INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 22, Number 6, Pages 860–896

doi: 10.4208/ijnam2025-1038

A MULTIGRID-BASED FOURTH ORDER FINITE DIFFERENCE METHOD FOR ELLIPTIC INTERFACE PROBLEMS WITH VARIABLE COEFFICIENTS

YIMING REN AND SHAN ZHAO*

Abstract. The paper introduces a fourth-order augmented matched interface and boundary (AMIB) method for solving elliptic interface problems with complex interfaces and piecewise smooth coefficients in two and three dimensions. To resolve the challenge posed by non-constant coefficients within the AMIB framework, the fast Fourier transform (FFT) Poisson solver of the existing AMIB methods is replaced by a geometric multigrid method to efficiently invert the Laplacian discretization matrix. In this work, a fourth order multigrid method will be employed in the framework of the AMIB method for elliptic interface problems with variable coefficients in two and three dimensions. Based on a Cartesian mesh, the standard fourth-order finite differences are employed to approximate the first and second derivatives involved in the Laplacian with variable coefficients. Near the interface, a fourth-order ray-casting matched interface and boundary (MIB) scheme is generalized to variable coefficient problems to enforce interface jump conditions in the corrected finite difference discretization. The augmented formulation of the AMIB allows us to decouple the interface treatments from the inversion of the Laplacian discretization matrix, so that one essentially solves an elliptic subproblem without interfaces. A fourth order geometric multigrid method is introduced to solve this subproblem with a Dirichlet boundary condition, where fourth order one-sided finite difference approximations are considered near the boundary in all grid levels. The proposed multigrid method significantly enhances the computational efficiency in solving variable coefficient problems, while achieving a fourth-order accuracy in accommodating complex interfaces and discontinuous solutions.

Key words. Variable coefficient elliptic interface problem, high order finite difference schemes, geometric multigrid, matched interface and boundary method (MIB), gradient recovery.

1. Introduction

This work focuses on solving multi-dimensional elliptic interface problems with variable coefficients. We consider an elliptic partial difference equation (PDE) in a domain Ω

$$\nabla \cdot (\beta \nabla u) + \kappa u = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{1}$$

subject to Dirichlet boundary conditions on the boundary $\partial\Omega$. The function $u(\mathbf{x})$ together with the corresponding source term $f(\mathbf{x})$ depend on a vector variable $\mathbf{x} = (x_1, x_2, \cdots, x_d)$ for $d = 2$ or $d = 3$ on a rectangular or cubic domain Ω. The interface Γ defined by $\Gamma = \Omega^+ \cap \Omega^-$ divides the computational domain Ω into disjoint subdomains $\Omega = \Omega^+ \cup \Omega^-$. An illustration of subdomains in two dimensions is given in Fig. 1. The coefficients $\beta(\mathbf{x})$ and $\kappa(\mathbf{x})$ are smooth functions on each disjoint subdomain, but may be discontinuous across the interface Γ, i.e., they are piecewise smooth functions

$$\beta(\mathbf{x}) = \begin{cases} \beta^-(\mathbf{x}) & \text{in } \Omega^- \\ \beta^+(\mathbf{x}) & \text{in } \Omega^+, \end{cases} \qquad \kappa(\mathbf{x}) = \begin{cases} \kappa^-(\mathbf{x}) & \text{in } \Omega^- \\ \kappa^+(\mathbf{x}) & \text{in } \Omega^+. \end{cases}$$

Received by the editors on November 23, 2024 and, accepted on March 26, 2025.
2000 *Mathematics Subject Classification.* 65N06, 65N55.
*Corresponding author.

Similarly, the source $f(\mathbf{x})$ is also piecewisely smooth with notation $f^+(\mathbf{x})$ and $f^-(\mathbf{x})$, respectively, in Ω^+ and Ω^-. It is assumed that $\beta(\mathbf{x})$ is always positive. Across the interface Γ, two jump conditions are known for the function and its flux in the normal direction

$$[\![u]\!] := u^+ - u^- = \phi(\mathbf{x}), \qquad \mathbf{x} \in \Gamma, \tag{2}$$

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

where $\vec{n}$ is the outward normal direction of Γ pointing from Ω^- to Ω^+, and the superscript stands for the limiting value from each side of the interface. Equations (2) and (3) are called as the zeroth and first order jump conditions. Such an elliptic interface problem with discontinuous coefficients has wide application in a variety of fields such as fluid dynamics, modeling of underground waste disposal, solidification processes, oil reservoir simulations, and many others.

When the coefficients $\beta(\mathbf{x})$ and $\kappa(\mathbf{x})$ are piecewise constants, the present problem reduces to the usual elliptic interface problem, for which the finite element method (FEM) is a commonly used approach. Classical FEM [3, 7, 12, 35] delivers satisfactory accuracy, particularly when the interfaces align well with the underlying meshes. However, practical scenarios often necessitate the construction of numerical methods on non-fitted meshes. This requirement has driven the development of the Immersed FEM (IFEM) [31], in which local basis functions are adapted to ensure compliance with the prescribed jump conditions.

Finite difference methods on Cartesian grids have received extensive attention in the context of elliptic interface problems. Peskin [44] laid the foundation for this field by introducing a first-order accurate immersed boundary method in the 1970s. LeVeque and Li [34] pioneered the first second-order finite difference approach, the Immersed Interface Method (IIM), which employs Taylor series expansions to determine stencil weights. Another popular technique is the Ghost Fluid Method (GFM) [17], typically a first-order method [40], but it has been extended to second order in [39]. The recovery of flux convergence in GFM has been investigated in [16]. Chen et al. [11] developed a second-order compact finite difference method for solving elliptic interface problems. In addition to finite element and finite difference methods, other effective algorithms for solving elliptic interface problems include virtual node method [4, 28], finite volume method [6], and coupling interface method [13, 48]. We note that the aforementioned methods usually deliver first or second order accuracy.

Addressing variable coefficients in elliptic interface problems is a challenging endeavor. These variable coefficients could exhibit significant variations across the interface, causing abrupt shifts in solutions. Managing such discontinuities presents a huge difficulty in maintaining numerical stability and precision. Due to the inherent complexity, only a limited number of studies have ventured into this domain. A weak formulation has been developed in [30] for a grid that fits the geometry of the problem, offering a solution to variable coefficient elliptic equations. Remarkably, it only requires Lipschitz continuity rather than smoothness for the interfaces. Expanding upon the foundational principles of the IIM, novel techniques like the Decomposed Immersed Interface Method (DIIM) [5] and Augmented Immersed Interface Method (AIIM) [37] have been proposed to address elliptic interface problems with variable coefficients. A second-order finite-volume method is presented in [42] that operates on Cartesian grids for variable coefficient elliptic equations with embedded interfaces. Ref. [41] describes a composite spectral scheme for solving variable coefficient elliptic boundary value problems with smooth coefficients. Recently, boundary-optimized summation-by-parts (SBP) finite difference operators for second derivatives with variable coefficients are presented in [49], which could be used to solve the variable coefficient elliptic equation. The operators achieve increased accuracy by utilizing non-equispaced grid points close to the boundaries of the grid. The compact finite difference scheme introduced in [23] has been generalized in [24] to address elliptic interface problems featuring discontinuous and highly contrasted variable coefficients.

One major focus of this study is on developing fast Poisson solvers for elliptic interface problems. It is well known that for elliptic PDEs in the absence of interfaces, algebraic computations can be significantly accelerated using fast Poisson solvers, such as geometric multigrid with a complexity of $O(N)$ and fast Fourier transform (FFT) with a complexity of $O(N \log N)$, where N represents the spatial degree of freedom. Consequently, there is a compelling motivation to integrate Poisson solvers into interface algorithms to ensure that the algebraic computation is not restricted by iterative solvers with a general complexity of $O(N^2)$.

For elliptic interface problems with piecewise constant coefficients, the utilization of FFT Poisson solver has gained significant popularity for accelerating algebraic computations [36, 50, 19, 21]. A remarkable breakthrough in this field is the AIIM [36, 37], which incorporates auxiliary variables to establish an augmented system, facilitating the approximation of the Laplacian operator through the standard finite difference stencil. This approximation results in a symmetric and diagonally dominant matrix, which can be inverted by the FFT. The iterative solution of auxiliary variables is achieved through a Schur complement procedure. Due to the total number of auxiliary variables being one dimension less than N, the augmented approach typically entails an algebraic complexity of $O(N \log N)$. AIIM's origins can be traced back to Li's pioneering work in 1998 when it was initially applied to problems involving piecewise constant coefficients [36]. Since then, AIIM has witnessed considerable success in a wide range of applications [43, 37, 53]. Recently, the Augmented Matched Interface and Boundary (AMIB) method [19, 20, 21, 46] has been introduced for solving elliptic interface problems and boundary value problems. With a FFT acceleration, the AMIB method can deliver fourth order accuracy in treating both interfaces [21] and boundaries [20]. Unlike AIIMs, AMIB does not necessitate additional jump conditions in local coordinates. Instead, it relies on only two low-order jump conditions as defined by (2) and (3).

However, the FFT is intractable for elliptic problems involving variable coefficients, because the FFT solver is limited to linear, time-invariant systems with constant coefficients. Therefore, the multigrid method [15, 10, 33, 32, 1, 37] is the only effective fast Poisson solver that works for variable coefficient problems. In multigrid solvers, the construction of interpolation, smoothing operators, and coarse grid point selection for structured or unstructured meshes requires specialized approaches [9]. Algebraic multigrid (AMG) [47] is notable for its purely algebraic selection of coarse grid points, eliminating the need for interface geometry and dimensionality constraints during construction. It utilizes discontinuous coefficients and geometry in matrix-dependent interpolation, albeit requiring substantial storage for the generated coarse grid operator matrices. AMG has been successfully applied in the immersed finite element method, solving both stationary and moving interface problems [18]. Geometric multigrid methods incorporate interface geometry into PDE discretization, considering boundary and interface jump conditions, and are usually much more efficient than AMG solvers. In the work by Adams and Li [1], a 2D IIM multigrid method was developed, preserving the maximum principle. Black-box multigrid interpolation is applied away from the interface, and interpolation weights are derived using Taylor expansion for grid points near the interface. Subsequently, an improved IIM-based multigrid method [2] was introduced, modifying interpolation and restriction operators to produce M-matrices for coarse grids, maintaining the number of V-cycles as mesh refines. Coco extended the multigrid approach [14] initially designed for continuous coefficient boundary value problems to address discontinuous coefficient interface problems [15]. We note that designing new multigrid approaches is always necessary when adopting new interface algorithms, each requiring specialized treatments for interpolation, knowledge of interface geometry, and definition of restriction and coarse grid operators.

Another major focus of this paper is on the development of high-order (third-order or higher) interface methods, which are essential for tackling real-world problems with intricate characteristics, such as those involving high-frequency waves. To the best of the authors' knowledge, no high-order interface method has ever been developed for solving variable coefficient problems with an $O(N)$ or $O(N \log N)$ complexity, whereas there exist successful developments for elliptic interface problems with constant coefficients. The high order interface methods were first developed without fast Poisson solvers. The matched interface and boundary (MIB) method, which incorporates fictitious points and enforces zeroth and first-order jump conditions iteratively, is able to achieve fourth or sixth order accuracy for smoothly curved interfaces [56, 52]. Within the IIM framework, a fourth-order method was introduced by discretizing high-order jump conditions and mixed derivatives [43]. Furthermore, a high-order IIM approach is readily applicable to practical two-phase flow problems, by requiring only the physical jump conditions for zero and first derivatives [54]. For usual elliptic interface problems, acceleration by the FFT fast solver has been successfully accomplished in a few high order interface methods. In [21], an augmented MIB (AMIB) method has been developed in two dimensions (2D), which is able to not only achieve a fourth order of accuracy, but also maintain the FFT efficiency, in solving elliptic interface problems with any boundary conditions. The AMIB has been further generalized to three-dimensions (3D) by using a novel ray-casting MIB interface treatment [45]. A kernel-free boundary integral (KFBI) method has been developed in [55] for solving elliptic PDEs in 2D and 3D. The boundary and volume integrals involved in the iterative solution of the integral systems are evaluated through equivalent interface problems, which are solved by fourth order compact finite difference methods with the FFT acceleration. Beyond these innovative interface schemes, the literature contains a wealth of attractive high-order numerical methods for addressing elliptic boundary value problems defined over irregular domains [38, 27, 25, 46].

In this paper, we will develop a novel AMIB method with a geometric multigrid solver for elliptic interface problems with variable coefficients and Dirichlet boundaries. We note that all existing fourth order AMIB methods [20, 21, 46, 45] are based on the FFT solver, which is not applicable to variable coefficient problems. A standard multigrid solver has been integrated with the AMIB in [22] for solving parabolic interface problems, which unfortunately can only deliver a second order of accuracy. In the present study, the ray-casting MIB scheme [45] will be extended to solve variable coefficient problems with curved interfaces and discontinuous solutions. The ray-casting MIB method is considerably simpler and more robust than the Cartesian MIB scheme [56, 52], for which jump condition discretization involves all Cartesian directions in multi-dimensions. The augmented formulation of the AMIB allows us to decouple the interface treatments from the inversion of the Laplacian discretization matrix so that one essentially solves an elliptic subproblem without interfaces. A new fourth order geometric multigrid method is introduced to solve this subproblem with Dirichlet boundaries, where fourth order one-sided finite difference approximations are considered near the boundary in all grid levels. The proposed multigrid method enjoys benefits such as the avoidance of storage for coarse grid matrices and a straightforward implementation because the interpolation and restriction processes are significantly simplified without considering interfaces.

The rest of the paper is organized as follows. In Section 2, a new finite difference scheme is proposed to solve interface problems in 2D. Then a uniform augmented MIB system will be formulated. Section 3 is dedicated to the numerical results to demonstrate the performance of the proposed algorithm in 2D and 3D. Lastly, a summary is given in section 4.

2. Theory and algorithm

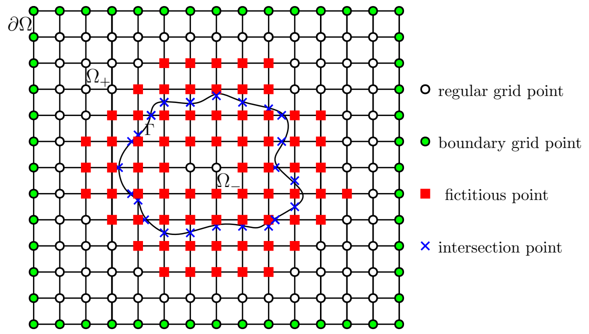


Figure 1. An illustration of the different type of grid points used in the AMIB method in 2D.

In this section, the formulation of the proposed AMIB method will be presented primarily in 2D. Due to the tensor product nature of the AMIB method, its generalization to 3D is straightforward.

In 2D, we assume uniform mesh spacings, denoted as h_x in the x-direction and h_y in the y-direction, which divide the rectangular domain $\Omega = [a, b] \times [c, d]$ into equally spaced intervals. This assumption implies that h_x equals $(b - a)/n_x$, and h_y equals $(d - c)/n_y$. The grid coordinates in 2D are defined as follows:

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

Before we proceed with the details of discretizations, different types of grid points need to be defined. All types of grid points are illustrated in Fig. 1. The locations where the interface Γ intersects with the grid lines are called interface intersection points. The boundary grid points at $\partial\Omega$ are where the boundary conditions are imposed. Furthermore, referring to Fig. 1, away from the interface Γ, the standard fourth-order central difference approximation will be employed at all regular points. However, for irregular points near the interface, the finite difference will need function values from the other side of the interface Γ. A formulation for irregular points in the fourth-order case will be discussed later. Define min and max functions at a node (x_i, y_j) as

$$\varphi_{ij}^{min} = \min\{\varphi_{i-2,j}, \varphi_{i-1,j}, \varphi_{i,j}, \varphi_{i+1,j}, \varphi_{i+2,j}, \varphi_{i,j-2}, \varphi_{i,j-1}, \varphi_{i,j+1}, \varphi_{i,j+2}\},$$
$$\varphi_{ij}^{max} = \max\{\varphi_{i-2,j}, \varphi_{i-1,j}, \varphi_{i,j}, \varphi_{i+1,j}, \varphi_{i+2,j}, \varphi_{i,j-2}, \varphi_{i,j-1}, \varphi_{i,j+1}, \varphi_{i,j+2}\},$$

where the interface Γ is assumed to be represented by the zero level set $\varphi(x, y) = 0$. When $\varphi_{i,j}^{min}\varphi_{i,j}^{max} < 0$, We call the grid node (x_i, y_j) a irregular point, otherwise regular point.

To discretize the original PDE (1) by using central differences, it is transformed to an equivalent form first

$$\begin{aligned}(\beta u_x)_x + (\beta u_y)_y + \kappa u &= \beta u_{xx} + \beta_x u_x + \beta u_{yy} + \beta_y u_y + \kappa u \\ &= f(\mathbf{x}), \quad \mathbf{x} \in (\Omega^- \cup \Omega^+) \setminus \Gamma.\end{aligned} \tag{4}$$

Here the subscripts denote derivatives, e.g., $u_{xx} = \frac{\partial^2 u}{\partial x^2}$. We assume that β is a piecewise smooth function and its derivatives β_x and β_y can be analytically calculated in each domain Ω^- or Ω^+. We will solve (4) subject to a Dirichlet boundary condition and interface jump conditions (2) and (3), which actually gives the solution of the original elliptic interface problem.

The fourth order finite difference discretization of the governing equation (4) involves approximations to both u_x and u_{xx}. For regular points away from the boundary, standard fourth-order central differences are employed. The finite difference discretization near the interface will be handled by the MIB scheme in an augmented formulation, and will be discussed later. At the boundary nodes, the Dirichlet boundary condition is simply imposed. However, the fourth-order central differences need to be modified for nodes immediately adjacent to boundary nodes. In order to facilitate the proposed fourth order multigrid method, fourth-order one-sided finite difference approximations will be employed. In particular, the following finite difference formulation will be studied, which maintains a truncation error of $O(h^4)$.

$$\begin{aligned}(\beta u_x)_x(x_i) \approx\ & \frac{\beta}{h^2}[-\frac{1}{12}u(x_{i-2}) + \frac{4}{3}u(x_{i-1}) - \frac{5}{2}u(x_i) + \frac{4}{3}u(x_{i+1}) - \frac{1}{12}u(x_{i+2})] \\ & + \frac{\beta_x}{h}[\frac{1}{12}u(x_{i-2}) - \frac{2}{3}u(x_{i-1}) + \frac{2}{3}u(x_{i+1}) - \frac{1}{12}u(x_{i+2})], \\ & i = 2, 3 \cdots, n_x - 2,\end{aligned} \tag{5}$$

$$\begin{aligned}(\beta u_x)_x(x_1) \approx\ & \frac{\beta}{h^2}[\frac{11}{12}u(x_0) - \frac{5}{3}u(x_1) + \frac{1}{2}u(x_2) + \frac{1}{3}u(x_3) - \frac{1}{12}u(x_4)] \\ & + \frac{\beta_x}{h}[-\frac{1}{4}u(x_0) - \frac{5}{6}u(x_1) + \frac{3}{2}u(x_2) - \frac{1}{2}u(x_3) + \frac{1}{12}u(x_4)],\end{aligned} \tag{6}$$

$$\begin{aligned}(\beta u_x)_x(x_{n_x-1}) \approx\ & \frac{\beta}{h^2}[-\frac{1}{12}u(x_{n_x-4}) + \frac{1}{3}u(x_{n_x-3}) \\ & + \frac{1}{2}u(x_{n_x-2}) - \frac{5}{3}u(x_{n_x-1}) + \frac{11}{12}u(x_{n_x})] \\ & + \frac{\beta_x}{h}[-\frac{1}{12}u(x_{n_x-4}) + \frac{1}{2}u(x_{n_x-3}) \\ & - \frac{3}{2}u(x_{n_x-2}) + \frac{5}{6}u(x_{n_x-1}) + \frac{1}{4}u(x_{n_x})].\end{aligned} \tag{7}$$

Analogously, finite difference discretization could be obtained for y-partial derivatives. Figure 2 provides a visual representation of the finite difference stencils and

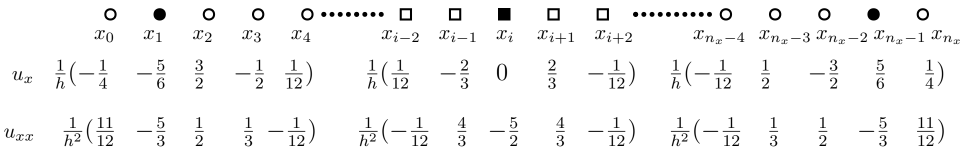

x_0 x_1 x_2 x_3 x_4 x_{i-2} x_{i-1} x_i x_{i+1} x_{i+2} x_{n_x-4} x_{n_x-3} x_{n_x-2} x_{n_x-1} x_{n_x}
u_x $\frac{1}{h}(-\frac{1}{4}\ \ -\frac{5}{6}\ \ \frac{3}{2}\ \ -\frac{1}{2}\ \ \frac{1}{12})$ $\frac{1}{h}(\frac{1}{12}\ \ -\frac{2}{3}\ \ 0\ \ \frac{2}{3}\ \ -\frac{1}{12})$ $\frac{1}{h}(-\frac{1}{12}\ \ \frac{1}{2}\ \ -\frac{3}{2}\ \ \frac{5}{6}\ \ \frac{1}{4})$
u_{xx} $\frac{1}{h^2}(\frac{11}{12}\ \ -\frac{5}{3}\ \ \frac{1}{2}\ \ \frac{1}{3}\ \ -\frac{1}{12})$ $\frac{1}{h^2}(-\frac{1}{12}\ \ \frac{4}{3}\ \ -\frac{5}{2}\ \ \frac{4}{3}\ \ -\frac{1}{12})$ $\frac{1}{h^2}(-\frac{1}{12}\ \ \frac{1}{3}\ \ \frac{1}{2}\ \ -\frac{5}{3}\ \ \frac{11}{12})$

FIGURE 2. The finite difference weights used in the fourth-order finite difference discretization for approximating the first-order derivative (as indicated in the first row) and the second-order derivative (as indicated in the second row) at the solid point. The squares denote the standard central difference scheme applied at x_i where $i = 3, 4, \cdots, n_x - 2$, while the circles signify the one-sided finite difference scheme at x_1 or x_{n_x-1}.

associated weights. In this paper, all finite difference weights are calculated according to the Ref. [26].

2.1. A multigrid method for the fourth order finite difference method. Without considering the interface, we propose a multigrid method for the fourth order finite difference discretization of a one-dimensional (1D) elliptic equation with variable coefficients,

$$(\beta u_x)_x + \kappa u = b, \tag{8}$$

subject to Dirichlet boundary conditions. Due to the tensor product nature of the AMIB method, such a 1D multigrid method can be extended to high dimensions. Denote $\beta_i = \beta(x_i)$, $\beta_i' = \beta_x(x_i)$, $\kappa_i = \kappa(x_i)$, $i = 1, 2, \cdots, n_x - 1$. Taking advantage of the discretization (5), (6), and (7) to $(\beta u_x)_x$, the 1D elliptic problem (8) could be rewritten to a matrix-vector form:

$$AU = B, \tag{9}$$

where $A \in \mathbb{R}^{n_x+1, n_x+1}, U \in \mathbb{R}^{n_x+1}$, and $B \in \mathbb{R}^{n_x+1}$. Here the vectors U and B contain function values of u and b, respectively, at grid nodes. The matrix A is given by

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & \cdots & 0 & 0 \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & \cdots & 0 & 0 \\ 0 & w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & w_{n_x-2,1} & w_{n_x-2,2} & w_{n_x-2,3} & w_{n_x-2,4} & w_{n_x-2,5} \\ 0 & 0 & \cdots & w_{n_x-1,1} & w_{n_x-1,2} & w_{n_x-1,3} & w_{n_x-1,4} & w_{n_x-1,5} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

The central difference weights for interior nodes $i = 2, 3, \cdots, n_x - 2$ are given as

$$w_{i,1} = -\frac{1}{12h^2}\beta_i + \frac{1}{12h}\beta_i', \quad w_{i,2} = \frac{4}{3h^2}\beta_i - \frac{2}{3h}\beta_i', \quad w_{i,3} = -\frac{5}{2h^2}\beta_i + \kappa_i,$$
$$w_{i,4} = \frac{4}{3h^2}\beta_i + \frac{2}{3h}\beta_i', \quad w_{i,5} = -\frac{1}{12h^2}\beta_i - \frac{1}{12h}\beta_i'.$$

The one-sided finite difference weights for the second and second last rows are given as

$$w_{11} = \frac{11}{12h^2}\beta_1 - \frac{1}{4h}\beta_1', \quad w_{12} = -\frac{5}{3h^2}\beta_1 - \frac{5}{6h}\beta_1' + \kappa_1, \quad w_{13} = \frac{1}{2h^2}\beta_1 + \frac{3}{2h}\beta_1',$$
$$w_{14} = \frac{1}{3h^2}\beta_1 - \frac{1}{2h}\beta_1', \quad w_{15} = -\frac{1}{12h^2}\beta_1 + \frac{1}{12h}\beta_1'.$$

$$w_{n_x-1,1} = -\frac{1}{12h^2}\beta_{n_x-1} - \frac{1}{12h}\beta_{n_x-1}', \quad w_{n_x-1,2} = \frac{1}{3h^2}\beta_{n_x-1} + \frac{1}{2h}\beta_{n_x-1}',$$
$$w_{n_x-1,3} = \frac{1}{2h^2}\beta_{n_x-1} - \frac{3}{2h}\beta_{n_x-1}', \quad w_{n_x-1,4} = -\frac{5}{3h^2}\beta_{n_x-1} + \frac{5}{6h}\beta_{n_x-1}' + \kappa_{n_x-1},$$
$$w_{n_x-1,5} = \frac{11}{12h^2}\beta_{n_x-1} + \frac{1}{4h}\beta_{n_x-1}'.$$

In the pursuit of an efficient solver for the aforementioned equation system, we turn our attention to the geometric multigrid approach. When coarsening the mesh by doubling the grid spacing (from h_l to h_{l+1}), the coefficient matrix $A_{h_{l+1}}$ and the right-hand side vector B_{l+1} are constructed by appropriately sampling values from the finer grid. Specifically, the β and κ values in $A_{h_{l+1}}$ and the corresponding entries in B_{l+1} at interior nodes are directly sampled from the refined grid. Similarly, the boundary values of B_{l+1} are collected from the boundary nodes of the refined grid. In fact, the coefficient matrix $A_{h_{l+1}}$ on the coarser mesh naturally forms a submatrix of A_{h_l} on the finer mesh, ensuring that the discretization consistently captures the variation in PDE coefficients β and κ across all grid levels.

The idea behind the multigrid method [8] is to reduce the high-frequency components of the solution error on the fine mesh, while addressing the low-frequency errors on coarser grids. Since low-frequency error components cannot be effectively reduced on fine grids, we restrict the residual to coarser grids, where error correction can be efficiently solved. These determined error correction solutions are subsequently interpolated back onto the fine grids, providing corrections to the already obtained fine mesh solution. Therefore, considerable computational time is saved by doing major computational work on the coarse grids. Throughout this iterative process, relaxation is applied on the fine mesh to enhance solution accuracy, while the restriction operator is used to transfer the residuals from the fine grid to the coarser grid and the interpolation operator interpolates the corrections from coarse grids to fine grids using the nearest coarse grid neighbors from coarse mesh to fine mesh. This process continues iteratively until the solution meets the desired accuracy.

In the implementation of the multigrid V-cycle scheme, as illustrated in **Algorithm 1**, we solve linear systems such as (9) using the Gauss-Seidel smoother as the relaxation method. The number of grid levels L in the multigrid algorithm is chosen such that the coarsest grid contains sufficient grid points to maintain the 5-point stencil required for fourth-order accuracy. To achieve the desired accuracy in the multigrid scheme, it is essential to ensure that the combined orders of the prolongation and restriction operators are at least equal to the order of the differential equation being solved, as established in [29]. Consequently, we employ bilinear prolongation (second-order) and full weighting restriction (second-order) to transfer information between grid levels.

The prolongation operator $I_{h_{l+1}}^{h_l}$ maps data from the coarser grid $l+1$ to the finer grid l using bilinear interpolation in two dimensions. This process is defined as follows:

$$U^{h_l}_{2i,2j} = U^{h_{l+1}}_{i,j},$$

$$U^{h_l}_{2i+1,2j} = \frac{1}{2}\left(U^{h_{l+1}}_{i,j} + U^{h_{l+1}}_{i+1,j}\right),$$

$$U^{h_l}_{2i,2j+1} = \frac{1}{2}\left(U^{h_{l+1}}_{i,j} + U^{h_{l+1}}_{i,j+1}\right),$$

$$U^{h_l}_{2i+1,2j+1} = \frac{1}{4}\left(U^{h_{l+1}}_{i,j} + U^{h_{l+1}}_{i+1,j} + U^{h_{l+1}}_{i,j+1} + U^{h_{l+1}}_{i+1,j+1}\right).$$

The restriction operator $R^{h_{l+1}}_{h_l}$ transfers data from the finer grid l to the coarser grid $l+1$. Using full weighting, the restriction in two dimensions is defined as:

$$\begin{aligned} U^{h_{l+1}}_{i,j} = \frac{1}{16}\Big(& 4U^{h_l}_{2i,2j} + 2\big(U^{h_l}_{2i-1,2j} + U^{h_l}_{2i+1,2j} + U^{h_l}_{2i,2j-1} + U^{h_l}_{2i,2j+1}\big) \\ & + \big(U^{h_l}_{2i-1,2j-1} + U^{h_l}_{2i+1,2j+1} + U^{h_l}_{2i+1,2j-1} + U^{h_l}_{2i-1,2j+1}\big)\Big). \end{aligned}$$

Algorithm 1 Multigrid V-cycle Scheme [8]

```
1: Set the initial guess for $U_{h_1}$ as $\mathbf{0}$, where $h_1$ represents the finest grid level.
2: Compute the initial residual on the finest grid: $r_{h_1} = B_{h_1} - A_{h_1}U_{h_1}$.
3: while $L_2$ error of the residual $r_{h_1}$ > tolerance do
4:     Pre-smooth the equation $A_{h_1}U_{h_1} = B_{h_1}$ using $v_1$ smoothing steps on the finest grid.
5:     for each grid level $l = 1, 2, \ldots, L-1$ do
6:         Compute the residual on the current level: $r_{h_l} = B_{h_l} - A_{h_l}U_{h_l}$.
7:         Restrict the residual to the next coarser grid: $r_{h_{l+1}} = R^{h_{l+1}}_{h_l} r_{h_l}$, where $R^{h_{l+1}}_{h_l}$ is the restriction operator, and $h_{l+1} = 2h_l$.
8:         Initialize $U_{h_{l+1}}$ on the coarser grid as $\mathbf{0}$.
9:     end for
10:    Solve the equation $A_{h_L}U_{h_L} = r_{h_L}$ exactly on the coarsest grid ($L$).
11:    for each grid level $l = L-1, L-2, \ldots, 1$ do
12:        Interpolate the correction to the finer grid: $e_{h_l} = I^{h_l}_{h_{l+1}} U_{h_{l+1}}$, where $I^{h_l}_{h_{l+1}}$ is the prolongation operator.
13:        Apply the correction to the finer grid:$U_{h_l} := U_{h_l} + e_{h_l}$.
14:        Post-smooth the equation $A_{h_l}U_{h_l} = B_{h_l}$ using $v_2$ smoothing steps on grid $h_l$.
15:    end for
16:    Update the residual on the finest grid: $r_{h_1} = B_{h_1} - A_{h_1}U_{h_1}$.
17: end while
```

For the practical implementation of our multigrid algorithm, we set both v_1 and v_2 equal to 2. The exact solution on the coarsest grid is obtained using the Gauss-Seidel iterative solver. The iteration stops when the error tolerance reaches 10^{-12} or when it reaches a maximum of 5000 iterations. Besides, the multigrid stopping criteria is the residual L_2 error less than 10^{-12}.

2.2. Corrected fourth order finite differences. We now switch our attention to the interface treatment. The finite difference formulation presented above, as given by equations (5), (6), and (7), is valid for regular points. However, when dealing with irregular points, a special MIB treatment is required because the function's regularity breaks down across the interface. In the AMIB method, corrected differences will be employed, which enable us to maintain the desired order of accuracy.

In our prior study, we addressed the correction of central difference approximations for second-order derivatives, as detailed in [20]. In this work, the correction for $\nabla \cdot (\beta \nabla u)$ involves treatments for both first and second-order derivatives. It is worth noting that the one-sided finite difference scheme requires no correction since the irregular points are located near the interface, rather than close to the boundary. We will present the correction for $(\beta u_x)_x$ only, and the y-direction correction can be conducted similarly.

For simplicity, denote δ_{xx} and δ_x as the central difference operators for the second derivative and first derivative, respectively,

$$\begin{aligned}\delta_{xx}(x_i) = &-\frac{1}{12h^2}u(x_{i-2}) + \frac{4}{3h^2}u(x_{i-1}) \\ &- \frac{5}{2h^2}u(x_i) + \frac{4}{3h^2}u(x_{i+1}) - \frac{1}{12h^2}u(x_{i+2}),\end{aligned} \tag{10}$$

$$\delta_x(x_i) = \frac{1}{12h}u(x_{i-2}) - \frac{2}{3h}u(x_{i-1}) + \frac{2}{3h}u(x_{i+1}) - \frac{1}{12h}u(x_{i+2}). \tag{11}$$

For the present study, it is natural to assume u as a piecewise smooth function of x, with a discontinuity at $x = \alpha$, where the interface Γ intersects the x grid line. Let $x_i \le \alpha \le x_{i+1}, h^- = x_i - \alpha$, and $h^+ = x_{i+1} - \alpha$. If the Cartesian jumps are known, we have the corrected Taylor expansions at two irregular points x_{i+i_2} and x_{i-i_1} as below [50]:

$$\begin{aligned}u(x_{i+i_2}) = &\sum_{k=0}^{K} \frac{[(i_2+i_1)h]^k}{k!} u^{(k)}(x_{i-i_1}) \\ &+ \sum_{k=0}^{K} \frac{[(i_2-1)h+h^+]^k}{k!} [u^{(k)}]_\alpha + O(h^{K+1}),\end{aligned} \tag{12}$$

and

$$\begin{aligned}u(x_{i-i_1}) = &\sum_{k=0}^{K} \frac{[(i_1+i_2)h]^k}{k!} u^{(k)}(x_{i+i_2}) \\ &- \sum_{k=0}^{K} \frac{[-i_1 h+h^-]^k}{k!} [u^{(k)}]_\alpha + O(h^{K+1}),\end{aligned} \tag{13}$$

where $[u^{(k)}]_\alpha = \lim_{x\to\alpha^+} u^{(k)}(x) - \lim_{x\to\alpha^-} u^{(k)}(x)$. The indices $i_2 \ge 1$ and $i_1 \ge 0$ represent index increments, with i denoting the index location. Based on the jump-corrected Taylor expansions (12) and (13), the fourth-order finite difference scheme can be corrected according to the following theorem.

Theorem 1. *Corrected fourth order finite differences.* Suppose $u \in C^6[x_i - 2h, \alpha) \bigcap C^6(\alpha, x_{i+1}+2h]$, with derivative extending continuously up to the interface

α. Then the following approximations hold to $O(h^4)$ when $K = 5$:

$$(\beta u_x)_x(x_{i-1}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i-1}) + (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i-1})\sum_{k=0}^{K}\frac{(h^+)^k}{k!}[u^{(k)}]_\alpha,$$

$$(\beta u_x)_x(x_i) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_i) - (\frac{4}{3h^2}\beta + \frac{2}{3h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h^+)^k}{k!}[u^{(k)}]_\alpha$$
$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h+h^+)^k}{k!}[u^{(k)}]_\alpha,$$

$$(\beta u_x)_x(x_{i+1}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i+1}) + (\frac{4}{3h^2}\beta - \frac{2}{3h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h^-)^k}{k!}[u^{(k)}]_\alpha$$
$$- (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h^- - h)^k}{k!}[u^{(k)}]_\alpha,$$

$$(\beta u_x)_x(x_{i+2}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i+2}) - (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i+2})\sum_{k=0}^{K}\frac{(h^-)^k}{k!}[u^{(k)}]_\alpha,$$

To achieve fourth-order accuracy in our computations, it is necessary to consider jump quantities up to the fifth-order derivative, thereby achieving a truncation error of $O(h^4)$. However, for practical purposes and expedited calculations, we find that setting $K = 4$ in the fourth-order corrected differences introduces a manageable third-order local truncation error of $O(h^3)$ when dealing with irregular points. This compromise allows us to maintain a desirable global fourth-order convergence rate while simplifying the computational complexity.

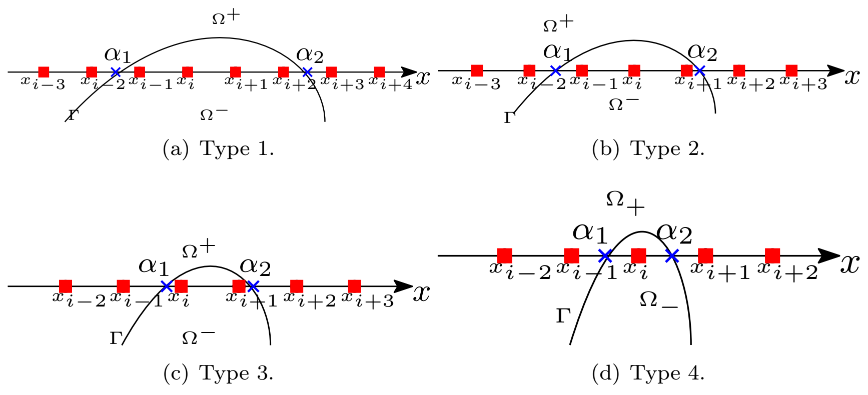


FIGURE 3. All possible corner scenarios in correcting the fourth order central difference. Red squares represent the irregular points, while blue crosses stand for interface intersection points.

The application of corrected fourth-order finite differences assumes a grid with sufficient resolution to accurately represent a smoothly curved interface. In practical scenarios, a Cartesian grid line may intersect the interface twice within a relatively short distance, resulting in two adjacent intersection points, denoted as α_1 and α_2 (as shown in Fig. 3). When there are at least four grid points available between α_1 and α_2, as shown in Figure 3 (a), Theorem 1 can be safely applied to correct the finite differences of $(\beta u_x)_x$ separately for both interface intersection points, and this situation is referred to as Type 1. However, in cases where there are only three or two grid points between α_1 and α_2 (as depicted in Fig. 3 (b) and (c)), we classify these scenarios as Type 2 and 3, respectively. In these cases, we can utilize jump-corrected Taylor expansions, as expressed in (12) and (13), to derive finite difference corrections for $(\beta u_x)_x$ at these irregular points. During the derivation of corrected finite differences at irregular points between α_1 and α_2, some grid points involved in the stencil of (12) and (13) might be positioned on different sides of either α_1 or α_2 from irregular points. As a result, the corrected finite differences at irregular points in between α_1 and α_2 will incorporate the Cartesian derivative jumps occurring at both intersection points of the interface. The corrections of fourth-order central difference in the case of Type 2 and 3 are presented in Theorem 2 and 3 in Appendix A.

Moreover, in the scenario illustrated in Figure 3 (d), where only one grid point is situated between α_1 and α_2, we categorize this as Type 4. Similar to Type 2 and 3, the corrected finite differences at irregular points in between α_1 and α_2 will involve the Cartesian derivative jumps occurring at both intersection points of the interface. For some other irregular points in this corner case, where the central differences stencil intersects with the interface twice, a new correction is required to achieve fourth-order finite differences. For example, for the corrected finite differences at x_{i-1}, even though both x_i and x_{i+1} are located on the right side of α_1, the jump-corrected Taylor expansion in Equation (12) is applied at x_i, while a classical Taylor expansion is employed at x_{i+1}. This is because x_{i+1} falls on the same side of the interface as x_{i-1}, obviating the need for a correction term. In Appendix A, Theorem 4 presents the corrected finite differences in Type 4 case.

The corrected differences discussed above focus on x-direction derivatives. Analogously, corrected differences for y-direction derivatives can be established. Moreover, these corrected differences simplify to (5) when no interface is encountered. The key to such corrected differences is the Cartesian derivative jumps at various interface points. The analytical values of such jumps are not available. We have to resort to numerical approximations to apply these corrections effectively at irregular points. In the subsequent subsection, we present a systematic procedure for reconstructing the Cartesian derivative jumps by using the AMIB method.

2.3. Cartesian derivative jumps reconstruction. The Cartesian derivative jump at $x = \alpha$ involved in the aforementioned corrected differences is defined as:

$$[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} = \lim_{x\to\alpha^+} \frac{\partial^k u}{\partial x^k} - \lim_{x\to\alpha^-} \frac{\partial^k u}{\partial x^k}, \tag{14}$$

where k takes values from 0 to 4. It is important to note that the one-sided limits considered here are the right-hand side limit $x \to \alpha^+$ and the left-hand side limit $x \to \alpha^-$. These superscripts are distinct from those used within the inside subdomain Ω^- and outside subdomain Ω^+.

The zeroth derivative in Eq. (14) represents the jump in the function across the interface. It the present problem, it can be analytically determined using Eq. (2). For cases where $k > 0$, the Cartesian derivative jumps usually cannot be obtained from the given PDE problem at the interface intersection points between Γ and grid lines. Subsequently, they have to be reconstructed numerically. This reconstruction involves two components. First, by assuming some necessary fictitious values, Cartesian derivative jumps are calculated by using finite differences. Second, the needed fictitious values will be generated by the ray-casting MIB scheme.

2.3.1. Approximation to Cartesian derivative jumps. We first consider how to numerically approximate Cartesian derivative jumps at α, i.e., $[\frac{\partial^k u}{\partial x^k}]_\alpha (k = 1, 2, 3, 4)$ with the aid of MIB fictitious values. To approximate the one-sided limits in Eq. (14), we construct Lagrange polynomials of degree 4, as illustrated in Fig. 4. As demonstrated in Fig. 1, two layers of fictitious nodes are assumed on both sides of the interface. For each side limit, Lagrange polynomials are derived using three real values on one side and two fictitious values on the other side of the interface. By taking derivatives of these one-sided polynomials, we can compute the various order derivatives needed.

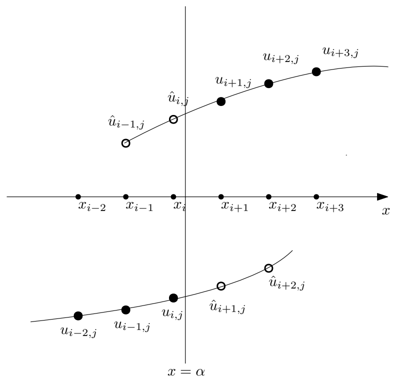


FIGURE 4. An illustration of the fourth order numerical approximation to Cartesian derivative jumps by two polynomials near the interface $x = \alpha$. The black dots stand for the real values, while the circles denote the fictitious values.

In the scenario depicted in Fig. 4, where the fictitious values are denoted with a hat notation, we approximate the Cartesian derivative jumps using the following expression

$$\begin{aligned}[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} \approx & \; (w^k_{i-1,j}\hat{u}_{i-1,j} + w^k_{i,j}\hat{u}_{i,j} + \sum_{l=1}^{3} w^k_{i+l,j}u_{i+l,j}) \\ & - (\sum_{l=1}^{3} w^k_{i-3+l,j}u_{i-3+l,j} + w^k_{i+1,j}\hat{u}_{i+1,j} + w^k_{i+2,j}\hat{u}_{i+2,j}),\end{aligned} \tag{15}$$

where $w^k_{p,q}$ represents the finite difference (FD) weights approximating at $x = \alpha$, and (p, q) stands for the grid location (x_p, y_q). As a result, we can calculate each Cartesian derivative jump using fictitious values surrounding the intersection points on the interface. Similarly, the approximation for the derivative jump in the y-direction can be established in a similar manner.

As previously discussed in the last subsection, it is essential to monitor grid resolution concerning interface changes when applying corrected differences. The same consideration applies to approximating derivative jumps. In [45], all potential corner cases related to reconstructing derivative jumps for elliptic interface problems have been thoroughly examined. Interested readers are directed to Subsection 2.3.3 of Ref. [45] for a detailed description of corner treatments to approximate Cartesian derivative jumps. To implement the Cartesian derivative jump approximation for all cases, it is necessary to construct fictitious values at all irregular points. In the following subsection, we will utilize the ray-casting MIB method to generate the required fictitious values.

2.3.2. Fictitious values formulation and ray-casting MIB method. To address the high-order central difference approximation at irregular points, the MIB scheme employs four layers of fictitious values in the vicinity of the interface, as described in [56, 52]. Achieving high-order accuracy does not necessitate high-order jump conditions. Instead, we consistently apply zeroth and first-order jump conditions, as provided in Eqns. (2) and (3), to compute all the fictitious values. These fictitious values can be viewed as smooth extensions of the solution across the interface Γ.

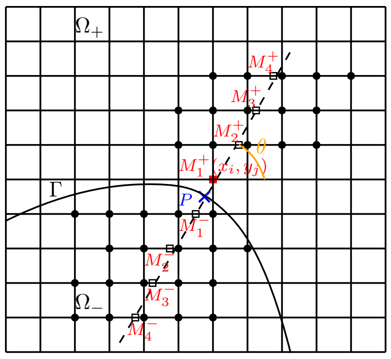


FIGURE 5. In the ray-casting MIB scheme, the desired fictitious value at $M_1^+ = (x_i, y_j)$(red filled squares) is generated along the normal line passing M_1^+ (dashed line), involving 8 auxiliary points M_l^- and M_l^+ for $l = 1, 2, 3, 4$ (empty squares). Filled circles stand for the chosen Cartesian grid points to interpolate or extrapolate each auxiliary point except for M_1^+.

In this study, we will apply the ray-casting MIB scheme [45] to determine fictitious values around the curved interface Γ. In the ray-casting MIB method, fictitious values will be computed in a 1D manner by imposing the interface conditions along the normal line at the corresponding interface intersection point. This approach differs from the Cartesian MIB method outlined in [21], where the jump conditions Eqns. (2) and (3) are recast into Cartesian directions, and approximated along grid lines.

We first illustrate how to calculate a fictitious value by using the ray-casting MIB treatment. Referring to the fictitious node $M_1^+(x_i, y_j)$ in Fig.5, we begin by identifying a normal line of Γ that passes through $M_1^+ = (x_i, y_j)$, by a procedure similar to IIM [43]. We assume that this normal line is perpendicular to the interface Γ at an interface intersection point P. Denote the angle between the normal line and the positive x direction to be θ. As we move along the inward normal direction, this normal line, or auxiliary line, first intersects an x-grid line at an intersection point M_1^-, as shown in Fig. 5. Extending this auxiliary line in both directions, it intersects other x-grid lines, giving us three auxiliary nodes M_l^+ for $l = 2, 3, 4$, all located on the same side of Γ, i.e., Ω^+. Similarly, four auxiliary nodes M_l^- on four x-grid lines $y = y_{j+l}$ for $l = 1, 2, 3, 4$ are situated on the other side of Γ_1, i.e., Ω^-.

We proceed by discretizing the zeroth and first-order jump conditions (2) and (3) at the point P using eight auxiliary points: $M_4^-, M_3^-, M_2^-, M_1^-, M_1^+, M_2^+, M_3^+, M_4^+$. We denote the function values at M_l^+ and M_l^- as $u_{M_l^+}$ and $u_{M_l^-}$ for $l = 1, 2, \cdots, 4$. This discretization enables us to determine two unknowns: $\hat{u}_{M_1^+} = \hat{u}_{i,j}$ and $\hat{u}_{M_1^-}$. The fictitious value $\hat{u}_{i,j}$ is the value required for the fourth-order finite difference approximation at a nearby irregular point, such as (x_i, y_{j-1}). Although the additional fictitious value $\hat{u}_{M_1^-}$ is not employed in this study, it holds potential value for future investigation involving more intricate geometries.

On the auxiliary line, we approximate each limiting value in the interface conditions (2) and (3) by utilizing four function values from the same side of Γ and one fictitious value from the opposite side of Γ. It is noted that the β values are known at the point P for both sides of the interface. With each limiting value properly approximated and without considering truncation errors at the level of $O(h^4)$, this results in two algebraic equations:

$$w_{0,1}^+ \hat{u}_{M_1^-} + \sum_{l=1}^{4} w_{0,l+1}^+ u_{M_l^+} - (w_{0,1}^- \hat{u}_{i,j,k} + \sum_{l=1}^{4} w_{0,l+1}^- u_{M_l^-}) = \phi(P), \tag{16}$$

$$\beta^+(P)(w_{1,1}^+ \hat{u}_{M_1^-} + \sum_{l=1}^{4} w_{1,l+1}^+ u_{M_l^+}) - \beta^-(P)(w_{1,1}^- \hat{u}_{i,j,k} + \sum_{l=1}^{4} w_{1,l+1}^- u_{M_l^-}) = \psi(P), \tag{17}$$

where $w_{m,l+1}^+$ and $w_{m,l+1}^-$ for $m = 0, 1$ and $l = 0, 1, \cdots, 4$ represent the finite difference weights. Here the subscript m stands for zeroth ($m = 0$) or first order derivative ($m = 1$) approximation at the interface point P. The superscripts $-$ and $+$ in w signify the Ω^- and Ω^+ domain separated by the interface Γ. Consequently, by solving the two algebraic equations (16) and (17), one can derive two fictitious values, namely $\hat{u}_{i,j}$ and $\hat{u}_{M_1^-}$. These values are obtained as linear combinations of $u_{M_l^+}$ and $u_{M_l^-}$ for $l = 1, 2, \cdots, 4$, and two jump values $\phi(P)$ and $\psi(P)$.

Furthermore, to ensure a complete Cartesian grid approach, we aim to determine fictitious values using function values from the grid nodes. To clarify, for each of the seven auxiliary points, M_l^- for $l = 1, 2, 3, 4$ and M_l^+ for $l = 2, 3, 4$, we employ the values of the five nearest grid nodes to interpolate or extrapolate the auxiliary point along each x-axis, as illustrated in Fig. 5. Consequently, each function value at an auxiliary point can be expressed in terms of grid node values.

Once we have accurately approximated the seven auxiliary function values, the formulation of the fictitious value $\hat{u}_{i,j}$ can be expressed in a general form as follows:

$$\hat{u}_{i,j} = \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 \phi(P) + W_1 \psi(P), \tag{18}$$

This representation of fictitious values guarantees an accuracy on the order of $O(h^4)$. $W_{I,J}$ represent the weight associated with each point in the set $\mathbb{S}_{i,j}$. This set comprises 5 grid nodes for each of the seven auxiliary points, along with $M_1^+ = (x_i, y_j)$, which are involved in the ray-casting MIB scheme. Consequently, the fictitious value $\hat{u}_{i,j}$ is a linear combination of 36 function values on 36 Cartesian nodes and two known jump values $\phi(P)$ and $\psi(P)$. Similarly, we can generate all the necessary fictitious values around Γ, including two layers inside and two layers outside.

The ray-casting MIB scheme introduces several notable advantages when compared to the classical Cartesian MIB scheme [56, 51]. Firstly, the ray-casting MIB scheme takes advantage of the normal direction to access a broader range of grid points both inside and outside the interface. This enhanced coverage makes the ray-casting MIB scheme more robust when dealing with complex geometric interface shapes. Moreover, in the ray-casting MIB approach, jump conditions are applied along the normal direction, simplifying the overall implementation. This differs from the traditional MIB scheme, which requires the decomposition of jump conditions into x, y, and z directions for finite difference approximations. Another benefit is the mitigation of corner problems. In the classical fourth-order MIB scheme, generating four fictitious values along one Cartesian direction can lead to corner problems, However, with sufficiently high grid resolution, the ray-casting MIB scheme avoids such corner problems, making it more reliable. In summary, the ray-casting MIB scheme offers a simpler and more robust solution for addressing problems involving PDEs with interfaces.

Remark 2.1. *The choice between two scenarios for auxiliary line intersections with x or y grid lines depends on the relationship between the mesh angle,* $\arctan(h_y/h_x)$*, and the angle θ. Specifically, when $h_x = h_y$ and θ falls within $\frac{\pi}{4} \leqslant \theta \leqslant \frac{3\pi}{4}$ or $\frac{5\pi}{4} \leqslant \theta \leqslant \frac{7\pi}{4}$ (as illustrated in Fig. 5), the auxiliary line meets the x grid lines first. In this case, the intersection of the auxiliary line with four x grid lines yields eight auxiliary nodes. Otherwise, when θ falls outside these ranges, the eight auxiliary points are chosen as the intersection points of the auxiliary lines with the y grid lines. This provides the necessary grid points for formulating fictitious values similarly.*

Remark 2.2. *In our computation, we adaptively generate auxiliary nodes based on the local interface geometry. This adaptivity is necessary because certain x or y grid lines may need to be skipped due to the unavailability of grid nodes within Ω^- or Ω^+ for interpolating specific auxiliary nodes. In such cases, the new auxiliary node is determined as the intersection of the auxiliary line with the next available x or y grid line. This essentially involves a downward shift of all relevant nodes along a particular grid line. This situation typically arises during the calculation of the second layer of fictitious values. For example, in Fig. 5, if a fictitious point like (x_i, y_{j+1}) is within Ω^+, the closest auxiliary point generated in the inward normal direction might still be within Ω^+. In this case, we retain this auxiliary point and only need two additional auxiliary points in Ω^+ above (x_i, y_{j+1}). However, when dealing with auxiliary points in Ω^-, we employ an adaptive downward shift process to locate the four necessary auxiliary regions within Ω^-. We repeat this downward shift process at most twice until we have enough grid nodes available on each grid line for interpolating the auxiliary nodes. Going beyond two shifts would place the chosen grid nodes far from the intended fictitious point, leading to a loss of accuracy to some extent.*

2.4. Augmented system.

In the AMIB method, we treat derivative jumps as auxiliary variables that are solved simultaneously with the unknown function values, thereby creating a unified augmented system. In solving the augmented system, we can decouple the solutions of auxiliary variables and unknown function values in different stages. This essentially allows us solve an elliptic subproblem without interfaces, and the proposed fourth order multigrid algorithm can be applied to efficiently invert the Laplacian discretization matrix. To construct this augmented system, we denote N_1 as the total number of grids across the entire rectangular domain Ω, and N_2 represents the total number of interface intersection points in both the x and y directions.

In the previous subsection, we reconstructed derivative jumps by incorporating fictitious values at each interface intersection point. By introducing these derivative jumps as auxiliary variables and substituting the fictitious value representation Eq. (18) into Eq. (15), we arrive at a generic linear equation:

$$\sum_{(x_I,y_J)\in\mathbb{S}i,j} C_{I,J}u_{I,J} + [\frac{\partial^k u}{\partial x^k}] = C_0\phi + C_1\psi, \tag{19}$$

Here, $C_{I,J}$ represents the weights of the function value $u_{I,J}$ in the approximation of the jump quantity $[\frac{\partial^k u}{\partial x^k}]$, while ϕ and ψ are the known interface data. Similar formulas to Eq. (19) can be obtained for all interface intersection points in the x and y directions.

Furthermore, we define a 1D column vector Q with dimensions $5N_2 \times 1$, representing the auxiliary variables introduced as $[\frac{\partial^k u}{\partial x^k}]_i$ and $[\frac{\partial^k u}{\partial y^k}]_j$ for $k = 0, 1, \cdots, 4$ at interface intersection points $i = 1, 2, \cdots$ and $j = 1, 2, \cdots$, totaling N_2 points. The unknown function values within the domain Ω, totaling N_1 grids, are arranged in another 1D column vector U with dimensions $N_1 \times 1$. Generalizing Eq.(19) from a single interface point to all interface points yields the matrix form of Eq.(19):

$$CU + IQ = \Phi, \tag{20}$$

In this equation, C represents a sparse matrix of dimensions $5N_2 \times N_1$, I is the identity matrix of dimensions $5N_2 \times 5N_2$, and Φ is a column vector of dimensions $5N_2 \times 1$ composed of known interface quantities.

For the elliptic problem described in Eq. (1), where $U_{i,j}$ denotes the discrete solution at (x_i, y_j), we discretize the PDE as follows:

$$L_hU_{i,j} + C_{i,j} = f_{i,j}, \quad 1 \le i \le n_x - 1, \quad 1 \le j \le n_y - 1 \tag{21}$$

Here, $C_{i,j}$ represents the correction term, and $L_hU_{i,j}$ represents the standard fourth-order central difference Eq. (5) or one-sided finite difference approximation Eq.(6) or Eq.(7) to $\nabla(\beta\nabla u)$, depending on the positions of interior nodes. This equation can be expressed in matrix form as:

$$AU + BQ = F, \tag{22}$$

In this equation, B is a sparse matrix with dimensions N_1 by $5N_2$, containing coefficients from correction terms, and F is a vector with dimensions $N_1 \times 1$ consisting of entries $f_{i,j}$. The Dirichlet boundary condition is incorporated into vector F by modifying some rows of matrix A, such that these rows are composed of the vector where only the diagonal element is one while all other entries are zero.

Coupling (22) and (20) yields an augmented system,

$$KW = R, \tag{23}$$

where

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

We compute BQ using the Schur complement, effectively eliminating U from Eq. (23) to create a linear system for Q:

$$(I - CA^{-1}B)Q = \varPhi - CA^{-1}F, \tag{24}$$

It is noted that the linear system (24) for Q has a much smaller degree of freedom compared to U, with a total of N_2. To efficiently solve for Q from Eq. (24), we propose an effective iterative solver that utilizes the proposed multigrid solver to handle A^{-1}. The implementation details for the Schur complement system (24) with a GMRES method are as follows:

(1) For the right-hand side (RHS), we determine $\Phi - CA^{-1}F$ by applying the multigrid solver on $A^{-1}F$ and performing some necessary arithmetic operations.
(2) For the left-hand side (LHS), we consider an iterative approach using the GMRES method. In this approach, we compute the matrix-vector product of $(I - CA^{-1}B)Q$ in several steps. Because it is equivalent to $IQ - CA^{-1}BQ$, we first apply a multigrid solver to the product BQ, i.e., $A^{-1}(BQ)$. This is followed by additional arithmetic operations on $IQ - CA^{-1}(BQ)$.
(3) We initialize the GMRES iteration with an initial guess of $Q = (0, 0, \cdots, 0)^T$. The GMRES iteration terminates either when it reaches the maximum iteration limit of 5000 or when the error tolerance reaches 10^{-15}. The GMRES solver employs a truncation parameter 100, meaning that the last 100 Krylov basis vectors are retained. The termination criteria are flexible and can be adjusted as needed for specific computational requirements.

After determining Q, the geometric multigrid algorithm is employed to solve the following equation and compute the solution U:

$$AU = F - BQ, \tag{25}$$

2.5. Three dimensional extension. The proposed two-dimensional AMIB algorithm for elliptic interface problems with variable coefficients can be easily extended to 3D. The proposed fourth-order finite differences and their corrections, as well as Cartesian derivative jump reconstruction, are all formulated in a 1D manner. In the augmented formulation, the multigrid method formulated for 1D elliptic system (8) can be simply applied to the matrix A in (23) for both 2D and 3D. Computationally, the only non-trivial extension is the ray-casting MIB for generating fictitious values. Fortunately, the 3D ray-casting MIB scheme has been developed in [45] for constant coefficient elliptic interface problems, which can be applied to the present study. The interface condition continues to be imposed along an auxiliary normal line in the 3D ray-casting MIB method. Compared with the 2D case, the difference is that each auxiliary node will be interpolated in a 2D plane, not along one grid line.

3. Numerical experiments

In this section, we will examine the accuracy and efficiency of the proposed AMIB method when applied to the solution of two or three dimensional elliptic equations characterized by variable coefficients. The performance of the proposed fourth order AMIB method will be compared with several MIB methods. For simplicity, the domain Ω will be assumed to be a square domain with equally spaced nodes $n = n_x = n_y = n_z$.

The numerical accuracy and convergence of the numerical solutions in $2D$ problems are tested by calculating errors under the maximum norm and L_2 norm defined as

$$L_\infty(u) = \max_{(x_i, y_j)\in\Omega} |u(x_i, y_j) - u_h(x_i, y_j)|,$$

$$L_2(u) = \sqrt{\frac{1}{(n+1)^2} \sum_{(x_i,y_j)\in\Omega} |u(x_i,y_j) - u_h(x_i,y_j)|^2},$$

where $u(x_i, y_j)$ and $u_h(x_i, y_j)$ are respectively analytical and numerical solutions inside the given computational domain Ω. The error norms in 3D can be similarly defined.

We will also investigate the accuracy of gradient approximations in 2D. Based on the calculated numerical solution $u_h(x_i, y_j)$, the fourth-order central differences, such as $\delta_x(x_i)$ defined in Eq. (11), will be employed in both x and y directions for $i = 2, 3, \dots, n_x - 2$ and $j = 2, 3, \dots, n_y - 2$. When $i = 1, n_x - 1$ or $j = 1, n_y - 1$, the one-sided fourth-order finite differences will be applied, which are essentially the first-derivative parts in Eqs. (6) and (7). When these fourth-order central difference stencils intersect the interface, the corresponding fictitious values will be supplied, just as in the original MIB scheme [56]. Then, the errors of gradient approximation can be measured in the maximum norm and L_2 norm

$$L_\infty(\nabla u) = \max_{(x_i,y_j)\in\Omega^-\cup\Omega^+} \max\{|\frac{\partial u}{\partial x}(x_i,y_j) - \frac{\partial u_h}{\partial x}(x_i,y_j)|, |\frac{\partial u}{\partial y}(x_i,y_j) - \frac{\partial u_h}{\partial y}(x_i,y_j)|\},$$

$$L_2(\nabla u) = \sqrt{\frac{1}{(n+1)^2} \sum_{(x_i,y_j)\in\Omega^-\cup\Omega^+} ||\nabla u(x_i,y_j) - \nabla u_h(x_i,y_j)||_2^2},$$

here $\nabla u(x_i, y_j)$ and $\nabla u_h(x_i, y_j)$ are respectively analytical and numerical gradient. The error norms of gradient approximation in 3D can be similarly estimated.

The convergence rate of the scheme will be examined by the formula

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

where $||E_i||$ is the numerical error on mesh of spacing h_i for $i = 1, 2$, using the above defined norms on $n+1$ by $n+1$ mesh for the computational domain Ω. For both the AMIB and MIB computations, the iteration numbers in the GMRES algorithm are reported in the discussion below.

All the experiments are carried out by using a single core on a Dell PowerEdge R7525 in The University of Alabama High-Performance Computer (UAHPC) (`https://oit.ua.edu/services/research/`) with AMD EPYC 7543 32-core CPUs operating at 2.8GHz clock speed.

3.1. Numerical examples in 2D. *Example 1.* We first consider a simple 2D Poisson's equation

$$(\beta u_x)_x + (\beta u_y)_y = f(x, y), \tag{26}$$

over a square $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with a circular interface Γ defined by $r^2 := x^2 + y^2 = 0.5^2$. The exact solution to this problem is prescribed as

$$u(x, y) = \begin{cases} \sin(kx)\sin(ky) & \text{in } \Omega^+ \\ e^{-x^2-0.5y^2} & \text{in } \Omega^-, \end{cases} \tag{27}$$

which is discontinuous across the interface Γ. Here the parameter k is chosen to be 3. The source term $f(x, y)$ is related to the above designated solution,

$$f(x, y) = \begin{cases} -2k^2\beta^+ \sin(kx)\sin(ky) + k(\beta_x^+ \cos(kx)\sin(ky) \\ \quad +\beta_y^+ \sin(kx)\cos(ky)) & \text{in } \Omega^+ \\ e^{-x^2-0.5y^2}((4x^2 + y^2 - 3)\beta^- - 2x\beta_x^- - y\beta_y^-) & \text{in } \Omega^-. \end{cases}$$

By setting $\beta^+ = 10^2$ and $\beta^- = 1$, Eq.(26) becomes a piecewise constant coefficient elliptic equation with a high-ratio jump. For such a problem, one usually divides both sides by β. This yields a simple Poisson equation $u_{xx} + u_{yy} = \frac{f(x,y)}{\beta}$. As the solution $u^+ = \sin(kx)\sin(ky)$ automatically satisfies the anti-symmetric property across the boundary of the given domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, which provides a foundation for FFT inversion. This allows us to compare the AMIB method with FFT (AMIB-FFT) [45] with the proposed AMIB method with Multigrid acceleration (AMIB-Multigrid). The numerical results of two methods are reported in Table 1. For both methods, the mesh size is taken to be $(n+1)\times(n+1)$, and different mesh refinements with corresponding $[n, n]$ values are reported. In Table 1, both methods exhibit fourth-order convergence, but AMIB-FFT is computationally more efficient than AMIB-Multigrid. However, when the anti-symmetric property is not satisfied across the boundary, AMIB-FFT requires additional techniques, such as introducing zero solutions outside the boundary [20], while AMIB-Multigrid does not. Furthermore, AMIB-Multigrid is capable of handling elliptic PDEs with variable coefficients, while AMIB-FFT faces challenges in such scenarios.

Next, we would like to investigate the performance of AMIB method with Multigrid acceleration for solving variable coefficient elliptic PDE. For this purpose, we redefine the diffusion coefficient β by

$$\beta = \begin{cases} e^{x+y} & \text{in } \Omega^+ \\ e^{-(x+y)} & \text{in } \Omega^-. \end{cases}$$

To demonstrate the efficiency of the AMIB-Multigrid, we will compare it with the fourth order MIB method, in which the calculated fictitious values by the ray-casting MIB scheme will be directly applied to the fourth-order finite difference approximations (5), (6), and (7), without using the corrected finite differences and the augmented system. The results of both MIB and AMIB methods are presented in Table 2. Both methods consistently achieve an average fourth-order convergence rate. In Table 2, the gradient approximation by the AMIB scheme is also reported. Based on the numerical solutions, the gradient can be simply estimated, and the corresponding average numerical order is also four. Additionally, we report the iteration numbers for both methods in the GMRES algorithm. It is evident that the AMIB exhibits slower growth in iteration numbers and is less dependent on the mesh size compared to the MIB. When the mesh size is doubled three times, the iteration number of the AMIB increases minimally, while that of the MIB becomes more than 14 times larger. Correspondingly, the CPU cost of the AMIB is significantly smaller than that of the MIB. In particular, on a mesh 256 by 256, the AMIB is about 22 times faster than the MIB.

Example 2. Besides the ray-casting method in the present AMIB method, the fictitious values can also be generated by using the classical Cartesian MIB scheme [21], which can then be further combined with the present augment formulation and multigrid solver, giving rise to a Cartesian AMIB scheme. It is interesting to compare such a Cartesian AMIB approach with the proposed ray-casting AMIB method. To this end, we consider again the 2D Poisson's equation (26). In this context, we define an interface Γ, which is constructed using a parametric function $\rho(\theta) = 0.5 + b\sin(k\theta)$. Here, the parameter b (where $b > 0$) governs the magnitude and curvature of the interface, while k (a positive integer) specifies the number of "heads" in the curve. The angle θ varies within the range $[0, 2\pi]$. We choose the parameter values $(b, k) = (0.13, 5)$, resulting in a five-headed interface. Additionally,

TABLE 1. Example 1a – Fourth order numerical error analysis of AMIB-FFT vs AMIB-Multigrid with $tol = 10^{-15}$ for Eq. (26) with $\beta^+ = 10^2$ and $\beta^- = 1$.

$[n_x, n_y]$	AMIB-FFT					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32]$	3.24E-05	–	1.50E-04	–	28	6.68E-03
$[64, 64]$	3.57E-06	3.182	1.42E-05	3.401	42	2.39E-02
$[128, 128]$	1.92E-07	4.2167	7.66E-07	4.2124	58	9.48E-02
$[256, 256]$	2.10E-08	3.1926	1.37E-07	2.4832	92	0.5105
$[n_x, n_y]$	AMIB-Multigrid					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32]$	3.74E-05	–	1.58E-04	–	43	6.29E-02
$[64, 64]$	3.68E-06	3.3453	1.45E-05	3.4458	40	0.210527
$[128, 128]$	1.90E-07	4.2800	7.34E-07	4.3041	76	1.5397
$[256, 256]$	2.11E-08	3.1707	1.36E-07	2.4322	102	7.9109

TABLE 2. Example 1b – Fourth order numerical error analysis of AMIB vs MIB with $tol = 10^{-15}$ for Eq. (26) with variable coefficient β.

$[n_x, n_y]$	AMIB Solution					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32]$	2.80E-04	–	1.27E-03	–	27	5.95E-02
$[64, 64]$	4.43E-06	5.9820	2.42E-05	5.7317	31	0.2655
$[128, 128]$	1.98E-07	4.4837	8.20E-07	4.8832	33	1.077
$[256, 256]$	6.93E-09	4.8365	4.78E-08	4.1005	43	5.6079
$[n_x, n_y]$	AMIB Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$[32, 32]$	1.10E-03	–	8.16E-03	–		
$[64, 64]$	2.20E-05	5.6439	1.85E-04	5.4630		
$[128, 128]$	9.30E-07	4.5641	1.67E-05	3.4696		
$[256, 256]$	5.12E-08	4.1830	2.17E-06	2.9441		
$[n_x, n_y]$	MIB Solution					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32]$	1.14E-04	–	5.94E-04	–	597	0.1288
$[64, 64]$	2.22E-06	5.6823	1.20E-05	5.6294	1509	1.2392
$[128, 128]$	1.26E-07	4.1391	8.10E-07	3.8890	4559	15.8649
$[256, 256]$	8.91E-09	3.8219	4.30E-08	4.2355	8737	123.7003

we define a diffusion coefficient β as follows

$$\beta = \begin{cases} \sin(x+y)+3 & \text{in } \Omega^+ \\ \cos(x+y)+2 & \text{in } \Omega^-. \end{cases} \tag{28}$$

and source term

$$f(x,y) = \begin{cases} -18\beta^+ \sin(3x)\sin(3y) \\ \quad +3(\beta_x^+ \cos(3x)\sin(3y) + \beta_y^+ \sin(3x)\cos(3y)) & \text{in } \Omega^+ \\ -18\beta^- \sin(3x)\cos(3y) \\ \quad +3(\beta_x^- \cos(3x)\cos(3y) - \beta_y^- \sin(3x)\sin(3y)) & \text{in } \Omega^-. \end{cases} \tag{29}$$

On the boundary of the domain $[-\frac{\pi}{3.5}, \frac{\pi}{3.5}] \times [-\frac{\pi}{3.5}, \frac{\pi}{3.5}]$, a Dirichlet boundary condition is assumed with the boundary data derived by the analytical solution

$$u(x,y) = \begin{cases} \sin(3x)\sin(3y) & \text{in } \Omega^+ \\ \sin(3x)\cos(3y) & \text{in } \Omega^-. \end{cases} \tag{30}$$

TABLE 3. Example 2– Fourth order numerical error analysis of the Ray-casting AMIB vs Cartesian AMIB with $tol = 10^{-15}$.

$[n_x, n_y]$	Ray-casting AMIB					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32]$	3.14E-05	–	1.60E-04	–	26	9.32E-02
$[64, 64]$	8.92E-07	5.1376	5.70E-06	4.8110	30	0.2354
$[128, 128]$	3.48E-08	4.6799	2.77E-07	4.3630	57	0.9984
$[256, 256]$	3.22E-09	3.4340	1.93E-08	3.8432	35	4.1216
$[n_x, n_y]$	Cartesian AMIB					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32]$	–	–	–	–	–	–
$[64, 64]$	3.73E-07	–	1.73E-06	–	32	0.2444
$[128, 128]$	3.30E-08	3.4986	1.46E-07	3.5667	57	0.9351
$[256, 256]$	6.53E-09	2.3373	3.84E-08	1.9268	64	4.2247

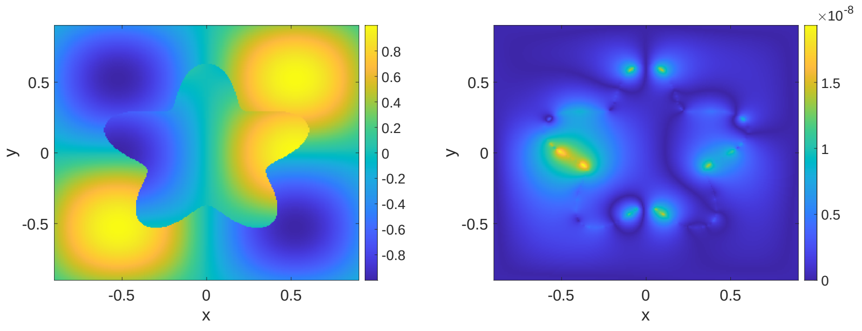


FIGURE 6. The numerical solution (left) and error (right) of Example 2 on a mesh with $n = 256$.

The numerical results of the ray-casting and Cartesian AMIB schemes are provided in Table 3. In terms of accuracy, the Cartesian AMIB scheme performs slightly better on coarser grids but exhibits reduced accuracy on finer meshes. Moreover, when the mesh size is set to $[n, n] = [32, 32]$, the Cartesian AMIB scheme encounters a specific corner issue, which results in its failure. This corner issue arises when a Cartesian grid line intersects the interface within a segment of length $2h$.

Unfortunately, for complex interfaces like the one in our current case, this corner issue could potentially arise at various mesh resolutions, particularly with coarser grids. In this particular instance, the failure becomes evident at $n = 32$, whereas the Cartesian AMIB scheme functions effectively for other values of n. By implementing the proposed corner treatments, the ray-casting AMIB scheme maintains fourth-order accuracy for both solution and gradient even when $[n, n] = [32, 32]$. In terms of efficiency, the ray-casting AMIB scheme is more efficient than the Cartesian AMIB scheme. The numerical solution and error for the ray-casting AMIB method are illustrated in Fig. 6 using $n = 256$. It is noted that the maximal errors are concentrated around the five-headed interface.

Example 3. we next examine an arch-shaped domain Ω^-, which is situated within a computational domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The interface Γ is composed of two half-circles: $\Gamma_1 = \{(x, y), y = \sqrt{0.3^2 - x^2}\}$ and $\Gamma_2 = \{(x, y), y = \sqrt{0.8^2 - x^2}\}$. The remaining portions of Γ are formed by the sides of two base columns, which take the shape of rectangles: $[-0.8, -0.3] \times [-0.8, 0]$ and $[0.3, 0.8] \times [-0.8, 0]$. Please refer to Fig. 7 for a visual representation. We study the elliptic PDE (4) with the same diffusion coefficient β given in (28), and we define the parameter κ as

$$\kappa(x,y) = \begin{cases} \sin(x+y)+1 & \text{in } \Omega^+ \\ \cos(x+y)+1 & \text{in } \Omega^-. \end{cases} \tag{31}$$

The exact solution to this problem is

$$u(x,y) = \begin{cases} \cos(kx)e^y & \text{in } \Omega^+ \\ \sin(kx)\cos(ky) & \text{in } \Omega^-. \end{cases} \tag{32}$$

and the parameter k is selected to be 3. The source term $f(x, y)$ is related to the above designated solution,

$$f(x,y) = \begin{cases} ((1-k^2)\beta^+ + \kappa^+ + \beta_y^+)\cos(kx)e^y & \\ \qquad -k\beta_x^+\sin(kx)e^y & \text{in } \Omega^+ \\ (-2k^2\beta^- + \kappa^-)\sin(kx)\cos(ky) & \\ \qquad +k(\beta_x^-\cos(kx)\cos(ky) - \beta_y^-\sin(kx)\sin(ky)) & \text{in } \Omega^-. \end{cases}$$

TABLE 4. Example 3– Fourth order numerical error analysis of AMIB with $tol = 10^{-15}$.

$[n_x, n_y]$	AMIB Solution					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
[32, 32]	2.31E-05	–	1.48E-04	–	24	5.62E-02
[64, 64]	2.89E-06	2.9988	1.91E-05	2.9540	32	0.2482
[128, 128]	2.79E-07	3.3727	1.41E-06	3.7598	33	0.9970
[256, 256]	5.78E-09	5.5931	2.87E-08	5.6185	34	4.0895
$[n_x, n_y]$	AMIB Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
[32, 32]	2.73E-04	–	3.06E-03	–		
[64, 64]	2.51E-05	3.4431	2.89E-04	3.4044		
[128, 128]	1.72E-06	3.8672	1.75E-05	4.0456		
[256, 256]	5.49E-08	4.9695	9.47E-07	4.2078		

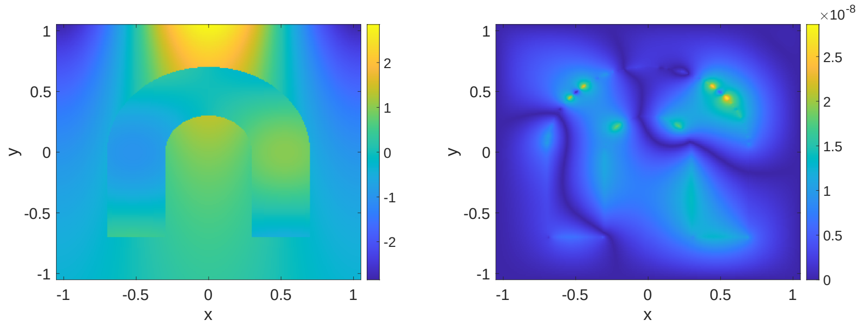


FIGURE 7. The numerical solution (left) and error (right) of Example 3 on a mesh with $n = 256$.

Table 4 demonstrates that the AMIB method remains effective in solving the elliptic equation (4), even when the reaction term contains variable coefficients. The results reaffirm an average fourth-order convergence for both solution and gradient. The efficiency of the AMIB is further validated. It is found in Fig. 7 that large errors usually occur on or near the interface.

Example 4. Reusing the analytical solution (30) and source term (29), we consider a complex domain Ω^- formed by the union of two distinct disks in $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. Notably, the embedded interface Γ comprises two distinct components: $\Gamma_1 = \{(x, y), x < 0, (x+0.2)^2+y^2 = 0.5^2\}$ and $\Gamma_2 = \{(x, y), x > 0, (x-0.2)^2+y^2 = 0.5^2\}$. The diffusion coefficient is defined as

$$\beta = \begin{cases} 2 - \cos(2x + y) & \text{in } \Omega^+ \\ 2 - \sin(x + y) & \text{in } \Omega^-. \end{cases}$$

and the coefficient in the reaction term

$$\kappa = \begin{cases} 1 - \cos(x + y) & \text{in } \Omega^+ \\ 1 + \sin(x + y) & \text{in } \Omega^-. \end{cases}$$

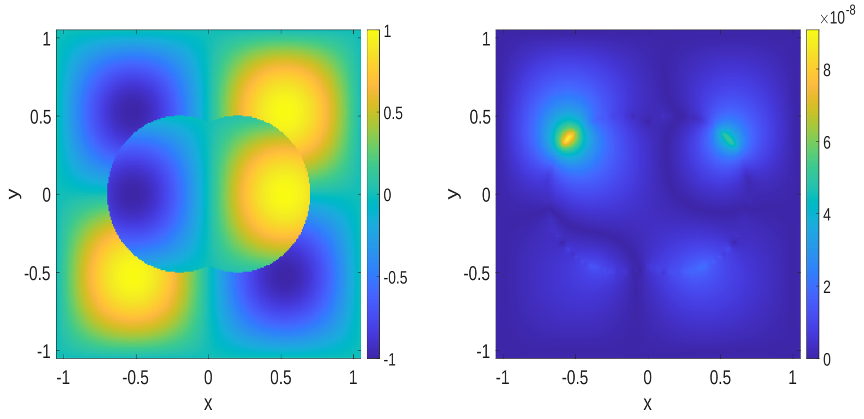


FIGURE 8. The numerical solution (left) and error (right) of Example 4 on a mesh with $n = 256$.

By taking the above coefficients κ and β, the numerical results are presented in Table 5. The fourth order convergence is attained for both solution and gradient. The numerical solution and error of the AMIB method for this example are

Table 5. Example 4– Fourth order numerical error analysis of AMIB with $tol = 10^{-15}$.

$[n_x, n_y]$	AMIB Solution					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32]$	8.03E-05	–	2.77E-04	–	26	5.30E-02
$[64, 64]$	2.92E-06	4.7814	1.41E-05	4.2961	34	0.2513
$[128, 128]$	1.57E-07	4.2171	1.21E-06	3.5426	58	1.6763
$[256, 256]$	1.08E-08	3.8617	9.13E-08	3.7282	38	4.7435
$[n_x, n_y]$	AMIB Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$[32, 32]$	4.82E-04	–	3.35E-03	–		
$[64, 64]$	1.84E-05	4.7113	1.83E-04	4.1942		
$[128, 128]$	1.15E-06	4.0000	1.83E-05	3.3219		
$[256, 256]$	7.52E-08	3.9348	1.87E-06	3.2907		
$[n_x, n_y]$	MIB Solution					
	L_2		L_∞			
	Error	Order	Error	Order		
$[32, 32]$	5.00E-05	–	1.76E-04	–	315	6.72E-02
$[64, 64]$	2.18E-06	4.5195	1.08E-05	4.0265	772	0.6273
$[128, 128]$	1.34E-07	4.0240	1.06E-06	3.3489	1760	6.0594
$[256, 256]$	7.98E-09	4.0697	6.15E-08	4.1073	4004	56.5940

presented in Fig. 8. Notably, the interface Γ exhibits non-smoothness at the two junction points of the two circles. It is worth mentioning that the AMIB scheme is specifically designed for C^1 continuous interfaces. Fortunately, in this particular example, the non-smoothness at these two cusps is not overly severe, and the AMIB method still delivers the desired order of accuracy. It is important to note that if the distance between the centers of the two disks were larger, the geometric singularity at these two cusps would become more pronounced. In such cases, achieving fourth-order convergence with the AMIB method or any other known methods would be unattainable. However, even in such challenging scenarios, the original MIB method could still maintain a second order of accuracy, as demonstrated in [51].

3.2. Numerical examples in 3D. *Example 5.* Consider a 3D Poisson's equation

$$(\beta u_x)_x + (\beta u_y)_y + (\beta u_z)_z = f(x, y, z) \tag{33}$$

over a cubic domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with a spherical interface defined by

$$\Gamma : x^2 + y^2 + z^2 = 0.4^2.$$

The analytical solution to this problem is constructed to be

$$u(x, y, z) = \begin{cases} \sin(kx)\sin(ky)\sin(kz), & \text{in } \Omega^+, \\ \sin(x + y + z), & \text{in } \Omega^-, \end{cases} \tag{34}$$

with $k = 3$. The source term $f(x, y, z)$ is related to the above designated solution,

$$f(x,y,z) = \begin{cases} (-3k^2\beta^+ + k(\frac{\beta_x^+}{\tan(kx)} + \frac{\beta_y^+}{\tan(ky)} + \frac{\beta_z^+}{\tan(kz)})\sin(kx)\sin(ky)\sin(kz), & \text{in } \Omega^+, \\ -3\beta^- \sin(x+y+z) + (\beta_x^- + \beta_y^- + \beta_z^-)\cos(x+y+z), & \text{in } \Omega^-. \end{cases}$$

On the boundary of the cubic domain, a Dirichlet boundary condition is assumed with the boundary data derived by the analytical solution. The diffusion coefficient is defined as

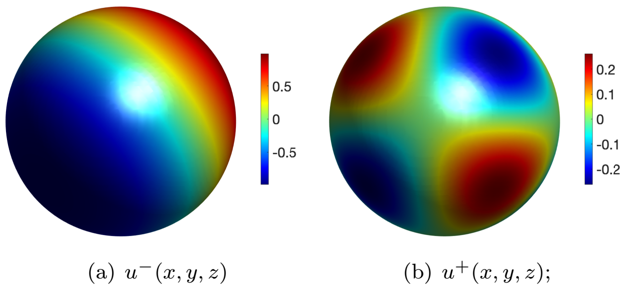
$$\beta = \begin{cases} e^{x+y+z} & \text{in } \Omega^+ \\ e^{-(x+y+z)} & \text{in } \Omega^-. \end{cases}$$

Table 6. Example 5– Fourth order numerical error analysis of AMIB with $tol = 10^{-15}$.

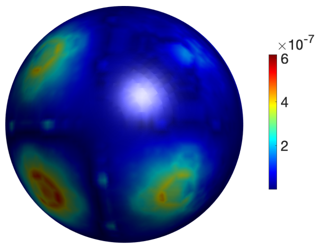
$[n_x, n_y, n_z]$	AMIB Solution					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32, 32]$	1.59E-05	–	2.65E-04	–	48	3.7126
$[64, 64, 64]$	5.51E-07	4.8508	1.23E-05	4.4293	54	33.4883
$[128, 128, 128]$	3.25E-08	4.0835	6.32E-07	4.2826	65	357.5614
$[256, 256, 256]$	2.27E-09	3.8397	5.18E-08	3.6089	94	4467.8754
$[n_x, n_y, n_z]$	AMIB Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$[32, 32, 32]$	2.46E-04	–	5.67E-03	–		
$[64, 64, 64]$	8.56E-06	4.8449	2.83E-04	4.3245		
$[128, 128, 128]$	4.55E-07	4.2337	6.35E-05	2.156		
$[256, 256, 256]$	2.44E-08	4.2209	3.77E-06	4.0741		
$[n_x, n_y, n_z]$	MIB Solution					
	L_2		L_∞			
	Error	Order	Error	Order		
$[32, 32, 32]$	1.45E-05	–	1.95E-04	–	1481	14.252783
$[64, 64, 64]$	5.67E-07	4.6766	1.31E-05	3.8958	3893	262.948881
$[128, 128, 128]$	3.33E-08	4.0898	8.26E-07	3.9873	8477	4517.941342
$[256, 256, 256]$	-	-	-	-	-	-

The analytical solutions, $u^-(x, y, z)$ and $u^+(x, y, z)$, are depicted in Fig. 9. The graphs are generated by mapping the solutions onto the spherical surface Γ from inside and outside, respectively, for $u^-(x, y, z)$ and $u^+(x, y, z)$. While the two solutions are smooth within their respective subdomains, the overall solution $u(x, y, z)$ exhibits a clear discontinuity across Γ.

The numerical results of both the MIB and AMIB schemes are reported in Table 6. In both schemes, the same mesh size is employed, i.e., $(n+1) \times (n+1) \times (n+1)$. The number of corner points is reported for each tested value of n. It is evident that the AMIB scheme achieves fourth-order accuracy, while the MIB scheme fails for $n = 256$. Similar to Example 1 in 2D, the AMIB method exhibits robust convergence and is less dependent on the mesh size n, resulting in significantly faster computations than the MIB method. The iteration count increases significantly for the MIB scheme, and on a $256 \times 256 \times 256$ mesh, the computation was terminated due to excessively long runtime. In Table 6, the gradient approximation by the



(a) $u^-(x, y, z)$ (b) $u^+(x, y, z)$;



(c) Error

FIGURE 9. The plots of exact solutions and numerical error of the ray-casting AMIB scheme by mapping onto the spherical surface Γ, based on a mesh with $n = 128$.

AMIB scheme is also reported. Based on the numerical solutions, the gradient can be estimated with a corresponding numerical order of four. The numerical error of the AMIB scheme at $n = 128$ is depicted in Fig. 9 (c), showing that large errors occur in regions where the solutions have large magnitudes.

Example 6. In this example, we focus on a molecular interface. Since the MIB scheme is not suitable for 3D computations due to its slow performance, we will exclusively consider the ray-casting AMIB scheme going forward. We study the 3D Poisson's equation (33) with a smooth molecular surface defined by two atoms as follows:

$$\Gamma : (x^2 + y^2 + z^2 + \frac{3}{5})^2 - \frac{5}{2}y^2 = \frac{3}{5}.$$

With the parameter $k = 3$, the exact solutions are constructed to be

$$u(x, y, z) = \begin{cases} \sin(kx)\sin(ky)\sin(kz), & \text{in } \Omega^+, \\ \cos(kx)\sin(ky)\cos(kz), & \text{in } \Omega^-, \end{cases} \tag{35}$$

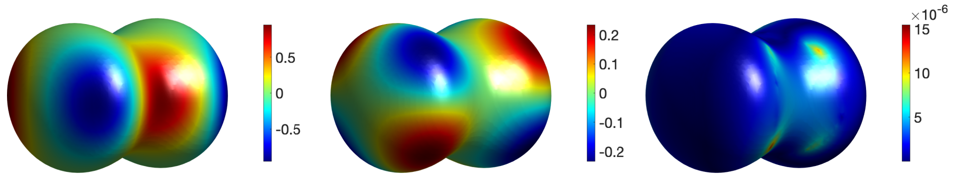
and the corresponding source terms are

$$f(x, y, z) = \begin{cases} (-3k^2\beta^+ + k(\frac{\beta_x^+}{\tan(kx)} + \frac{\beta_y^+}{\tan(ky)} + \frac{\beta_z^+}{\tan(kz)}) & \\ \quad \cdot \sin(kx)\sin(ky)\sin(kz), & \text{in } \Omega^+, \\ (-3k^2\beta^- + k(-\beta_x^- \tan(kx) + \frac{\beta_y^-}{\tan(ky)} - \beta_z^- \tan(kz)) & \\ \quad \cdot \cos(kx)\sin(ky)\cos(kz), & \text{in } \Omega^-, \end{cases}$$

Here, the diffusion coefficient is defined as:

$$\beta = \begin{cases} \cos(x + y + z) + 2 & \text{in } \Omega^+ \\ \sin(x + y + z) + 2 & \text{in } \Omega^-. \end{cases} \tag{36}$$

The plots of solutions u^- and u^+ mapped onto the molecular surface are given in Fig. 10.



(a) $u^-(x, y, z)$ (b) $u^+(x, y, z)$; (c) Error

FIGURE 10. The plots of exact solutions and numerical error of the ray-casting AMIB scheme by mapping onto the molecular surface Γ, based on a mesh with $n = 128$.

TABLE 7. Example 6 – Molecular interface of two atoms. The number of corner points is 8, 8, 0 and 8, respectively, for n =32, 64, 128, and 256.

$[n_x, n_y, n_z]$	AMIB Solution					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32, 32]$	2.33E-03	–	1.99E-02	–	46	3.5693
$[64, 64, 64]$	4.04E-05	5.8498	4.45E-04	5.4828	49	31.4859
$[128, 128, 128]$	2.46E-06	4.0376	5.64E-05	2.9800	84	448.9337
$[256, 256, 256]$	9.06E-08	4.7630	1.35E-06	5.3847	101	4413.5281
$[n_x, n_y, n_z]$	AMIB Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$[32, 32, 32]$	7.73E-03	–	1.12E-01	–		
$[64, 64, 64]$	1.96E-04	5.3015	6.54E-03	4.0986		
$[128, 128, 128]$	1.18E-05	4.0540	1.06E-03	2.6252		
$[256, 256, 256]$	4.42E-07	4.7386	5.12E-05	4.3718		

The numerical results are summarized in Table 7. As before, a straightforward Dirichlet boundary condition is applied, and the average fourth-order convergences in the solution and gradient are once more confirmed. Furthermore, the efficiency of the AMIB scheme is thoroughly validated. The error plot in Fig. 10 shows that the maximum errors are concentrated in regions where the most significant discontinuities occur across the interface.

Example 7. Reusing the diffusion coefficient (36), the last example is considered to tackle a 3D Helmholtz-type problem described by the equation:

$$(\beta u_x)_x + (\beta u_y)_y + (\beta u_z)_z + \kappa u = f(x, y, z) \tag{37}$$

with the analytical solution

$$u(x, y, z) = \begin{cases} \sin(3x)e^{y+z}, & \text{in } \Omega^+, \\ \cos(x + y + z), & \text{in } \Omega^-, \end{cases} \tag{38}$$

and source term

$$f(x,y,z) = \begin{cases} (-7\beta^+ + \beta_y^+ + \beta_z^+ + \kappa^+)\sin(3x)e^{y+z} + 3\beta_x^+\cos(3x)e^{y+z}, & \text{in } \Omega^+, \\ (-3\beta^- + \kappa^-)\cos(x+y+z) - (\beta_x^- + \beta_y^- + \beta_z^-)\sin(x+y+z), & \text{in } \Omega^-. \end{cases}$$

Here, the coefficient κ is defined as:

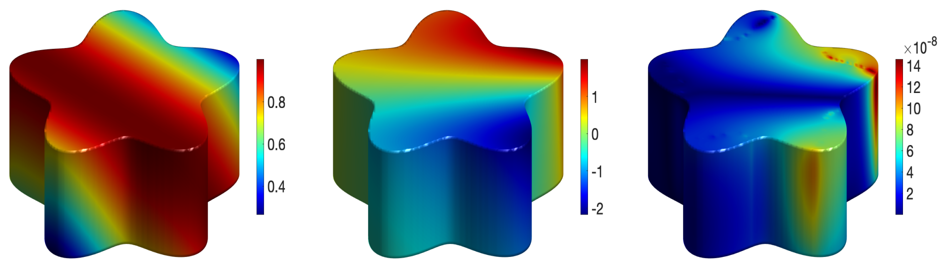
$$\kappa = \begin{cases} 1 - \cos(x+y+z) & \text{in } \Omega^+ \\ 1 + \sin(x+y+z) & \text{in } \Omega^-. \end{cases} \tag{39}$$

The computational domain spans $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, and it features a non-smooth flower-like interface [51] defined as follows

$$\Gamma : \rho(\theta) = 0.6 + 0.12\sin 5\theta, \quad -0.3 \leqslant z \leqslant 0.3,$$

TABLE 8. Example 7 – Flower interface. The number of corner points is 27, 57, 37, 73, respectively, for n =32, 64, 128, and 256.

$[n_x, n_y, n_z]$	AMIB Solution					
	L_2		L_∞		iter no.	CPU time (s)
	Error	Order	Error	Order		
$[32, 32, 32]$	1.21E-05	–	9.60E-05	–	70	5.2840
$[64, 64, 64]$	5.44E-07	4.4753	4.04E-06	4.5706	77	48.26074
$[128, 128, 128]$	3.05E-08	4.1567	1.67E-07	4.5964	101	522.1567
$[256, 256, 256]$	1.89E-09	4.0124	1.18E-08	3.8230	104	4557.9268
$[n_x, n_y, n_z]$	AMIB Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
$[32, 32, 32]$	1.79E-04	–	1.73E-03	–		
$[64, 64, 64]$	1.04E-05	4.1053	1.20E-04	3.8497		
$[128, 128, 128]$	6.68E-07	3.9606	7.81E-06	3.9416		
$[256, 256, 256]$	4.29E-08	3.9608	5.22E-07	3.9032		

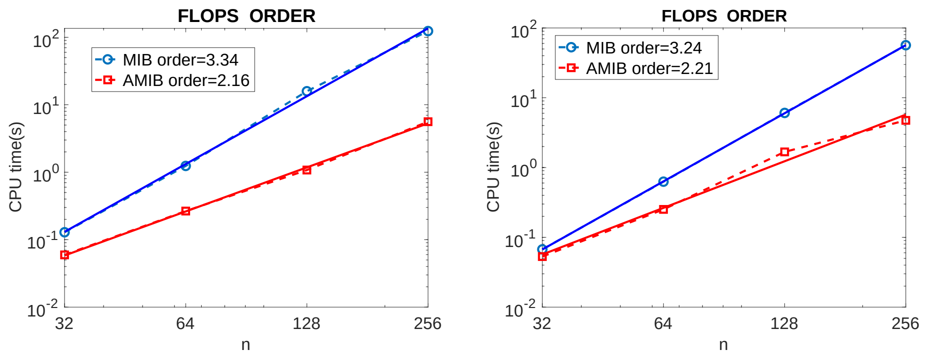


(a) $u^-(x,y,z)$ (b) $u^+(x,y,z)$; (c) Error

FIGURE 11. The plots of exact solutions and numerical error of the ray-casting AMIB scheme by mapping onto the flower surface Γ, based on a mesh with $n = 128$.

The numerical results for this example are provided in Table 8. It is worth noting that the horizontal cross-section of the 3D shape resembles the 2D five-leaf interface studied in Example 2. As a result, many fictitious values can be generated by employing ray-casting lines that are entirely contained within certain xy planes, essentially treating them in a 2D manner. In the case of the 3D flower-like interface, the AMIB scheme exhibits fourth-order convergence, as indicated in Table 8. The flower-like interface is visually represented in Fig. 11, which includes surface maps of both the exact solutions and numerical errors. Notably, significant numerical errors are observed in regions characterized by substantial curvatures.

3.3. Computational efficiency. In this subsection, we delve into the computational efficiency of the AMIB method when applied to 2D variable coefficient elliptic interface problems. To further assess their performance, we analyze the computational time of the AMIB and MIB methods as presented in Table 1 and Table 5, enabling us to quantify the numerical complexity of both methods. To visualize this comparison, Fig. 12 illustrates the computational efficiency of AMIB and MIB in 2D by plotting CPU time against n in a log-log manner. Here, we simplify the analysis by considering n as the degree of freedom in each direction. We employ a least squares fitting technique to represent the CPU time in the form of n^r. Fig. 12 indicates that the computational complexity of MIB is, at minimum, of the order $O(n^3)$. In contrast, for the AMIB scheme, the value of exponent r is slightly above 2. Based on these findings, we can conclude that the computational cost of the AMIB method in 2D exhibits a roughly quadratic order of complexity, approximately $O(n^2)$. Similarly, Fig.13 (a) compares the computational efficiency of AMIB and MIB for 3D elliptic interface problems. The r value is slightly above 3 for AMIB scheme and 4 for MIB scheme. Based on the computational time of the AMIB method reported in Table 7, it can be observed in Fig.13 (b) that the flop order r for the proposed AMIB method is slightly above 3. One can conclude that the computational cost of the proposed AMIB method is roughly $O(n^d)$ where d is the dimension number.



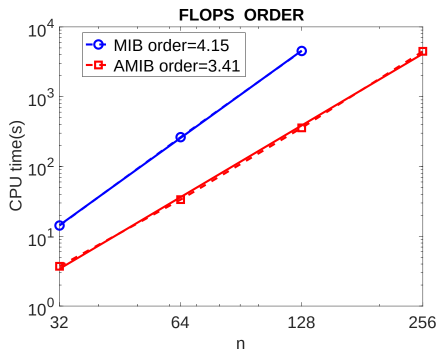
(a) Example 1.

(b) Example 4.

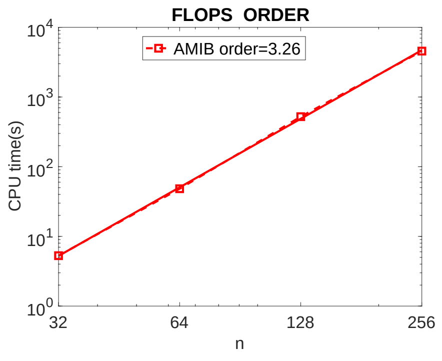
FIGURE 12. Flops order in CPU time is examined for several 2D examples. The numerical flop order r is slightly larger than 2.

4. Conclusion

In this study, we have introduced an efficient and fourth-order accurate finite difference method designed to address variable coefficient elliptic interface problems. For interior regular points, fourth-order central differences are employed



(a) Example 5.



(b) Example 7.

FIGURE 13. Flops order in CPU time is examined for several 3D examples. The numerical flop order r is slightly larger than 3.

to discretize the first and second order derivatives involved in the Laplacian with piecewise smooth coefficients, while one-sided finite differences are used near the boundary. The central differences are corrected near the interface, by using the fictitious value representation generated by the ray-casting MIB scheme. In the augmented formulation, Cartesian derivative jumps at interface points are treated as auxiliary variables, so that the finite difference matrix does not sense the interface. A fourth order accurate multigrid method is proposed to invert this finite difference matrix, in each iterative step of the Schur complement procedure for the augmented system. The iteration number of the Schur complement process only weakly depends on the mesh size n. Thus, the proposed ray-casting AMIB method not only achieves a fourth order of accuracy for complex interfaces, but also delivers an overall complexity of $O(n^3)$ on a $n \times n \times n$ mesh. Moreover, with a little additional computation, the AMIB method can produce fourth order accurate approximation of solution gradients.

The proposed AMIB method is believed to be the first interface algorithm developed in the literature for solving variable coefficient elliptic interface problems, that not only achieves a fourth-order convergence, but also delivers a multigrid or FFT efficiency. A comparison between the existing AMIB-FFT [21, 45] and the proposed AMIB-Multigrid method is in order.

- The AMIB-Multigrid can attack variable coefficient elliptic interface problems governed by Eq. (1), while the AMIB-FFT is limited to elliptic PDEs with piecewise constant coefficients. In particular, when the AMIB-FFT is applied to Eq. (1), β is required to be a piecewise constant and κ can only be a multiply of β. This is because the FFT algorithm cannot handle a finite difference matrix with non-constant diagonal coefficients. Thus, when κ is a piecewise constant that is independent of β, the AMIB-FFT algorithm is still not applicable. Similarly, the AMIB-FFT algorithm can not handle parabolic interface problems, which motivates the development of a second order AMIB-Multigrid method in [22]. The generalization of the proposed AMIB-Multigrid method to parabolic interface problems will be reported in the future.
- Both multigrid and FFT algorithms provide a fast Poisson solver for inverting the matrix obtained from a finite difference discretization of the

Laplacian. Their flops orders are similar, e.g. $O(n^3 \log n)$ and $O(n^3)$, respectively, for AMIB-FFT and AMIB-Multigrid in 3D. However, as demonstrated in the present study, when the AMIB-FFT method is applicable, it is always faster than the AMIB-Multigrid. This is because the multigrid solution of the finite difference matrix involves an iterative solution, while the FFT inversion can be regarded as a direct algorithm.

- The fourth order boundary treatments of the AMIB-FFT and AMIB-Multigrid are different. A zero-padding treatment was introduced in the AMIB-FFT algorithm [20] to handle all commonly used boundary conditions, including Dirichlet, Neumann, and Robin conditions. With this treatment, the anti-symmetric property holds over the boundary of the extended domain, so that the FFT inversion can be applied to the fourth-order central difference. However, this practice cannot be applied to the present multigrid method, because the underlying function still does not satisfy the anti-symmetric property. In particular, this property is no longer valid for coarser mesh levels involved in the multigrid procedure. To bypass this issue, one-sided finite differences are used near the boundary in the proposed AMIB-Multigrid method in treating the Dirichlet boundary condition. The accommodation of other boundary conditions in the AMIB-Multigrid solution will be explored in the future.

Acknowledgments

This research was supported by the National Science Foundation (NSF) under grants DMS-2110914 and DMS-2306991.

Declarations

Conflicts of interest: The authors declare no conflict of interest.
Data Availability Statement: Data sharing is not applicable to this article as no data sets were generated or analyzed during the current study.

References

[1] L. Adams, Z.L. Li, The immersed interface/multigrid methods for interface problems, SIAM J. Sci. Comput. 24(2002) 463-479.
[2] L. Adams, T.P. Chartier, New geometric immersed interface multigrid solvers, SIAM J. Sci. Comput. 25 (2004) 1516-1533.
[3] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, Computing, 5 (1970) 207-213.
[4] J. Bedrossian, J. H. von Brecht, S. W. Zhu, E. Sifakis, and J. M. Teran. A finite element method for interface problems in domains with smooth boundaries and interfaces, J. Comput. Phys. 229 (2010) 6405-6426.
[5] P.A. Berthelsen, A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions, J. Comput. Phys. 197 (2004) 364-386.
[6] D. Bochkov and F. Gibou, Solving elliptic interface problems with jump conditions on Cartesian grids, J. Comput. Phys. 407 (2020) 109269.
[7] J. Bramble and J. King, A finite element method for interface problems in domains with smooth boundaries and interfaces, Adv. Comput. Math. 6 (1996) 109-138.
[8] W. L. Briggs, V. E. Henson, and S. F. McCormick. A multigrid tutorial. Society for Industrial and Applied Mathematics, (2000).
[9] T.F. Chan, W. Wan, Robust multigrid methods for nonsmooth coefficient elliptic linear systems, J. Comput. Appl. Math. 123 (2000) 323-352.
[10] T. Chen and J. Strang, Piecewise-polynomial discretization and Krylov-accelerated multigrid for elliptic interface problems, J. Comput. Phys. 227 (2008) 7503-7542.

[11] X. Chen, X. Feng and Z. Li, A direct method for accurate solution and gradient computations for elliptic interface problems. Numer. Algorithms 80 (2019) 709-740.

[12] Z. Chen and J. Zou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numer. Math. 79 (1998) 175-202.

[13] I.-L. Chern and Y.-C. Shu, A coupling interface method for elliptic interface problems, J. Comput. Phys. 225 (2007) 2138-2174.

[14] A. Coco, G. Russo, Finite-difference ghost-point multigrid methods on Cartesian grids for elliptic problems in arbitrary domains, J. Comput. Phys. 241 (2013) 464-501.

[15] A. Coco, G. Russo, Second order finite-difference ghost-point multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface, J. Comput. Phys. 361(2018) 299-330.

[16] R. Egan and F. Gibou, xGFM: Recovering convergence of fluxes in the ghost fluid method, J. Comput. Phys. 409 (2020) 109351.

[17] R.P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), J. Comput. Phys. 152 (1999) 457-492.

[18] W. Feng, X. He, Y. Lin, X. Zhang, Immersed finite element method for interface problems with algebraic multigrid solver. Commun. Comput. Phys. 15 (2014), 1045-1067.

[19] H. Feng, G. Long, and S. Zhao, An augmented matched interface and boundary (MIB) method for solving elliptic interface problem, J. Comput. Appl. Math. 361 (2019) 426-433.

[20] H. Feng and S. Zhao, FFT-based high order central difference schemes for three-dimensional Poisson's equation with various types of boundary conditions, J. Comput. Phys. 410 (2020) 109391.

[21] H. Feng and S. Zhao, A fourth order finite difference method for solving elliptic interface problems with the FFT acceleration, J. Comput. Phys. 419 (2020) 109677.

[22] H. Feng and S. Zhao, A multigrid based finite difference method for solving parabolic interface problem, Electronic Research Archive, 29, 3141-3170, (2021)

[23] Q. W. Feng, B. Han and P. Minev, Sixth order compact finite difference scheme for Poisson interface problem with singular sources. Comput. Math. Appl. 99 (2021) 2-25.

[24] Q. W. Feng, B. Han and P. Minev, A high order compact finite difference scheme for elliptic interface problems with discontinuous and high-contrast coefficients. Appl. Math. Comput. 431 (2022) 127314.

[25] J. Fernández-Fidalgo, S. Clain, L. Ramírez, I. Colominas, and X. Nogueira, Very high-order method on immersed curved domains for finite difference schemes with regular Cartesian grids, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112782.

[26] B. Fornberg, Classroom note: Calculation of weights in finite difference formulas. SIAM Review 40 (1998) 685-691.

[27] F. Gibou and R.P. Fedkiw, A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem, J. Comput. Phys. 202 (2005) 577-601.

[28] J. L. Hellrung Jr., L. M. Wang, E. Sifakis, and J. M. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. 231 (2012) 2015-2048.

[29] P.Hemker, On the order of prolongations and restrictions in multigrid procedures,J. Comput. Appl. Math. 32(3)(1990)423-429.

[30] S. Hou,and X.D. Liu, A numerical method for solving variable coefficient elliptic equation with interfaces, J. Comput. Phys. 202(2) (2005)411-445.

[31] X. He, T. Lin, and Y. Lin, Immersed finite element methods for elliptic interface problems with non-homogeneous jump conditions, Int. J. Numer. Anal. Model. 8 (2011) 284-301.

[32] G. Jo and D. Y. Kwak, Geometric multigrid algorithms for elliptic interface problems using structured grids, Numer. Algorithms 81 (2019) 211-235.

[33] G. Jo and D.Y. Kwak, A semi-uniform multigrid algorithm for solving elliptic interface problems, Comput. Methods Appl. Math. 21 (2021) 127-143.

[34] R.J. LeVeque and Z.L. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019-1044.

[35] J. Li, J. M. Melenk, B. Wohlmuthc, and J. Zou, Optimal a priori estimates for higher order finite elements for elliptic interface problems, Appl. Numer. Math. 60 (2010) 19-37.

[36] Z.L. Li, A fast iterative algorithm for elliptic interface problem, SIAM J. Numer. Anal. 35 (1998) 230-254.

[37] Z.L. Li, H. Ji, and X. Chen, Accurate solution and gradient computation for elliptic interface problems with variable coefficients, SIAM J. Numer. Anal. 55 (2017) 670-697.
[38] M.N. Linnick and H.F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, J. Comput. Phys. 204 (2005) 157-192.
[39] C. Liu and C. Hu, A second order ghost fluid method for an interface problem of the Poisson equation, Commun. Comput. Phys. 22 (2017) 965-996.
[40] X. Liu, R. Fedkiw, and M. kang, A boundary condition capturing method for Poisson's equation on irregular domains, J. Comput. Phys. 160 (2000) 151-178.
[41] P. Martinsson, A direct solver for variable coefficient elliptic pdes discretized via a composite spectral collocation method, J. Comput. Phys. 242 (2013) 460-479.
[42] M. Oevermann, R. Klein, A Cartesian grid finite volume method for elliptic equations with variable coefficients and embedded interfaces, J. Comput. Phys. 219 (2006) 749-769.
[43] Z. Li and K. Ito, The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains, SIAM, (2006).
[44] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220-252.
[45] Y. Ren and S. Zhao, A FFT accelerated fourth order finite difference method for solving three-dimensional elliptic interface problems, J. Comput. Phys. 477, 111924, (2023)
[46] Y. Ren, H. Feng, and S. Zhao, A FFT accelerated high order finite difference method for elliptic boundary value problems over irregular domains, J. Comput. Phys. 448 (2022) 110762.
[47] J. W. Ruge and K. Stuben, Algebraic multigrid, in Multigrid Methods, S. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73-130.
[48] Y. Shu, I. Chern, and C. Chang, Accurate gradient approximation for complex interface problems in 3D by an improved coupling interface method, J. Comput. Phys. 275 (2014) 642-661.
[49] V. Stiernstrom, M. Almquist, and K. Mattsson, Boundary-optimized summation-by-parts operators for finite difference approximations of second derivatives with variable coefficients, J. Comput. Phys. 491(2023)112376
[50] A. Wiegmann and K. P. Bube, The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth olutions, SIAM J. Numer. Anal. 37(2004) 827-862.
[51] S. Yu and G. Wei, Three-dimensional matched interface and boundary (MIB) method for treating geometric singularities, J. Comput. Phys. 227 (2007) 602-632.
[52] S. Zhao, High order matched interface and boundary method for the Helmholtz equation in media with arbitrarily curved interfaces, J. Comput. Phys. 229 (2010) 3155-3170.
[53] C. Zhang, Z. Li, and X. Yue, An acceleration technique for the augmented IIM for 3D elliptic interface problems, Numer. Math. Theory Meth. Appl. 14 (2021) 773-796.
[54] X. Zhong, A new high-order immersed interface method for solving elliptic equations with imbedded interface of discontinuity, J. Comput. Phys. 225 (2007) 1066-1099.
[55] H. Zhou and W. Ying, A correction function-based kernel-free boundary integral method for elliptic PDEs with implicitly defined interfaces, J. Comput. Phys. 496 (2024) 112545.
[56] Y.C. Zhou, S. Zhao, M. Feig, and G.W. Wei, High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular source, J. Comput. Phys. 213 (2006) 1-30.

Appendix

Theorem 2. *Corrected fourth order FD for Type 2 case in Fig. 3.* Let $x_{i-2} \leq \alpha_1 \leq x_{i-1} \leq x_i \leq x_{i+1} \leq \alpha_2 \leq x_{i+2}, h_1^- = x_{i-2} - \alpha_1$, $h_1^+ = x_{i-1} - \alpha_1, h_2^- = x_{i+1} - \alpha_2$, and $h_2^+ = x_{i+2} - \alpha_2$. Suppose $u \in C^6[x_{i_2} - 2h, \alpha_1) \bigcap C^6(\alpha_1, \alpha_2) \bigcap C^6(\alpha_2, x_{i+2} + 2h]$, with derivative extending continuously up to the interface. Then the following approximations hold to $O(h^4)$ when $K = 5$:

$$(\beta u_x)_x(x_{i-3}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i-3}) + (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i-3}) \sum_{k=0}^{K} \frac{(h_1^+)^k}{k!}[u^{(k)}]_{\alpha_1},$$

$$(\beta u_x)_x(x_{i-2}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i-2}) - (\frac{4}{3h^2}\beta + \frac{2}{3h}\beta_x)(x_{i-2})\sum_{k=0}^{K}\frac{(h_1^+)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i-2})\sum_{k=0}^{K}\frac{(h+h_1^+)^k}{k!}[u^{(k)}]_{\alpha_1},$$

$$(\beta u_x)_x(x_{i-1}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i-1}) + (\frac{4}{3h^2}\beta - \frac{2}{3h}\beta_x)(x_{i-1})\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$- (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i-1})\sum_{k=0}^{K}\frac{(h_1^- - h)^k}{k!}[u^{(k)}]_{\alpha_1},$$

$$(\beta u_x)_x(x_{i}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i}) - (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i})\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i})\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+1}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i+1}) - (\frac{4}{3h^2}\beta + \frac{2}{3h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}[u^{(k)}]_{\alpha_2}$$
$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h+h_2^+)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+2}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i+2}) + (\frac{4}{3h^2}\beta - \frac{2}{3h}\beta_x)(x_{i+2})\sum_{k=0}^{K}\frac{(h_2^-)^k}{k!}[u^{(k)}]_{\alpha_2}$$
$$- (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i+2})\sum_{k=0}^{K}\frac{(h_2^- - h)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+3}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i+3}) - (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i+3})\sum_{k=0}^{K}\frac{(h_2^-)^k}{k!}[u^{(k)}]_{\alpha_2},$$

***Theorem 3**. Corrected fourth order FD for Type 3 case in Fig. 3.* Let $x_{i-1} \leq \alpha_1 \leq x_i \leq x_{i+1} \leq \alpha_2 \leq x_{i+2}, h_1^- = x_{i-1} - \alpha_1$, $h_1^+ = x_i - \alpha_1, h_2^- = x_{i+1} - \alpha_2$, and $h_2^+ = x_{i+2} - \alpha_2$. Suppose $u \in C^6[x_{i_1} - 2h, \alpha_1)\bigcap C^6(\alpha_1, \alpha_2)\bigcap C^6(\alpha_2, x_{i+2} + 2h]$, with derivative extending continuously up to the interface. Then the following approximations hold to $O(h^4)$ when $K = 5$:

$$(\beta u_x)_x(x_{i-2}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i-2}) + (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i-2})\sum_{k=0}^{K}\frac{(h_1^+)^k}{k!}[u^{(k)}]_{\alpha_1},$$

$$(\beta u_x)_x(x_{i-1}) \approx (\beta\delta_{xx}+\beta_x\delta_x)(x_{i-1}) - (\frac{4}{3h^2}\beta + \frac{2}{3h}\beta_x)(x_{i-1})\sum_{k=0}^{K}\frac{(h_1^+)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i-1})\sum_{k=0}^{K}\frac{(h_1^+ + h)^k}{k!}[u^{(k)}]_{\alpha_1},$$

$$(\beta u_x)_x(x_i) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_i) + (-\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h_1^- - h)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$+ (\frac{4}{3h^2}\beta - \frac{2}{3h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h_2^+ + h)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+1}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i+1}) + (-\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$- (\frac{4}{3h^2}\beta + \frac{2}{3h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}[u^{(k)}]_{\alpha_2}$$
$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h_2^+ + h)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+2}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i+2})$$
$$+ (-\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i+2})\sum_{k=0}^{K}\frac{(h_2^- - h)^k}{k!}[u^{(k)}]_{\alpha_2}$$
$$+ (\frac{4}{3h^2}\beta - \frac{2}{3h}\beta_x)(x_{i+2})\sum_{k=0}^{K}\frac{(h_2^-)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+3}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i+3}) - (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i+3})\sum_{k=0}^{K}\frac{(h_2^-)^k}{k!}[u^{(k)}]_{\alpha_2},$$

Theorem 4*. Corrected fourth order FD for Type 4 case in Fig. 3.* Let $x_{i-1} \le \alpha_1 \le x_i \le \alpha_2 \le x_{i+1}, h_1^- = x_{i-1} - \alpha_1, h_1^+ = x_i - \alpha_1, h_2^- = x_i - \alpha_2, h_2^+ = x_{i+1} - \alpha_2$. Suppose $u \in C^6[x_{i_1} - 2h, \alpha_1) \bigcap C^6(\alpha_1, \alpha_2) \bigcap C^6(\alpha_2, x_{i+1} + 2h]$, with derivative extending continuously up to the interface. Then the following approximations hold to $O(h^4)$ when $K = 5$:

$$(\beta u_x)_x(x_{i-2}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i-2}) + (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_{i-2})\sum_{k=0}^{K}\frac{(h_1^+)^k}{k!}[u^{(k)}]_{\alpha_1},$$

$$(\beta u_x)_x(x_{i-1}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i-1}) - (\frac{4}{3h^2}\beta + \frac{2}{3h}\beta_x)(x_{i-1})\sum_{k=0}^{K}\frac{(h_1^+)^k}{k!}[u^{(k)}]_{\alpha_1},$$

$$(\beta u_x)_x(x_i) \approx (\frac{\beta}{h^2}\delta_{xx} + \frac{\beta_x}{h}\delta_x)(x_i) + (-\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h_1^- - h)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$+ (\frac{4}{3h^2}\beta_x - \frac{2}{3h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h_1^-)^k}{k!}[u^{(k)}]_{\alpha_1}$$
$$- (\frac{4}{3h^2}\beta + \frac{2}{3h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h_2^+)^k}{k!}[u^{(k)}]_{\alpha_2}$$

$$+ (\frac{1}{12h^2}\beta + \frac{1}{12h}\beta_x)(x_i)\sum_{k=0}^{K}\frac{(h_2^+ + h)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+1}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i+1}) + (\frac{4}{3h^2}\beta - \frac{2}{3h}\beta_x)(x_{i+1})\sum_{k=0}^{K}\frac{(h_2^-)^k}{k!}[u^{(k)}]_{\alpha_2},$$

$$(\beta u_x)_x(x_{i+2}) \approx (\beta\delta_{xx} + \beta_x\delta_x)(x_{i+2}) - (\frac{1}{12h^2}\beta - \frac{1}{12h}\beta_x)(x_{i+2})\sum_{k=0}^{K}\frac{(h_2^-)^k}{k!}[u^{(k)}]_{\alpha_2},$$

Department of Mathematics, University of Alabama, Tuscaloosa AL, 35487, USA
E-mail: `szhao@ua.edu`
URL: `https://sites.ua.edu/szhao/`