convergence Tests, Example 5

$[N_x,N_y,N_z]$	L^∞		L^2		Wall Clock (s)
	error	order	error	order	
$[33, 33, 33]$	1.19E-03		7.61E-05		121
$[65, 65, 65]$	1.21E-04	3.29	5.20E-06	3.87	1,341
$[129, 129, 129]$	8.12E-06	3.90	3.58E-07	3.86	12,058
$[257, 257, 257]$	4.00E-07	4.34	1.86E-08	4.27	107,094

Spatial convergence tests are conducted by choosing a small time step $\Delta t = 10^{-5}$. The BCG approach is used since there are only 1,000 time steps evolved from initial time $t = 0$ to final time $T = 10^{-2}$. Obtained errors and the corresponding orders are presented in Table 10, in which fourth order of convergence in space is clearly observed. In addition, the final solution and error on the surface of the sphere for the case of $N_x = N_y = N_z = 129$ are plotted in Fig. 9a - 9b. Both subfigures are rotated so that the most significant changes on the surface face front for easy observation.



(a) Solution (b) Error

Fig. 9: Demonstration of the numerical solution and errors on the surface of a sphere in Example 5.

Temporal convergence tests are conducted by fixing $N_x = N_y = N_z = 257$ and the final time $T = 1.0$ and then varying the number of times steps from $N_t = 2$ to 128. The obtained errors at the final time and associated order of convergence in time are presented in Table 11. Order 2 can be clearly observed

in Table 11. In fact, some observed orders are greater than 2 and even reach an order of 3 in the case of $N_t = 128$. Nevertheless, second order of convergence in time is confirmed in three dimensions by this example.

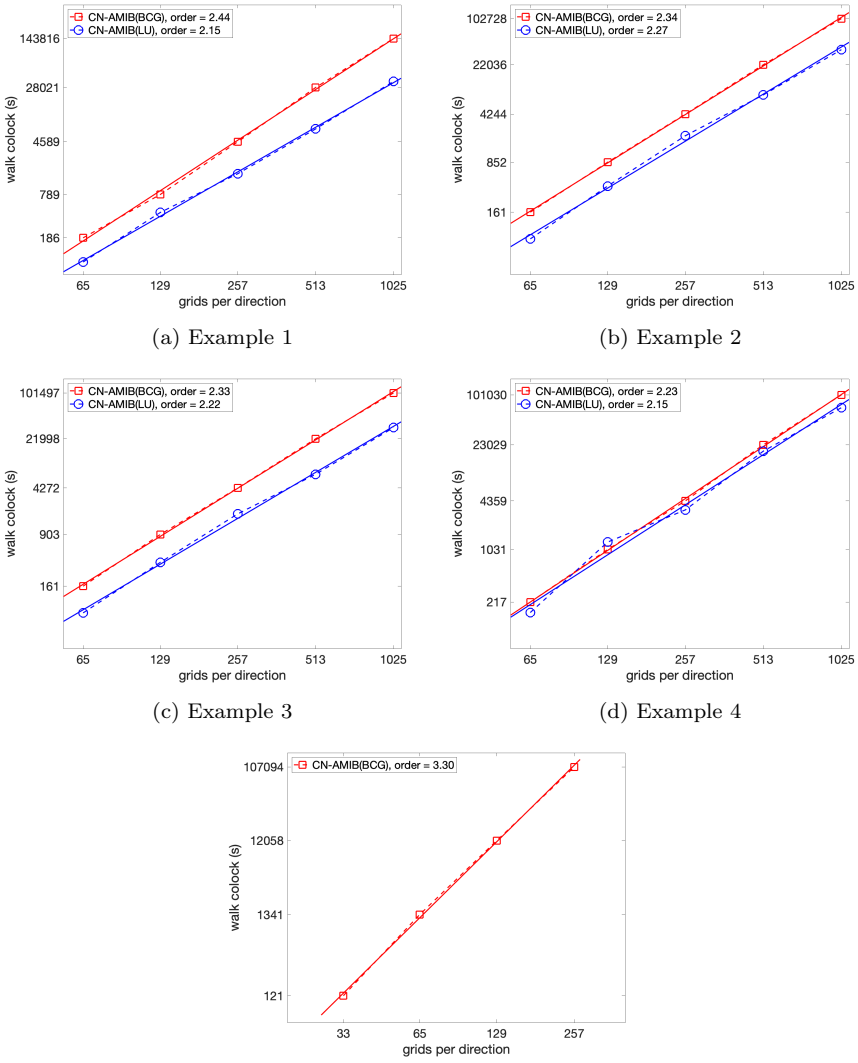
Table 11: Temporal Convergence Tests, Example 5

N_t	$\frac{\beta\Delta t}{h^2}$	L^∞		L^2		Wall Clock (s)
		error	order	error	order	
2	1,868	2.08E-03		5.05E-04		1,287
4	934	4.67E-04	2.16	8.75E-05	2.53	2,232
8	467	1.13E-04	2.04	1.58E-05	2.47	4,227
16	233	2.81E-05	2.01	2.85E-06	2.47	7,007
32	117	6.92E-06	2.02	5.06E-07	2.50	13,548
64	58	1.63E-06	2.08	8.58E-08	2.56	24,860
128	29	2.04E-07	3.00	7.29E-09	3.56	40,521

4.2 Numerical Verification of Computational Complexity

In subsection 3.6, the complexity of the LU and BCG approaches in 2D is estimated to be $O(N^3 \log N) + O(N_t N^2 \log N)$ and $O(N_t N^2 \log N)$, respectively, where N is the number of mesh points per direction and N_t is the number of time steps. In this subsection, the computational complexity will be quantitatively verified using Example 1 - 5. To this end, the CPU time will be measured by the wall clock time for all computations, and is plotted against N or N_t in log-log scales. Moreover, least squares fitting will be conducted in all plots so that the slopes of obtained straight lines are used to demonstrate the orders of FLOPS for computational complexity.

Plots of wall clock times versus N for all five numerical examples are shown in Fig. 10, in which N_t is fixed to be a large number in all cases for each figure. In Fig. 10a - 10d, results of both BCG and LU approaches for solving the four 2D examples are demonstrated. It can be seen that the complexity of both approaches is approximately a linear function in the log-log scale. For the LU approach, since N_t is much larger than N, its complexity essentially behaves as $O(N_t N^2 \log N)$ or $O(N^2 \log N)$ when N_t is a constant. The effect of $\log N$ is hard to capture in the log-log scale. Thus, we will consider a simple power form $O(N^r)$ and determine the numerical ratio r via the least squares fitting. The fitted ratios are reported in the legends, which range from 2.15 to 2.27 for the LU approach. This demonstrates the $O(N^2 \log N)$ complexity for the LU approach. For the BCG approach with a fixed N_t, the complexity should also take the form $O(N^2 \log N)$. The numerical ratio r of the BCG approach is



(a) Example 1

(b) Example 2

(c) Example 3

(d) Example 4

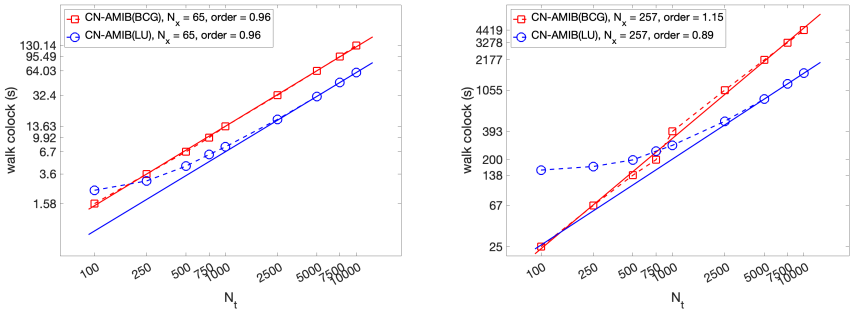
(e) Example 5

Fig. 10: Orders of FLOPS with respect to number of mesh points per direction for Example 1-5.

found to be in between 2.23 to 2.44, which is slightly larger than that of the LU approach. This is because in the AMIB method [20,35], the number of BCG iterations will increase slightly as N becomes larger, while the LU approach has a fixed number of steps for different N in the time integration. The present study shows that the difference in r for LU and BCG approaches is about 0.1

or 0.15. Thus, fairly speaking, the complexity of the BCG approach can still be regarded as $O(N^2 \log N)$.

In three dimensions, wall clock times obtained by the BCG approach for solving Example 5 are shown in Fig. 10e. Using the least squares, the complexity of the BCG approach is estimated to be $O(N^{3.30})$. This is very close to the expected FLOPS on the order of $O(N^3 \log N)$, or $O(N_t N^3 \log N)$ if N_t is not fixed.



(a) $N_x = N_y = 65$ (b) $N_x = N_y = 257$

Fig. 11: Orders of FLOPS with respect to N_t for Example 1 with different mesh sizes.

Next, wall clock times versus the number of time steps, N_t, for Example 1 are plotted in Fig. 11. For a better comparison, results obtained for two spatial discretizations, $N = N_x = N_y = 65$ and $N = N_x = N_y = 257$, are shown side by side in Fig. 11a - 11b. For both spatial discretizations, time step is fixed, $\Delta t = 10^{-3}$, while the final time varies so that the total number of time steps varies from 100 (small) to 10,000 (large).

For the BCG approach, it is clear that the complexity is still a linear function with respect to N_t in both cases. Least squares fittings are also conducted to estimate the numerical ratio for N_t, i.e., $O(N_t^r)$. Both obtained ratios (0.96 and 1.15) are very close to one. This validates the FLOPS on the order of $O(N_t)$ for the BCG approach for the present study. Combining with the previous test, one can conclude that the complexity of the BCG approach is $O(N_t N^2 \log N)$ and $O(N_t N^3 \log N)$, respectively, in two and three dimensions.

For the LU approach, it can be seen from Fig. 11 that the wall clock time does not increase much at the beginning, while it increases in a rate close to linear when N_t gets larger. This agrees with our estimation of the complexity in two dimensions, i.e., $O(N^3 \log N) + O(N_t N^2 \log N)$. Note that the first term does not depend on N_t, and is larger than the second term when N_t is small. Thus, the wall clock time increases slowly at the beginning. When N_t is large enough, the second term dominates so that the wall clock time increases linearly with respect to N_t. To verify this, least squares fittings using wall

clock times obtained at the three largest numbers of time steps for the LU approach are drawn in both subfigures. One can see the obtained orders of FLOPS are both close to one, suggesting wall clock time indeed increases in a linear manner with respect to N_t for the LU approach when N_t is large enough.

One can also see how well the two approaches perform when N varies by comparing where the two broken lines meet in Fig. 11a - 11b. In Fig. 11a ($N = 65$ (small)), the BCG approach outperforms the LU approach (less wall clock time) when $N_t < 250$, but the LU approach catches up quickly and starts to outperform the BCG approach when $N_t \geq 250$. On the other hand, in Fig. 11b ($N = 257$ (large)), the BCG approach is more efficient than the LU approach when $N_t < 750$, while the LU approach becomes more efficient that the BCG approach when $N_t \geq 750$. Note that the critical N_t value becomes larger when N is bigger. In general, our conclusion is that the BCG approach is more suitable for short-term simulations and the LU approach is more suitable for long-term simulations in two dimensions.

5 Conclusion

A novel FD method is proposed in this work to solve parabolic equations in irregularly shaped domains in two and three dimensions. Results obtained in numerical experiments suggest that the proposed ray-casting AMIB method enjoys merits such as high-order convergence (second order in time and fourth order in space), unconditional stability, accelerated calculations via multi-dimensional Fast Sine Transforms, and capability of dealing with a variety of boundary conditions and complicated geometries. The ray-casting AMIB method was first developed in [35] for solving elliptic boundary value problems over irregular domains. Besides extending to parabolic problems, the present study further improves the robustness of the ray-casting AMIB method by studying all potential cases of irregular and corner points in the correction to the fourth order central difference approximation. Consequently, the present AMIB method can accommodate more complicated geometry, while maintaining a fourth order of accuracy in space. When compared to other high-order methods for parabolic problems, the unique feature of the proposed AMIB method is its high efficiency on the order of $O(N_t N^2 \log N)$ and $O(N_t N^3 \log N)$, respectively, in two and three dimensions. By using the LU approach, the efficiency can be further improved in two dimensions for long time simulations. The generalization of the ray-casting AMIB method for solving parabolic interface problems with material interfaces and jump conditions is under our investigation.

Acknowledgements The research of Long was supported in part by the Natural Science Foundation of Guangxi in China under grant AD20238065, and the key project of Guangxi Provincial Natural Science Foundation of China under grant 2018GXNSFDA050014. The research of Boerman was supported in part by the grant of Research in Mathematics and the Sciences (RIMS) offered by the College of the Sciences and Mathematics at West Chester

University of Pennsylvania, USA. The research of Zhao was supported in part by the National Science Foundation (NSF) grant DMS-2110914.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Adams, L., Li, Z.: The immersed interface/multigrid methods for interface problems. SIAM Journal on Scientific Computing **24**(2), 463–479 (2002)
2. Barrett, A., Fogelson, A.L., Griffith, B.E.: A hybrid semi-lagrangian cut cell method for advection-diffusion problems with robin boundary conditions in moving domains. Journal of Computational Physics **449**, 110805 (2022). DOI https://doi.org/10.1016/j.jcp.2021.110805. URL https://www.sciencedirect.com/science/article/pii/S0021999121007002
3. Bochkov, D., Gibou, F.: Solving poisson-type equations with robin boundary conditions on piecewise smooth interfaces. Journal of Computational Physics **376**, 1156–1198 (2019). DOI https://doi.org/10.1016/j.jcp.2018.10.020. URL https://www.sciencedirect.com/science/article/pii/S0021999118306831
4. Bruno, O.P.: Fast, high-order, high-frequency integral methods for computational acoustics and electromagnetics. In: Topics in computational wave propagation, pp. 43–82. Springer (2003)
5. Bruno, O.P., Han, Y., Pohlman, M.M.: Accurate, high-order representation of complex three-dimensional surfaces via fourier continuation analysis. Journal of Computational Physics **227**(2), 1094–1125 (2007). DOI https://doi.org/10.1016/j.jcp.2007.08.029. URL https://www.sciencedirect.com/science/article/pii/S0021999107003816
6. Bruno, O.P., Lyon, M.: High-order unconditionally stable fc-ad solvers for general smooth domains i. basic elements. Journal of Computational Physics **229**(6), 2009–2033 (2010). DOI https://doi.org/10.1016/j.jcp.2009.11.020. URL https://www.sciencedirect.com/science/article/pii/S0021999109006391
7. Chai, M., Luo, K., Shao, C., Wang, H., Fan, J.: A finite difference discretization method for heat and mass transfer with robin boundary conditions on irregular domains. Journal of Computational Physics **400**, 108890 (2020). DOI https://doi.org/10.1016/j.jcp.2019.108890. URL https://www.sciencedirect.com/science/article/pii/S0021999119305881
8. Chai, M., Luo, K., Wang, H., Zheng, S., Fan, J.: Imposing mixed dirichlet-neumann-robin boundary conditions on irregular domains in a level set/ghost fluid based finite difference framework. Computers & Fluids **214**, 104772 (2021). DOI https://doi.org/10.1016/j.compfluid.2020.104772. URL https://www.sciencedirect.com/science/article/pii/S004579302030342X
9. Chen, H., Min, C., Gibou, F.: A supra-convergent finite difference scheme for the poisson and heat equations on irregular domains and non-graded adaptive cartesian grids. Journal of Scientific Computing **31**(1), 19–60 (2007). DOI 10.1007/s10915-006-9122-8. URL https://doi.org/10.1007/s10915-006-9122-8
10. Chen, Z., Zou, J.: Finite element methods and their convergence for elliptic and parabolic interface problems. Numerische Mathematik **79**(2), 175–202 (1998)
11. Clain, S., Lopes, D., Pereira, R.: Very high-order cartesian-grid finite difference method on arbitrary geometries. Journal of Computational Physics **434**, 110217 (2021). DOI https://doi.org/10.1016/j.jcp.2021.110217. URL https://www.sciencedirect.com/science/article/pii/S0021999121001121
12. Coco, A., Russo, G.: Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface. Journal of Computational Physics **361**, 299–330 (2018). DOI https://doi.org/10.1016/j.jcp.2018.01.016. URL https://www.sciencedirect.com/science/article/pii/S0021999118300263

13. Coco, A., Semplice, M., Serra Capizzano, S.: A level-set multigrid technique for nonlinear diffusion in the numerical simulation of marble degradation under chemical pollutants. Applied Mathematics and Computation **386**, 125503 (2020). DOI https://doi.org/10.1016/j.amc.2020.125503. URL https://www.sciencedirect.com/science/article/pii/S0096300320304616
14. Douglas Jr, J., Peaceman, D.W.: Numerical solution of two-dimensional heat-flow problems. AIChE Journal **1**(4), 505–512 (1955)
15. Fedkiw, R.P., Aslam, T., Merriman, B., Osher, S.: A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). Journal of Computational Physics **152**(2), 457–492 (1999). DOI https://doi.org/10.1006/jcph.1999.6236. URL https://www.sciencedirect.com/science/article/pii/S0021999199962368
16. Feng, H., Long, G., Zhao, S.: An augmented matched interface and boundary (mib) method for solving elliptic interface problem. Journal of Computational and Applied Mathematics **361**, 426–443 (2019)
17. Feng, H., Long, G., Zhao, S.: Fft-based high order central difference schemes for poisson's equation with staggered boundaries. Journal of Scientific Computing **86**(1), 1–25 (2021)
18. Feng, H., Zhao, S.: Fft-based high order central difference schemes for three-dimensional poisson's equation with various types of boundary conditions. Journal of Computational Physics **410**, 109391 (2020)
19. Feng, H., Zhao, S.: A fourth order finite difference method for solving elliptic interface problems with the fft acceleration. Journal of Computational Physics **419**, 109677 (2020)
20. Feng, H., Zhao, S.: A multigrid based finite difference method for solving parabolic interface problem. Electronic Research Archive **29**(5), 3141 (2021)
21. Fernández-Fidalgo, J., Clain, S., Ramírez, L., Colominas, I., Nogueira, X.: Very high-order method on immersed curved domains for finite difference schemes with regular cartesian grids. Computer Methods in Applied Mechanics and Engineering **360**, 112782 (2020). DOI https://doi.org/10.1016/j.cma.2019.112782. URL https://www.sciencedirect.com/science/article/pii/S0045782519306747
22. Fornberg, B.: Classroom note: Calculation of weights in finite difference formulas. SIAM review **40**(3), 685–691 (1998)
23. Gibou, F., Fedkiw, R.: A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem. Journal of Computational Physics **202**(2), 577–601 (2005). DOI https://doi.org/10.1016/j.jcp.2004.07.018. URL https://www.sciencedirect.com/science/article/pii/S0021999104002980
24. Gibou, F., Min, C., Fedkiw, R.: High resolution sharp computational methods for elliptic and parabolic problems in complex geometries. Journal of Scientific Computing **54**(2), 369–413 (2013). DOI 10.1007/s10915-012-9660-1. URL https://doi.org/10.1007/s10915-012-9660-1
25. Li, C., Long, G., Li, Y., Zhao, S.: Alternating direction implicit (adi) methods for solving two-dimensional parabolic interface problems with variable coefficients. Computation **9**(7) (2021). DOI 10.3390/computation9070079. URL https://www.mdpi.com/2079-3197/9/7/79
26. Li, C., Wei, Z., Long, G., Campbell, C., Ashlyn, S., Zhao, S.: Alternating direction ghost-fluid methods for solving the heat equation with interfaces. Computers & Mathematics with Applications **80**(5), 714–732 (2020)
27. Li, C., Zhao, S.: A matched peaceman–rachford adi method for solving parabolic interface problems. Applied Mathematics and Computation **299**, 28–44 (2017)
28. Li, Z., Chen, X., Zhang, Z.: On multiscale adi methods for parabolic pdes with a discontinuous coefficient. SIAM Multiscale Model. Simul. **16**(4), 1623–1647 (2018)
29. Li, Z., Mayo, A.: Adi methods for heat equations with discontinuous along an arbitrary interface. In: Proceedings of Symposia in Applied Mathematics, vol. 48, pp. 311–315 (1993)
30. Lin, T., Yang, Q., Zhang, X.: Partially penalized immersed finite element methods for parabolic interface problems. Numerical Methods for Partial Differential Equations **31**(6), 1925–1947 (2015). DOI https://doi.org/10.1002/num.21973. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/num.21973

31. Lyon, M., Bruno, O.P.: High-order unconditionally stable fc-ad solvers for general smooth domains ii. elliptic, parabolic and hyperbolic pdes; theoretical considerations. Journal of Computational Physics **229**(9), 3358–3381 (2010). DOI https://doi.org/10.1016/j.jcp.2010.01.006. URL https://www.sciencedirect.com/science/article/pii/S0021999110000215
32. Marques, A.N., Nave, J.C., Rosales, R.R.: A correction function method for poisson problems with interface jump conditions. Journal of Computational Physics **230**(20), 7567–7597 (2011). DOI https://doi.org/10.1016/j.jcp.2011.06.014. URL https://www.sciencedirect.com/science/article/pii/S0021999111003780
33. Pan, K., Wu, X., Hu, H., Yu, Y., Li, Z.: A new fv scheme and fast cell-centered multigrid solver for 3d anisotropic diffusion equations with discontinuous coefficients. Journal of Computational Physics **449**, 110794 (2022). DOI https://doi.org/10.1016/j.jcp.2021.110794. URL https://www.sciencedirect.com/science/article/pii/S0021999121006896
34. Papac, J., Gibou, F., Ratsch, C.: Efficient symmetric discretization for the poisson, heat and stefan-type problems with robin boundary conditions. Journal of Computational Physics **229**(3), 875–889 (2010). DOI https://doi.org/10.1016/j.jcp.2009.10.017. URL https://www.sciencedirect.com/science/article/pii/S0021999109005622
35. Ren, Y., Feng, H., Zhao, S.: A fft accelerated high order finite difference method for elliptic boundary value problems over irregular domains. Journal of Computational Physics **448**, 110762 (2022)
36. Song, L., Zhao, S.: Symmetric interior penalty galerkin approaches for two-dimensional parabolic interface problems with low regularity solutions. Journal of Computational and Applied Mathematics **330**, 356–379 (2018). DOI https://doi.org/10.1016/j.cam.2017.09.018. URL https://www.sciencedirect.com/science/article/pii/S0377042717304491
37. Stein, D.B., Guy, R.D., Thomases, B.: Immersed boundary smooth extension: A high-order method for solving pde on arbitrary smooth domains using fourier spectral methods. Journal of Computational Physics **304**, 252–274 (2016). DOI https://doi.org/10.1016/j.jcp.2015.10.023. URL https://www.sciencedirect.com/science/article/pii/S0021999115006877
38. Wan, J.W.L., Liu, X.D.: A boundary condition–capturing multigrid approach to irregular boundary problems. SIAM Journal on Scientific Computing **25**(6), 1982–2003 (2004). DOI 10.1137/S1064827503428540. URL https://doi.org/10.1137/S1064827503428540
39. Wei, Z., Li, C., Zhao, S.: A spatially second order alternating direction implicit (adi) method for solving three dimensional parabolic interface problems. Computers & Mathematics with Applications **75**, 2173–2192 (2018)
40. Wiegmann, A., Bube, K.P.: The explicit-jump immersed interface method: finite difference methods for pdes with piecewise smooth solutions. SIAM Journal on Numerical Analysis **37**(3), 827–862 (2000)
41. Xie, Y., Ying, W.: A fourth-order kernel-free boundary integral method for the modified helmholtz equation. Journal of Scientific Computing **78**(3), 1632–1658 (2019). DOI 10.1007/s10915-018-0821-8. URL https://doi.org/10.1007/s10915-018-0821-8
42. Yang, Q., Zhang, X.: Discontinuous galerkin immersed finite element methods for parabolic interface problems. Journal of Computational and Applied Mathematics **299**, 127–139 (2016). DOI https://doi.org/10.1016/j.cam.2015.11.020. URL https://www.sciencedirect.com/science/article/pii/S0377042715005658. Recent Advances in Numerical Methods for Systems of Partial Differential Equations
43. Zhao, S.: On the spurious solutions in the high-order finite difference methods for eigenvalue problems. Computer methods in applied mechanics and engineering **196**(49-52), 5031–5046 (2007)
44. Zhao, S.: A fourth order finite difference method for waveguides with curved perfectly conducting boundaries. Computer methods in applied mechanics and engineering **199**(41-44), 2655–2662 (2010)
45. Zhao, S.: A matched alternating direction implicit (adi) method for solving the heat equation with interfaces. Journal of Scientific Computing **63**(1), 118–137 (2015)
46. Zhao, S., Wei, G.: High-order fdtd methods via derivative matching for maxwell's equations with material interfaces. Journal of Computational Physics **200**(1), 60–103 (2004). DOI https://doi.org/10.1016/j.jcp.2004.03.008. URL https://www.sciencedirect.com/science/article/pii/S0021999104001330

47. Zhao, S., Wei, G.: Matched interface and boundary (mib) for the implementation of boundary conditions in high-order central finite differences. International journal for numerical methods in engineering **77**(12), 1690–1730 (2009)
48. Zhao, S., Wei, G., Xiang, Y.: Dsc analysis of free-edged beams by an iteratively matched boundary method. Journal of Sound and Vibration **284**(1-2), 487–493 (2005)
49. Zhou, Y., Zhao, S., Feig, M., Wei, G.: High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. Journal of Computational Physics **213**(1), 1–30 (2006). DOI https://doi.org/10.1016/j.jcp.2005.07.022. URL https://www.sciencedirect.com/science/article/pii/S0021999105003578
50. Zhu, P., Zhang, Q., Liu, T.: Stable generalized finite element method (sgfem) for parabolic interface problems. Journal of Computational and Applied Mathematics **367**, 112475 (2020). DOI https://doi.org/10.1016/j.cam.2019.112475. URL https://www.sciencedirect.com/science/article/pii/S0377042719304789