Journal of Computational Physics 419 (2020) 109677

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A fourth order finite difference method for solving elliptic interface problems with the FFT acceleration

Hongsong Feng, Shan Zhao *

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

ARTICLE INFO

Article history:
Received 27 April 2020
Received in revised form 16 June 2020
Accepted 16 June 2020
Available online 22 June 2020

Keywords:
Elliptic interface problem
Mixed boundary conditions
Fast Fourier transform (FFT)
High order central difference schemes
Matched interface and boundary method (MIB)
Gradient recovery

ABSTRACT

In this paper, a new Cartesian grid finite difference method is introduced based on the fourth order accurate matched interface and boundary (MIB) method and fast Fourier transform (FFT). The proposed augmented MIB method consists of two major parts for solving elliptic interface problems in two dimensions. For the interior part, in treating a smoothly curved interface, zeroth and first order jump conditions are enforced repeatedly by the MIB scheme to generate fictitious values near the interface. For the exterior part, two layers of zero-padding solutions are introduced beyond the original rectangular domain so that the FFT inversion becomes feasible. Different types of boundary conditions, including Dirichlet, Neumann, Robin and their mix combinations, can be imposed via the MIB scheme to generate fictitious values near boundaries. Based on fictitious values at both interfaces and boundaries, the augmented MIB method reconstructs Cartesian derivative jumps as auxiliary variables. Then, by treating such variables as unknowns, an enlarged linear system is obtained. In the Schur complement solution of such system, the FFT algorithm will not sense the solution discontinuities, so that the discrete Laplacian can be efficiently inverted. Therefore, the FFT-based augmented MIB not only achieves a fourth order of accuracy in dealing with interfaces and boundaries, but also produces an overall complexity of $O(n^2 \log n)$ for a $n \times n$ uniform grid. Moreover, the augmented MIB scheme can provide fourth order accurate approximations to solution gradients and fluxes.

1. Introduction

This paper concerns with a two-dimensional (2D) elliptic interface problem of the Poisson's equation

$$-\nabla \cdot (\beta \nabla u) = f(x, y), \quad (x, y) \in \Omega \tag{1}$$

with a Dirichlet boundary condition

$$u(x, y) = g(x, y), \quad (x, y) \in \partial\Omega. \tag{2}$$

Other boundary conditions such as Robin condition will also be considered. Here the 2D domain Ω is assumed to be a rectangle one, and is partitioned into $\Omega = \Omega^+ \cup \Omega^-$ by a smooth interface $\Gamma = \Omega^+ \cap \Omega^-$, across which there exist discontinuous coefficients and singular source. In this paper, we concern ourselves with the case of piecewise constant coefficients, i.e., β is defined to be β^+ in Ω^+ and β^- in Ω^-. Meanwhile, the discontinuous source term $f(x, y)$ is denoted as $f^+(x, y)$ and

* Corresponding author.
E-mail address: szhao@ua.edu (S. Zhao).

https://doi.org/10.1016/j.jcp.2020.109677
0021-9991/

$f^-(x, y)$ in Ω^+ and Ω^-, respectively. Across the interface Γ, the function and flux jumps are assumed to be given by the jump conditions

$$[\![u]\!] := u^+ - u^- = \phi(x, y), \tag{3}$$

$$[\![\beta u_v]\!] := \beta^+ \nabla u^+ \cdot \vec{\nu} - \beta^- \nabla u^- \cdot \vec{\nu} = \psi(x, y), \tag{4}$$

where $\vec{\nu}$ is the unit outer normal vector pointing from Ω^- to Ω^+, and the superscript stands for the limiting value from each side of the interface. Equations (3) and (4) are called as the zeroth and first order jump conditions.

Elliptic interface problem has been extensively discussed and researched due to its various applications including computational electromagnetics and optics [24,55], biomolecular electrostatics [42,21], and material science [28]. Due to the loss of regularity of solution across the interface, traditional numerical methods fail to provide accurate solutions for the interface problems. To recover the numerical accuracy near the interface, a variety of methods have been developed via enforcing the jump conditions (3) and (4) into numerical discretization such that accurate and robust numerical algorithms can be designed.

Finite element method (FEM) is a commonly used approach for elliptic interface problem, and was studied in early time by Babuška [2] in 1970s. With the help of body-fitted unstructured meshes, continuous FEMs [5,7,10,13,26] and discontinuous FEMs including discontinuous Galerkin (DG) [14,31] and weak Galerkin (WG) [44,45], have been developed for handling elliptic interfaces and irregular geometries. For interfaces with complex topologies, the construction of high quality body-fitted meshes could be difficult and time-consuming. This motivates the development of FEMs based on non-body-fitted structured meshes, such as extended FEM (XFEM) [20,32] and immersed FEMs [16,25,29,30,51]. To tackle elements where the interface is cut-through, XFEM introduces additional discontinuous basis functions, while immersed FEMs modify the local basis function to satisfy jump conditions. Semi-structured meshes have also been constructed by using Delaunay triangulation [8] and Voronoi diagram [22], where Cartesian meshes are locally modified near the interface. Based on such meshes, virtual element method [8] and finite volume method [22] provide efficient solution of elliptic interface problems.

Finite difference methods associated with Cartesian grids have also been intensively investigated for elliptic interface problems. Peskin pioneered the field by introducing a first order accurate immersed boundary method in 1970s [46]. The immersed interface method (IIM) introduced by LeVeque and Li in 1994 [33] is known to be the first second order Cartesian grid method for elliptic interface problem. In the IIM, jump conditions are rigorously imposed by using generalized Taylor expansions to modify the classic central difference stencil near the interface. Some further researches on the IIM were carried out to improve the stability [36] or efficiency [35]. Ghost fluid method (GFM) originally developed for the inviscid Euler equation [17] has been reformulated in [41] for solving elliptic interface problems. Albeit being first order, the GFM is quite efficient, thanks to its symmetric matrix structure maintained in the finite difference discretization. An extended GFM for recovering convergence of fluxes has recently been constructed in [15]. The improvement of the GFM accuracy has been investigated in [40,39]. Zhou et al. [58] proposed the matched interface and boundary (MIB) method by iteratively enforcing the zeroth and first order jump conditions, yielding several layers of fictitious values neighboring the interface in order to apply high order central difference stencil. The MIB method is able to not only achieve high order accuracy for smoothly curved interfaces [58,55], but also maintain second order accuracy for sharp-edged interfaces [54]. In recent years, a lot of other effective interface algorithms have been designed, such as Coupling interface method [11], piecewise-polynomial interface method [9], kernel free integral equation method [53], virtual node method [3,27], and finite volume method [4].

Based on the success of the aforementioned approaches for solving elliptic partial differential equation (PDE), the current research hot topics in this field focus on more challenging interface problems, such as low regularity solutions [30,45], geometric singularities [54], higher order convergence [58,34], and fast Poisson solvers [35,18]. The present paper concerns with two of such difficulties, i.e., combining higher order convergence with fast Poisson solvers.

Higher order methods are cost-efficient and desirable for problems associated with high frequency waves. Nevertheless, most interface schemes in the literature are designed to be of second order accuracy. Only a few methods can achieve a order higher than two for elliptic PDEs with smooth interfaces. For FEMs, it is known [34] that the higher order of convergence crucially depends on how well the interface is resolved by the triangular mesh. In practice, subparametric, isoparametric or superparametric elements are usually employed to secure the optimal order of $p + 1$ in the L_2 norm for a polynomial order of p, for both continuous [34] and DG [31] FEMs. By properly handling jumps of solution and flux in a weak variational form, the WG FEM delivers fourth order convergence for L_2 projected solutions [44,45]. For Cartesian grid finite difference methods, a fourth order IIM could be constructed by computing high order jump conditions involving mixed derivatives [37]. By iteratively using zero and first order jump conditions, the MIB method [58,55] normally achieves a fourth order of convergence in resolving curved interfaces based on simple Cartesian grids and could reach up to sixth order when the interfaces are smooth enough. In [57], another fourth order IIM was developed in two-dimensions (2D) by iteratively using zero and first order jump conditions too.

Besides the accuracy improvement, the acceleration of algebraic solution of elliptic interface problems also draws a lot of attentions. Having symmetric matrix structures, some interface algorithms [22,41,50] enjoy a better efficiency in iterative solution of algebraic systems. Moreover, it is well known that the computation of elliptic PDEs can be significantly accelerated by using fast Poisson solvers, including geometric multigrid method with a complexity $O(N)$ and fast Fourier transform (FFT) with a complexity $O(N \log N)$ for a spatial degree of freedom N. Thus, there exists a strong interest to adopt Poisson

solvers in interface algorithms so that the algebraic computation is not limited by iterative solvers with a general complexity $O(N^2)$. However, due to sophisticated interface treatments, the incorporation of fast solvers into interface algorithms is highly nontrivial. For example, in multigrid methods, great challenge exists in the combination of interface treatment with the formulation of the restriction and prolongation in a multigrid cycle. Even though several multigrid interface algorithms have been successfully developed for elliptic interface problems, including IIM [1,38], piecewise-polynomial interface method [9], ghost point method [12], and virtual node method [3,27], the use of multigrid solvers in other interface algorithms is still not straightforward.

For FFT Poisson solver, an effective approach has been developed in [35,50,18,19] by introducing auxiliary variables to form an augmented system so that the Laplacian operator can be approximated by the standard finite difference stencil, which results in a symmetric and diagonally dominant matrix for FFT inversion. By using a Schur complement procedure, the auxiliary variables are solved iteratively. Since the total number of auxiliary variables is one dimensionally smaller than N, the algebraic complexity of the augmented approach is usually $O(N \log N)$. Such an idea was first introduced in the augmented IIM (AIIM) [35,38] which defines the jump in the normal derivative of the solution $[\![u_v]\!]$ as an auxiliary variable. The AIIM has been successfully applied for solving interface problems with piecewise constant [35] and variable coefficients [38], respectively, by using FFT and multigrid Poisson solvers. Later, a different type of auxiliary variables has been considered in explicit-jump IIM (EJIIM) [50] and augmented MIB (AMIB) method [18], i.e., Cartesian derivative jumps $[\![u_x]\!]$, $[\![u_{xx}]\!]$, $[\![u_y]\!]$ and $[\![u_{yy}]\!]$. We note that a fundamental difference of the AMIB [18] in comparing with the AIIM [35,38] and EJIIM [50] is that the auxiliary variables or jump quantities are approximated in the AMIB by only using two low order jump conditions (3) and (4), while in IIMs, more jump conditions in local coordinates are needed and have to be derived by using the PDE.

However, all fast interface algorithms mentioned above are just second order accurate. To the best of the authors' knowledge, no FFT or multigrid based fourth order methods have ever been developed for solving elliptic interface problems. In a related field, such methods have been successfully constructed for solving Poisson or Helmholtz equation on irregular domains [6,43,48,52]. But without material interfaces, the feasibility for extending these methods to interface problems is unclear.

For elliptic interface problems, the implementation of fast solvers in existing fourth order interface algorithms faces various difficulties. So does the generalization of existing fast interface algorithms to fourth order. For example, in verifying higher order convergence of FEMs, elements near the interface have to be adjusted in each mesh refinement, in order to ensure that they are interface-aligned [34]. This brings extra difficulties for applying geometrical multigrid methods to fourth order FEMs. For fast IIMs including AIIM [35,38] and EJIIM [50], the extension from second order to fourth order requires up to fourth order jump conditions involving mixed derivatives [37], which impose a big challenge in numerical implementation. We note that such a difficulty is simply bypassed in the fourth order MIB [58] by imposing zeroth and first order jump conditions repeatedly. Therefore, the combination of fourth order MIB [58] with the augmented MIB [18] seems to be promising.

The objective of this work is to develop a novel augmented MIB for solving elliptic interface problems, which not only achieves a fourth order of accuracy in dealing with interfaces and boundaries, but also maintains the FFT complexity of $O(N \log N)$. Based on our previous second order AMIB [18], the present study addresses two major difficulties. In resolving the first difficulty, the fourth order MIB [58] is applied in the framework of augmented approach to treat a smoothly curved interface. However, the combination of the fourth order MIB [58] with the AMIB [18] faces another grand difficulty: FFT-based fourth order central difference scheme has never been reported for Poisson boundary value problems over cubic domains before our studies. Fortunately, another AMIB method [19] has been developed recently to handle the Dirichlet, Neumann, Robin or any combination of boundary conditions, and can achieve arbitrarily high order in principle. By introducing auxiliary variables both at interior interfaces and exterior boundaries, the present AMIB provides a unified framework for solving elliptic interface problems with various boundary conditions. Our algorithm can achieve the fourth order accuracy with an overall computational complexity of $O(N \log N)$.

Recently, gradient recovery for elliptic interface problems has received a considerable attention. Improvements in both immersed FEM [23] and finite difference methods [49,15] have been introduced to provide an accurate approximation to solution gradients or fluxes. In the proposed AMIB method, gradients and fluxes can be simply computed by using the calculated numerical solutions and fictitious values, and such approximations are found to be fourth order accurate too.

The rest of this paper is organized in the following way. Section 2 is devoted to our new treatments for interior interface and exterior boundary. A unified augmented system is formed and its fast computation is discussed. Section 3 focuses on the numerical results to demonstrate the performance of the proposed algorithm. Lastly, a summary is given in section 4.

2. Theory and algorithm

For elliptic interface problems with piecewise constant coefficients, the original PDE (1) can be rewritten as below after moving the coefficient β to the right hand side,

$$\triangle u = -\frac{f(x, y)}{\beta}, \quad (x, y) \in (\Omega^- \cup \Omega^+) \setminus \Gamma, \tag{5}$$

with the same Dirichlet boundary conditions (2) and interface jump conditions (3) and (4) imposed.

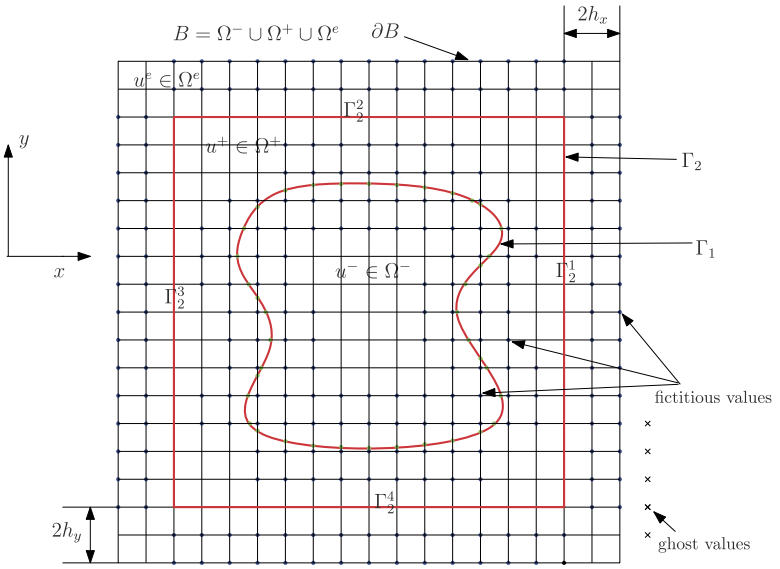


Fig. 1. The original problem is recasted into an immersed boundary problem. The red curves and lines indicate the original and introduced interfaces, respectively. Blue dots stand for the needed fictitious points for correcting the fourth order central difference. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

We are concerned with solving (5) on an rectangular domain $\Omega = [a, b] \times [c, d]$, which is separated into $\Omega = \Omega^+ \cup \Omega^-$ by a closed interface Γ. A uniform grid is employed. In particular, we partition the domain Ω into m and n equally spaced intervals in x- and y-directions respectively with mesh sizes $h_x = (b-a)/m$ and $h_y = (d-c)/n$. The grid coordinates are therefore defined as

$$x_i = a + ih_x, \;\; y_j = c + jh_y, \;\; i = 0, \cdots, m, \;\; j = 0, \cdots, n. \tag{6}$$

This work aims to develop a fourth order AMIB method for Poisson interface problem (5), with special attentions to not only the *fourth order accuracy*, but also the *computational efficiency*. The development consists of two major parts. For the interior part, the classical fourth order MIB scheme [58] will be formulated in the augmented approach [18] to treat smoothly curved interfaces. For the exterior part, the fourth order AMIB scheme introduced in [19] will be adopted to handle various different boundary conditions. Then, a unified augmentation framework will be applied to combine both interface and boundary treatments into one discretization. Following [19], the first step of our AMIB algorithm is to introduce an immersed boundary.

2.1. Immersed boundary

As shown in [19], the FFT inversion of higher order central difference discretization of the Laplacian operator requires that the solution satisfies an anti-symmetric property at boundaries. Since this property is generally invalid, a simple procedure is introduced in [19] by adding a zero-padding zone outside the boundary $\partial\Omega$. In this way, anti-symmetric property across the new boundary of the extended domain is trivially satisfied, which provides a foundation for FFT inversion. Consequently, the original boundary $\partial\Omega$ becomes an immersed boundary.

For achieving fourth order of accuracy, we need to equip the extended region with a width equaling to $2h_x$ or $2h_y$ for x- or y-direction outside the domain Ω. Then, the grid coordinates over the extended domain are redefined as

$$R = \{(x_i, y_j) | x_i = a + (i-2)h_x, \;\; y_j = c + (j-2)h_y, \;\; i = 0, \cdots, m+4, \;\; j = 0, \cdots, n+4\}, \tag{7}$$

where h_x, h_y are defined as above. One may refer to Fig. 1 for the setting of grid nodes (7).

From now on, we denote the extended subdomain as Ω^e, yielding a new computational domain composed of three subdomains: $B = \Omega^+ \cup \Omega^- \cup \Omega^e$. The solution in the extended domain Ω^e is simply defined as $u^e = 0$, with the related source term as $f^e = 0$. Now the original interface problem becomes an immersed boundary problem in a larger domain, with the original boundary condition converted to additional interface condition defined on the introduced interface $\Gamma_2 = \Omega^+ \cap \Omega^e$:

$$[\![u]\!] := u^+ - u^e = g(x, y). \tag{8}$$

On the other hand, we may redefine the original interface $\Gamma = \Omega^- \cap \Omega^+$ as $\Gamma_1 = \Gamma$. Moreover, we will abuse the notation by denoting the two sets of interfaces as $\Gamma = \Gamma_1 \cup \Gamma_2$. Then the immersed boundary problem can be modeled as

$$\triangle u = -\frac{f(x,y)}{\beta}, \quad (x,y) \in B \setminus \Gamma, \tag{9}$$

with source term being

$$f = \begin{cases} f^-, \vec{x} \in \Omega^-, \\ f^+, \vec{x} \in \Omega^+, \\ f^e, \vec{x} \in \Omega^e, \end{cases}$$

and coefficient defined as

$$\beta = \begin{cases} \beta^-, \vec{x} \in \Omega^-, \\ \beta^+, \vec{x} \in \Omega^+, \\ 1, \vec{x} \in \Omega^e. \end{cases}$$

The new problem is now subject to the original interface conditions (3)-(4) and the immersed interface condition (8). Besides, a Dirichlet boundary condition is trivially assumed for the zero-padding solution

$$u(x,y) = 0, \qquad (x,y) \in \partial B.$$

Outside ∂B, one layer of ghost points with zero values is assumed. Such ghost values have their anti-symmetric counterparts defined inside ∂B. Therefore, anti-symmetric property is trivially satisfied across the new boundary ∂B.

In the proposed AMIB method, the fourth order central difference scheme will be applied in a translation-invariant manner in both x and y directions for every node inside ∂B. However, due to the loss of solution regularity across the interfaces Γ_1 and Γ_2, central differences may fail to provide desired accuracy for some grid points near the interfaces. Special MIB treatments are needed to compensate the solution discontinuity at such irregular points.

We first define irregular points near Γ_1. For this purpose, we assume that the interface Γ_1 is governed by a level set function $\Gamma_1 = \{(x,y), \varphi(x,y) = 0\}$, with $\varphi(x,y) < 0$ in Ω^- and $\varphi(x,y) > 0$ in $\Omega^+ \cup \Omega^e$. Then two grid functions are calculated at each grid point

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

If $\varphi_{i,j}^{min}\varphi_{i,j}^{max} > 0$, then the grid point (x_i, y_j) is called a regular point, otherwise irregular point. It can be seen from Fig. 1 that there are four layers of fictitious points surrounding Γ_1, two inside and two outside.

Irregular points near interface Γ_2 can be similarly defined. But a level set function is not needed here, because each side of Γ_2 is a straight line and aligns with a grid line. The irregular points include two layers of grid points outside Γ_2, the grid points on Γ_2, and the nearest one layer of grid points inside Γ_2. In Fig. 1, only two layers of fictitious points outside Γ_2 are highlighted, on which nontrivial fictitious values are required. The other two layers are not highlighted, because the corresponding fictitious values are simply zeros for zero-padding solutions.

To avoid interference between MIB treatments for Γ_1 and Γ_2, in our computations, we require that the distance between Γ_1 and Γ_2 should be at least $4h_x$ or $4h_y$ in each direction. Such a requirement is not satisfied in the illustration given in Fig. 1. A rather tight grid setting is shown in Fig. 1 to save the space.

2.2. Fictitious values formulation

At all irregular points defined above, the fourth order central difference approximations need to be modified to account for interface and boundary conditions. The main idea of the MIB interface treatment [58] is that to approximate function or its derivatives on one side of interface, one never directly refers to function values from the other side. Instead, fictitious values from the other side of the interface will be supplied. Therefore, for every irregular point, a fictitious value needs to be assumed, which could be regarded as a smooth extension of the solution from the other side. In the MIB method, such fictitious value is generated by rigorously imposing jump conditions [58] or boundary conditions [56]. In the present study, there are two types of interfaces Γ_1 and Γ_2. Their MIB treatments require different strategies.

2.2.1. Fictitious values near interface Γ_1

In the first place, we focus ourselves on the case of interface Γ_1. We are taking advantage of the jump conditions (3) and (4) together with one more jump condition, which is analytically derived by differentiating Eq. (3) along the tangential direction τ of the interface as below,

$$[\![u_\tau]\!] = u_\tau^+ - u_\tau^- = \rho(x,y). \tag{10}$$

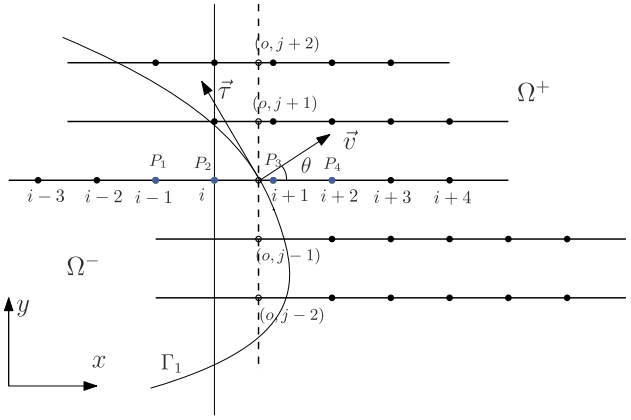


Fig. 2. Consider the case where the interface intersects $y = y_j$ at (x_o, y_j). At four irregular points $P_1(i-1, j)$, $P_2(i, j)$, $P_3(i+1, j)$, and $P_4(i+2, j)$, fictitious values (in blue color) can be constructed. Here θ is the angle between positive x-direction and the normal vector $\vec{\nu}$.

Consider a point (x^*, y^*) on the interface. Define θ as the angle between positive x-direction and normal direction. Then the tangential and normal direction could be expressed as $\vec{\tau} = (-\sin\theta, \cos\theta)$ and $\vec{\nu} = (\cos\theta, \sin\theta)$, respectively. See Fig. 2 in case that (x^*, y^*) is located on a y grid line. At (x^*, y^*), the three known interface conditions can be reinterpreted in the Cartesian form [58],

$$[\![u]\!] = u^+ - u^- = \phi(x^*, y^*), \tag{11}$$

$$[\![u_\tau]\!] = (-u_x^+ \sin\theta + u_y^+ \cos\theta) - (-u_x^- \sin\theta + u_y^- \cos\theta) = \rho(x^*, y^*), \tag{12}$$

$$[\![\beta u_\nu]\!] = \beta^+(u_x^+ \cos\theta + u_y^+ \sin\theta) - \beta^-(u_x^- \cos\theta + u_y^- \sin\theta) = \psi(x^*, y^*), \tag{13}$$

where function values of ϕ, ρ, and ψ can be computed at a given point (x^*, y^*). In this way, jump conditions from tangential or normal direction are transformed into x- or y-direction involving four partial derivatives u_x^+, u_x^-, u_y^+, and u_y^-. By eliminating one of the four partial derivatives, the following formulations (14) or (15) can be obtained with three of the four quantities remaining. If u_y^- is eliminated, the following equations can be derived

$$[\![u]\!] = u^+ - u^-, \quad [\![\beta u_\nu]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ u_x^+ - C_x^- u_x^- + C_y^+ u_y^+, \tag{14}$$

where $C_x^+ = \beta^+ \cos\theta + \beta^- \tan\theta \sin\theta$, $C_x^- = \beta^- \cos\theta + \beta^- \tan\theta \sin\theta$ and $C_y^+ = \beta^+ \sin\theta - \beta^- \sin\theta$. If u_x^- is removed, we can obtain that

$$[\![u]\!] = u^+ - u^-, \quad [\![\beta u_\nu]\!] + \beta^- \cot\theta [\![u_\tau]\!] = D_y^+ u_y^+ - D_y^- u_y^- + D_x^+ u_x^+, \tag{15}$$

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

We note that in the MIB scheme [58,55], high order jump conditions will not be required for achieving high order accuracy. Instead, we repeatedly impose zeroth and first order jump conditions given in Eqns. (14) and (15) to determine all fictitious values. Moreover, Eqns. (14) and (15) enable us to treat 2D jump conditions only along Cartesian directions. For example, if u_y^+ is known or can be numerically determined in (14), then Eq. (14) involves only one-dimensional (1D) jump conditions in x direction. Similarly, after approximating u_x^+ in the perpendicular direction, Eq. (15) becomes 1D jump conditions in y direction.

Let us focus only on the x direction to illustrate the MIB fictitious value generation. We aim at creating four layers of fifth order accurate fictitious values along x direction. To illustrate the idea, we refer to the case shown in Fig. 2 with the interface intersecting with $y = y_j$ at (x_o, y_j). Totally four fictitious values at the grid points P_1, P_2, P_3 and P_4 are needed in fourth order central difference for approximating u_{xx} at some grid point. They are generated in two steps in the MIB scheme.

We first generate a pair of fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ at points P_2 and P_3 on each side of the interface simultaneously. Jump conditions in (14) are discretized as below

$$[\![u]\!] = W_0^+ U^+ - W_0^- U^- \tag{16}$$

$$[\![\beta u_\nu]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ W_1^+ U^+ - C_x^- W_1^- U^- + C_y^+ P^+ U_\Gamma, \tag{17}$$

with the following vector representations

$$\begin{cases} U^+ = (\hat{u}_{i,j}, u_{i+1,j}, u_{i+2,j}, u_{i+3,j}, u_{i+4,j})^T \\ U^- = (u_{i-3,j}, u_{i-2,j}, u_{i-1,j}, u_{i,j}, \hat{u}_{i+1,j})^T \end{cases},$$

$$\begin{cases} W_0^+ = (w_{0,i}^+, w_{0,i+1}^+, w_{0,i+2}^+, w_{0,i+3}^+, w_{0,i+4}^+) \\ W_0^- = (w_{0,i-3}^-, w_{0,i-2}^-, w_{0,i-1}^-, w_{0,i}^-, w_{0,i+1}^-) \end{cases},$$

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

$$\begin{cases} U_\Gamma = (u_{o,j-2}, u_{o,j-1}, u_{o,j}, u_{o,j+1}, u_{o,j+2})^T \\ P^+ = (p_{1,j-2}^+, p_{1,j-1}^+, p_{1,j}^+, p_{1,j+1}^+, p_{1,j+2}^+) \end{cases}.$$

The discretization in (16) and (17) is formulated via the Lagrange interpolation involving the above vectors.

Here U^+, U^- indicate the involved function and fictitious values and $W_0^+, W_0^-, W_1^+, W_1^-$ represent the Lagrange interpolation weights when discretizing (14) along x-direction. Note that fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ in U^+ and U^- participate in the process of approximation to the x-direction interpolation. Analogously, U_Γ and P^+ are the needed auxiliary function values and corresponding weights to approximate u_y^+. The superscripts $-$ and $+$ in the above vector notations or the elements in each vector signify the Ω^- and Ω^+ domain. Beside, the subscript 0 or 1 indicates the zeroth or first order derivative. The index i or its variance tells the information located at $x = x_i$. Note that five auxiliary function values in U_Γ are at points $(x_o, y_{j-2}), (x_o, y_{j-1}), (x_o, y_j), (x_o, y_{j+1}), (x_o, y_{j+2})$, see Fig. 2. Additionally, each of these five auxiliary values are interpolated or extrapolated by five function values in Ω^+. For example, in Fig. 2, $u_{o,j+1}$ is interpolated by function values $u_{i,j+1}, u_{i+1,j+1}$ and $u_{i+2,j+1}, u_{i+3,j+1}, u_{i+4,j+1}$. Similarly $u_{o,j+2}$, $u_{o,j}, u_{o,j-1}$ and $u_{o,j-2}$ will be extrapolated by five function values from their right side in the Ω^+ domain.

With the function values in U_Γ appropriately approximated, (16) and (17) produce fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ at points (x_i, y_j) and (x_{i+1}, y_j) represented as a linear combination of surrounding points near the interface and three jump conditions at interface point (x^*, y^*). For instance, the representation of $\hat{u}_{i,j}$ can take a general form as

$$\begin{aligned} \hat{u}_{i,j} &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 [\![u]\!] + W_1 [\![u_\tau]\!] + W_2 [\![\beta u_\nu]\!] \\ &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 \phi(x^*, y^*) + W_1 \rho(x^*, y^*) + W_2 \psi(x^*, y^*), \end{aligned} \tag{18}$$

where $\mathbb{S}_{i,j}$ represents a set of surrounding nodes involved in (16) and (17).

In the second step, with two known fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ at P_2 and P_3, we can then generate fictitious values $\hat{u}_{i-1,j}, \hat{u}_{i+2,j}$ at P_1 and P_4 in a similar fashion. For accuracy concern, we embed the information of already formed fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ into the new process of interpolation. To be more clear, U^+ and U^- are now updated as

$$\begin{cases} U^+ = (\hat{u}_{i-1,j}, \hat{u}_{i,j}, u_{i+1,j}, u_{i+2,j}, u_{i+3,j}, u_{i+4,j})^T \\ U^- = (u_{i-3,j}, u_{i-2,j}, u_{i-1,j}, u_{i,j}, \hat{u}_{i+1,j}, \hat{u}_{i+2,j})^T \end{cases},$$

and correspondingly $W_0^+, W_0^-, W_1^+, W_1^-$ involve longer stencils for their weights. But U_Γ and P^+ preserve the same representation. With this setup, we may then obtain fictitious values $\hat{u}_{i-1,j}$ and $\hat{u}_{i+2,j}$, which also have the general representation form (18).

In this manner, four layers of fictitious values can be generated by repeatedly enforcing the jump conditions (14) and (15). One may refer to [58] for more details.

Remark 2.1. The above MIB formulation utilizes extrapolations in Ω^+ only, i.e., u_y^+ and u_x^+ in (14) and (15) are perpendicularly approximated by using auxiliary values, and each auxiliary value will be extrapolated by using function values in Ω^+. Recall that Eqs. (14) and (15) are derived after u_y^- or u_x^- being eliminated from Eqs. (11)-(13). On the other hand, if we remove u_y^+ or u_x^+ from Eqs. (11)-(13), we can have another set of formulations involving u_y^- or u_x^-. Numerically, u_y^- or u_x^- needs to be extrapolated by only using function values in Ω^-. Both approaches have been formulated in the previous MIB method [58]. Theoretically, these two approaches should be equivalent to each other. But in practical applications, there might be some minor numerical differences depending on the interface topology or the ratio of β^+ over β^-. We call the first approach as MIB-PLUS and second one as MIB-MINUS in terms of function values used from Ω^+ or Ω^- in the extrapolation process. A similar switching procedure has also been considered in other interface methods, such as [4].

2.2.2. *Fictitious values near interface* Γ_2

Generation of fictitious values required near the interface Γ_2 is relatively easier to realize by using the MIB boundary closure approach developed in [56]. Two layers of fictitious values are needed outside the interface Γ_2 as shown in Fig. 2. Take for instance the generation of fictitious values left to Γ_2^3 of the interface Γ_2 shown in Fig. 2. All these fictitious values can be formed using the interior function values from the x-direction as depicted in Fig. 3. The given Dirichlet boundary

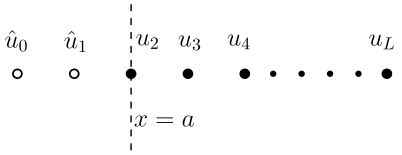


Fig. 3. Fictitious values $\hat{u}_0$ and $\hat{u}_1$ are generated in x-direction by using interior function values and related boundary condition at $(x, y) = (a, y_j)$.

condition (2) imposed at Γ_2^3 can be written as $u(a, y) = g(a, y)$. Combined with the governing equation $-\beta^+ \triangle u = f(x, y)$ in Ω^+, another condition can be derived as [56]

$$u_{xx}(a, y) = -\frac{f(a, y)}{\beta^+} - g_{yy}(a, y) \triangleq G(y).$$

For example, to obtain fictitious values along $y = y_j$ needed near Γ_2^3 of the interface Γ_2, we can adopt the condition

$$u_{xx}(a, y_j) = G(y_j),$$

which is then discretized by the MIB scheme [19,56]. For simplicity, we may ignore the dependence on y_j, and reduce the problem to an 1D problem as in Fig. 3. To approximate $u_{xx}(a)$, we use fictitious value $\hat{u}_1$ and real function values u_2, u_3, u_4, u_5,

$$C_{2,1}^{(2)} \hat{u}_1 + \sum_{i=2}^{5} C_{2,i}^{(2)} u_i = G(y_j), \tag{19}$$

where $C_{2,i}^{(2)}$ are finite difference weights in approximating second order derivative at $x = a$ by using a stencil $\{\hat{u}_1, u_2, u_3, u_4, u_5\}$. The notation 2 in first subscript of $C_{2,i}^{(2)}$ is because u_2 is the second point in the stencil.

As a result, unknown $\hat{u}_1$ can be solved as a linear combination of u_i for $i = 2, 3, 4, 5$ and $G(y_j)$.

$$\hat{u}_1 = \sum_{i=2}^{5} W_i u_i + W_o G(y_j) \tag{20}$$

Taking $\hat{u}_1$ as known, we can further determine another fictitious value $\hat{u}_0$ by one more iteration, which can also be represented as Eq. (20).

Besides the Dirichlet boundary condition, systematic approaches for treating Neumann and Robin boundary conditions have also been developed in [56]. Moreover, any combination of boundary conditions is allowed at four sides of Γ_2 without affecting the FFT computations [19]. This is a unique feature of the AMIB method, which is not shared by the other FFT based fast Poisson solvers.

2.3. Laplacian approximation

In the standard MIB methods [58,56], the fictitious values calculated above will be directly substituted into the fourth order central difference stencil to fulfill the approximation for Laplacian at irregular points. However, this leads to irregular matrix structures, which limit the computational efficiency. To accelerate the computation, an augmented procedure has been developed in [18,19] for preserving the standard discrete Laplacian in the MIB finite difference discretization. Such an augmented approach will be utilized in the present work for handling both interfaces Γ_1 and Γ_2 at the same time. The finite difference approximation to the Laplacian operator will be carried out in two steps, i.e., correcting central difference via Cartesian derivative jumps and reconstructing Cartesian derivative jumps.

2.3.1. Correcting central difference

For simplicity, we will assume a uniform grid spacing, i.e., $h_x = h_y = h$. With the availability of fictitious values, the four order central difference can be applied in a translation-invariant manner at every grid node. Moreover, the two-dimensional (2D) Laplacian can be approximated through one-dimensional (1D) finite difference in a tensor product style. The following theorem concerns on the correction of 1D central difference for a function $u = u(x)$ across an interface.

Theorem 1. Corrected fourth order finite differences. *Let $x_j \le \alpha \le x_{j+1}$, $h^- = x_j - \alpha$, and $h^+ = x_{j+1} - \alpha$. Suppose $u \in C^6[x_j - 2h, \alpha) \bigcap C^6(\alpha, x_{j+1} + 2h]$, with derivative extending continuously up to the interface α. Then the following approximations hold to $O(h^4)$ when $K = 5$:*

$$u_{xx}(x_{j-1}) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_{j-3}) + \frac{4}{3}u(x_{j-2}) - \frac{5}{2}u(x_{j-1}) + \frac{4}{3}u(x_j) - \frac{1}{12}u(x_{j+1})] + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h^+)^k}{k!}[u^{(k)}],$$

$$u_{xx}(x_j) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_{j-2}) + \frac{4}{3}u(x_{j-1}) - \frac{5}{2}u(x_j) + \frac{4}{3}u(x_{j+1}) - \frac{1}{12}u(x_{j+2})] \\ -\frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h^+)^k}{k!}[u^{(k)}] + \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h+h^+)^k}{k!}[u^{(k)}],$$

$$u_{xx}(x_{j+1}) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_{j-1}) + \frac{4}{3}u(x_j) - \frac{5}{2}u(x_{j+1}) + \frac{4}{3}u(x_{j+2}) - \frac{1}{12}u(x_{j+3})] \\ +\frac{4}{3h^2}\sum_{k=0}^{K}\frac{(h^-)^k}{k!}[u^{(k)}] - \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h^- - h)^k}{k!}[u^{(k)}],$$

$$u_{xx}(x_{j+2}) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_j) + \frac{4}{3}u(x_{j+1}) - \frac{5}{2}u(x_{j+2}) + \frac{4}{3}u(x_{j+3}) - \frac{1}{12}u(x_{j+4})] - \frac{1}{12h^2}\sum_{k=0}^{K}\frac{(h^-)^k}{k!}[u^{(k)}],$$

where $[u^{(k)}] = [u^{(k)}]_\alpha = \lim_{x\to\alpha^+} u^{(k)}(x) - \lim_{x\to\alpha^-} u^{(k)}(x)$.

It is seen in these corrected finite differences that the standard fourth order central difference is preserved. The corrected differences come with the interface correction quantities composed of different derivative jumps across the interface. Such correction terms occur for approximation at irregular points while they vanish at regular points, which means corrected differences reduce to standard differences for the regular point case. Theoretically, K needs to be at least 5 for a fourth order truncation error. However, in practice, it is sufficient for us to take $K = 4$ in the fourth order corrected differences with a third order local truncation error $O(h^3)$ for approximation to Laplacian, since global fourth order accuracy can still be preserved.

The corrected fourth order finite differences are applicable to deal with the original interface Γ_1 and the introduced interface Γ_2. We notice that the interface α in the Theorem 1 is located between x_i and x_{i+1} with $h^- = x_j - \alpha$, and $h^+ = x_{j+1} - \alpha$. But the Theorem is applicable in case when α coincides with a grid point just like for interface Γ_2. We here need to define $h^- = -h_x$ and $h^+ = 0$ for the part Γ_2^3, and define $h^- = 0$ and $h^+ = h$ for the part Γ_2^1. The same treatment applies to the lower part interface Γ_2^4 and upper part interface Γ_2^2.

The application of the corrected fourth order finite differences assumes a sufficient grid resolution in resolving a smoothly curved interface. In practice, a grid line may cut through the interface twice in a short distance, yielding two intersection points α_1 and α_2. But due to the derivation process of the corrected differences, at least two grid points should be available between α_1 and α_2 such that the above corrected formula is well defined. If the distance between α_1 and α_2 is too small, one has to refine the mesh. Thus, in dealing with the interface especially Γ_1, we need to monitor the local interface topology in order to apply the corrected difference in a safe mode.

Compared to the original MIB approximation to Laplacian at the irregular points near the interface $\Gamma = \Gamma_1 \cup \Gamma_2$, corrected differences can avoid irregular matrix structure, and preserve the standard central difference stencil. However, the correction term $[u^{(k)}]$ are introduced, and need to be numerically approximated due to the fact that those corrections terms are normally not given or not readily derivable from the given PDE problem. To maintain the third order truncation error in the fourth order corrected differences, it is sufficient to approximate each $[u^{(k)}]$ with error $O(h^{5-k})$. In the following subsection, we give a simple but systematic approach for the approximation with aid of fictitious values.

2.3.2. Approximation to derivative jumps

The derivative jumps in the fourth order corrected differences are defined as

$$[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} = \lim_{x\to\alpha^+}\frac{\partial^k u}{\partial x^k} - \lim_{x\to\alpha^-}\frac{\partial^k u}{\partial x^k}, \tag{21}$$

where $k = 0, 1, \cdots, 4$. The zeroth derivative in (21) is the function jump across the interface. Hence it can be analytically determined by the Eq. (3) or (8). As parameter k is up to $K = 4$, we can approximate each one-sided limit in (21) by constructing a Lagrange polynomial of degree 4, and compute each order derivative by differentiating the polynomial. In this way, the third order truncation error can be ensured. As shown in Fig. 4, the two layers of fictitious values are available on each side of the interface Γ_1 and outside Γ_2. So along each grid line, two fictitious values will be combined with three real function values on the other side of the interface to construct the degree 4 polynomial. One advantage for such a polynomial construction is that it uses central differences or pseudo-central differences to approximate these one-sided limits, see Fig. 4.

Consequently, derivative jumps can be approximated by using polynomials from both sides of the interface,

$$[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} \approx (w^k_{i-1,j}\hat{u}_{i-1,j} + w^k_{i,j}\hat{u}_{i,j} + \sum_{l=1}^{3} w^k_{i+l,j}u_{i+l,j}) \\ -(\sum_{l=1}^{3} w^k_{i-3+l,j}u_{i-3+l,j} + w^k_{i+1,j}\hat{u}_{i+1,j} + w^k_{i+2,j}\hat{u}_{i+2,j}), \tag{22}$$

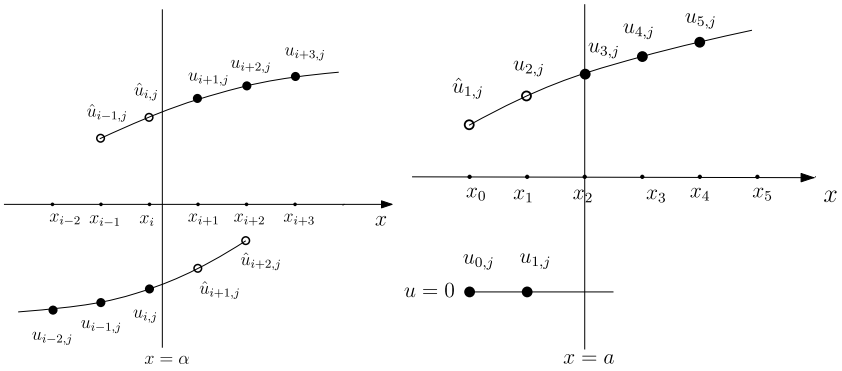


Fig. 4. The numerical approximation to the derivative jumps for the two types of interfaces. The left one is for Γ_1, while the right one is for the boundary Γ_2. In both charts, filled circles stand for real values while empty circles denote fictitious values.

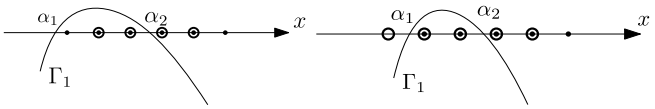


Fig. 5. Derivative jump approximation in two scenarios. Empty circles denote the fictitious values at the grid points, while black dots indicate real function values.

where $w^k_{p,q}$ stands for the weights for each function value in finite difference approximation at $x = \alpha$. Such an approximation has interpolation error of $O(h^{5-k})$ for the corresponding jump. Hence, the local truncation error in the fourth order corrected differences will keep third order of $O(h^3)$. The above approximation Eq. (22) takes a general form. In the case of approximation near Γ_2, limit from outside of the interface Γ_2 is trivially zero due to the zero-padding solution. In that case, Eq. (22) only involves approximation from the inside.

As mentioned in the last subsection, grid resolution with respect to interface change has to be monitored when applying the corrected differences. The same issue has to be taken care of too in constructing derivative jumps by Eq. (22). Two scenarios are shown in Fig. 5. When at least three grid points are available between two adjacent intersection points α_1 and α_2, such as the left chart of Fig. 5, Eq. (22) can be safely applied. For the right chart, in which only two grid points locate between α_1 and α_2, we need involve more fictitious values in approximation (22). For instance, in reconstructing derivative jumps at $\alpha = \alpha_2$, we may simply use fictitious value $\hat{u}_{i-3+l,j}$ instead of real function value $u_{i-3+l,j}$ in Eq. (22). Again, in case of only one grid point in between α_1 and α_2, a grid refinement is needed.

Remark 2.2. In the preceding subsection, we have mentioned two approaches to generate fictitious values near the interface Γ_1, i.e., MIB-PLUS and MIB-MINUS. It is found in our experiments that the accuracy of derivative jump reconstruction in Eq. (22) could be affected by the fictitious value generation from MIB-PLUS or MIB-MINUS depending on β^+ and β^-. When $\beta^+ > \beta^-$, the accuracy could be better if MIB-PLUS is used. Similarly, when $\beta^+ < \beta^-$, the accuracy could be better if MIB-MINUS is adopted. Nevertheless, for efficiency concern, we will use fictitious values obtained by MIB-PLUS in the case when $\beta^- > \beta^+$, while MIB-MINUS for the case with $\beta^- < \beta^+$. This strategy can reduce the iteration number in the overall algorithm. We note that such a selection strategy is essentially the same as that in Ref. [4] for reducing the condition number.

2.4. Formulation of augmented system

Comparing with the standard MIB method [58,56], a major advance of the AMIB method [18,19] is that Cartesian derivative jumps in the corrected central differences are treated as additional unknowns or auxiliary variables, so that the standard finite difference discretization of Laplacian operator can be preserved.

2.4.1. Auxiliary variables

At each interface point, which is an intersection point between a x or y grid line and Γ_1 or Γ_2 (see Fig. 1), a total of five auxiliary variables will be introduced and their algebraic equations will be formulated. In particular, after plugging the fictitious value representation Eq. (18) or (20) into Eq. (22), the approximation to the derivative jumps can be expressed as

$$\sum_{(x_I, y_J)\in \mathbb{S}_{i,j}} C_{I,J} u_{I,J} + [\frac{\partial^k u}{\partial x^k}] = C_o G + C_0 \phi + C_1 \rho + C_2 \psi, \tag{23}$$

where $C_{I,J}$ is the corresponding weights of function value $u_{I,J}$ in approximation to jump quantity $[\frac{\partial^k u}{\partial x^k}]$, and G, ϕ, ρ, and ψ are the known interface quantities. Taking $[\frac{\partial^k u}{\partial x^k}], k = 0, 1, \cdots, 4$ as auxiliary variables, system of equation (23) can be written in one matrix-vector form:

$$CU + IQ = \Phi, \tag{24}$$

where C stands for the weights coefficients matrix, U represents the function variables, $I \in \mathbb{R}^{5,5}$ is the identity matrix, Q is the introduced auxiliary variables, and Φ is from the known interface quantities. Similarly, approximation to jump quantities in the y-direction can be completed.

2.4.2. Augmented system

For fourth order study on 2D Poisson's problem (9), let $U_{i,j}$ indicate the discrete solution while $u(x_i, y_j)$ is analytical solution at (x_i, y_j). Based on the corrected difference analysis, the problem (9) can be discretized as

$$L_h U_{i,j} + S_{i,j} = f_{i,j}, \quad 1 \le i \le m+3, \quad 1 \le j \le n+3, \tag{25}$$

where $S_{i,j}$ is the correction term, and $L_h U_{i,j}$ is the standard fourth order central difference approximation to Laplacian with a degree of freedom $N_1 = (m+3) \times (n+3)$. Note that the correction term only exists at irregular points but vanishes at regular points. Making use of the introduced variables Q in Eq. (24), the correction term S can be written as $S = BQ$. Consequently, Eq. (25) sampling at different i and j gives rise to

$$AU + BQ = F, \tag{26}$$

where A is a symmetric, diagonally dominant matrix of dimension N_1 by N_1, B is a sparse matrix of dimension N_1 by N_2 consisting of coefficients from correction terms, and F is a vector with entries being f_{ij}. The vector Q involves auxiliary variables $[u]_i, [\frac{\partial u}{\partial x}]_i, [\frac{\partial^2 u}{\partial x^2}]_i, [\frac{\partial^3 u}{\partial x^3}]_i, [\frac{\partial^4 u}{\partial x^4}]_i$ for $i = 1, 2, \cdots$, and $[u]_j, [\frac{\partial u}{\partial y}]_j, [\frac{\partial^2 u}{\partial y^2}]_j, [\frac{\partial^3 u}{\partial y^3}]_j, [\frac{\partial^4 u}{\partial y^4}]_j$, for $j = 1, 2, \cdots$, at all intersection points between interface Γ and grid lines. The total number of auxiliary variables, denoted as N_2, is 5 times that of all intersection points. Thus, N_2 is one dimensionally smaller than N_1.

Let us abuse the notation by generalizing (24) from one interface point to all interface points

$$CU + IQ = \Phi, \tag{27}$$

where matrix C is a sparse matrix of dimension $N_2 \times N_1$, and I is a N_2 by N_2 identity matrix. An augmented matrix is therefore obtained by combining (26) and (27),

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

where

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

2.4.3. Schur complement

The matrix A enables fast solution via FFT. For this reason, we intend to obtain solution U by enforcing FFT inversion on

$$AU = F - BQ,$$

if we can move the determined correction term to the right-hand side. We will determine BQ through Schur complement, which means we eliminate U in (28) to form a linear system for Q,

$$(I - CA^{-1}B)Q = \Phi - CA^{-1}F. \tag{29}$$

Note that the linear system (29) for Q has a much smaller degree of freedom than U, i.e. N_2. An effective iteration solver can be used to solve for Q with aid from FFT inversion on A as A^{-1} is involved in (29).

The details of implementation on Schur complement system (29) with a biconjugate gradient method is formulated as below:

1. One determines right side $\hat{F} = \Phi - CA^{-1}F$ with one FFT for $A^{-1}F$ and some arithmetic operations.
2. For the left hand, the matrix vector product of $(I - CA^{-1}B)Q$ is completed in a few steps. Based on the equivalence to $IQ - CA^{-1}BQ$, a FFT is carried out on the product of BQ, that's to say, $A^{-1}(BQ)$. Then it is followed with simple multiplication and summation for $IQ - CA^{-1}(BQ)$. The transpose of $(I - CA^{-1}B)^T Q$ is conducted on $IQ - B^T A^{-1} C^T Q$ in a similar fashion.

3. A initial guess $Q = (0, 0, \cdots, 0)^T$ starts the biconjugate gradient iteration, and either the maximal iteration number 5000 or error tolerance equal to 10^{-12} terminates the iteration. The termination criteria are flexible according to actual computational needs.

Remark 2.3. The efficiency of the fourth order AMIB method depends on the iteration number for the biconjugate gradient iteration in step 2. Note that each iteration involves one FFT inversion with a complexity $O(N_1 \log N_1)$. Fortunately, the iteration is for solving (29), whose dimension $N_2 \ll N_1$. Generally, for any iteration solver, the better conditioned the iteration matrix is, the smaller the iteration number is. If iteration number does not increase or just increases moderately as the mesh is refined, we can obtain an overall complexity of $O(mn \log(mn))$ for the AMIB algorithm on a $(m+5) \times (n+5)$ mesh. In our numerical experiments, which will be presented in the next section, we have found that the iteration number weakly depends the mesh size.

2.5. FFT inversion

We finally discuss the inversion of matrix A by the FFT algorithm. The matrix A involved in the augmented system (28) represents the discrete Laplacian operator obtained via the fourth order central difference. As in the usual fast Poisson solution, the FFT inversion of A can be carried out in a 1D manner by using tensor products. Moreover, as demonstrated in [19], an anti-symmetric property at the boundary of the extended domain is automatically satisfied in the proposed AMIB formulation. Furthermore, with the corrected central differences and auxiliary variables, the FFT will not sense the solution discontinuities at the interface Γ_1 and boundary Γ_2. Therefore, the following 1D FFT algorithm is sufficient for the FFT inversion of A.

Without loss of generality, consider a 1D linear system with dimension $P = m + 3$,

$$Au = f, \tag{30}$$

where A is a symmetric, pentadiagonal matrix obtained through fourth order central difference discretization of u_{xx} [19]. The FFT inversion of (30) is obtained via the fast Sine transform as presented below:

1. Compute Sine transform for $f(x_j)$ via inverse fast Sine transform (IFST).
   $\hat{f}_l = \frac{2}{P+1} \sum_{j=1}^{P} f_j \sin(\frac{lj\pi}{P+1}), \quad \text{for} \quad l = 1, \cdots, P.$
2. $\hat{u}_l = \frac{\hat{f}_l}{\lambda_l}, \quad \text{for} \quad l = 1, 2, \cdots, P,$ where $\lambda_l = -\frac{1}{3h_x^2}[\cos(\frac{l\pi}{P+1}) - 1][\cos(\frac{l\pi}{P+1}) - 7].$
3. Compute u_j via fast Sine transform (FST): $u_j = \sum_{l=1}^{P} \hat{u}_l \sin(\frac{lj\pi}{P+1}), \quad \text{for} \quad j = 1, \cdots, P.$

3. Numerical experiments

We will verify the accuracy and efficiency of the proposed algorithm in this section. The performance of the proposed fourth order AMIB (AMIB4) method will be compared with the standard fourth order MIB (MIB4) method [58], as well as the second order AMIB (AMIB2) method developed in [18].

For simplicity, a square domain with uniform number of grids in each direction is assumed, i.e., $m = n$ and $h = h_x = h_y$. The accuracy of the numerical solution is measured by considering errors in the maximum norm and L_2 norm

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

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

where $u(x_i, y_j)$ and $u_h(x_i, y_j)$ are respectively analytical and numerical solution.

The accuracy in gradient and flux approximations will also be explored. To compute the discrete gradient $\nabla u_h(x_i, y_j)$, we adopt the fourth order central difference for both $\frac{\partial u_h}{\partial x}$ and $\frac{\partial u_h}{\partial y}$. For instance,

$$\frac{\partial u_h}{\partial x}(x_i, y_j) \triangleq -\frac{1}{12} u_h(x_{i-2}, y_j) + \frac{4}{3} u_h(x_{i-1}, y_j) - \frac{5}{2} u_h(x_i, y_j) + \frac{4}{3} u_h(x_{i+1}, y_j) - \frac{1}{12} u_h(x_{i+2}, y_j). \tag{31}$$

When the fourth order central difference stencil crosses the interface, we need to substitute the corresponding fictitious value into (31) as in the usual MIB scheme [58]. Then, the errors of gradient approximation can be measured in the maximum norm and L_2 norm

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

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

where $\nabla u(x_i, y_j)$ and $\nabla u_h(x_i, y_j)$ are respectively analytical and numerical gradient. The fluxes can be simply approximated by multiplying the numerical gradients with the corresponding β values.

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 error based a mesh spacing h_i for $i = 1, 2$, using the above defined norms on $(n+1)$ by $(n+1)$ mesh for the interested domain $\Omega^- \cup \Omega^+$. The solution calculation is facilitated by a FFT subroutine from Numerical Recipes [47] with 2^k summation. Due to the restriction of partition number equaling to 2^k in the subroutine, non-bisectional mesh refinement is usually conducted in the error analysis.

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

3.1. Accuracy studies

Example 1. Consider a 2D Poisson's equation

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

defined in a square $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with a circle interface defined by $r^2 := x^2 + y^2 = \frac{1}{4}$. The exact solution to this problem is prescribed as

$$u(x,y) = \begin{cases} e^{-x^2-0.5y^2}, & r \le 0.5, \\ \sin(kx)\sin(ky), & \text{otherwise}, \end{cases} \tag{32}$$

with the diffusion coefficient

$$\beta = \begin{cases} 1, & r \le 0.5, \\ 100, & \text{otherwise}, \end{cases}$$

and the parameter k is chosen to be 3. Here we call β as β^+ when it is outside the interface Γ, and β^- when it is inside the interface Γ. The same principle is followed for the below notations. The source term $q(x, y)$ is related to the above designated solution,

$$q(x,y) = \begin{cases} \beta^- e^{-x^2-0.5y^2}(4x^2+y^2-3), & r \le 0.5, \\ -2k^2\beta^+ \sin(kx)\sin(ky), & \text{otherwise}. \end{cases}$$

On the boundary of the domain, Dirichlet boundary condition is assumed with the boundary data derived by the analytical solution.

As the solution $u^+ = \sin(kx)\sin(ky)$ automatically satisfies the anti-symmetric property across the boundary of the given domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ (see Fig. 6 Left), the introduction of addition interface Γ_2 and extended domain Ω^e is unnecessary for this example. In the AMIB4 method, the augmented system is only constructed with respect to interface Γ_1 to carry out fourth order fast solver.

The comparisons among results of the AMIB4, AMIB2, and MIB4 are shown in Table 1. For all three methods, the mesh size is taken to be $(n+1)$ by $(n+1)$, and different mesh refinements with corresponding $[n, n]$ values are reported. It can be seen that all three methods achieve the desired order of convergence. Moreover, the AMIB method produces accurate approximation to the solution gradient. In particular, the AMIB2 and AMIB4 attain second and fourth order accuracy, respectively, in gradient approximation. The iteration numbers of three methods are also reported. It can be observed that the iteration numbers of both AMIB2 and AMIB4 only weakly depend on the mesh size n, but AMIB4 is superior to AMIB2 in several aspects. On the same mesh size, the AMIB4 can achieve a much better accuracy due to the higher order discretization. To reach the same accuracy level as that of the AMIB4 on mesh 65 by 65, the AMIB2 has to adopt a dense mesh 1025 by 1025. Correspondingly, the CPU cost of the AMIB2 is about 143 times more expensive than that of the AMIB4. On the other hand, the AMIB4 is also much improved in comparing with the MIB4. Both methods give rise to the same accuracy levels with different mesh sizes, while the AMIB4 is always faster. In particular, on mesh 513 by 513, the AMIB4 is about 26 times faster.

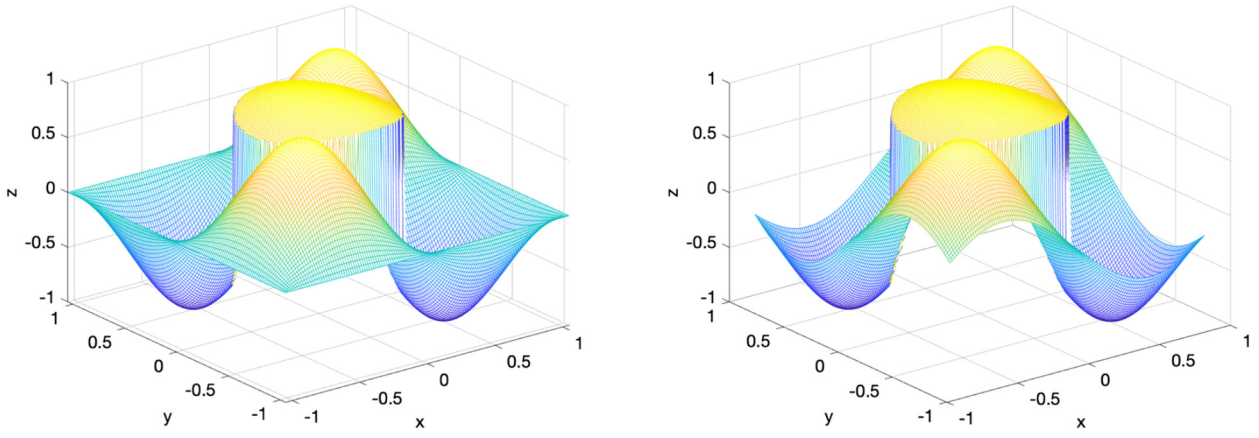


Fig. 6. The computed solution of Example 1 (left) and Example 2 (right) on a mesh with $n = 128$.

Example 2. In this example, the domain of Example 1 is redefined as a square $[-\frac{\pi}{3.5}, \frac{\pi}{3.5}] \times [-\frac{\pi}{3.5}, \frac{\pi}{3.5}]$ such that the anti-symmetric property is no longer satisfied (see Fig. 6 right), while everything else is the same. In treating such a Dirichlet boundary condition, an extended domain Ω^e is needed in the MIB4 scheme, and the mesh size for the entire domain $\Omega = \Omega^- \cup \Omega^+ \cup \Omega^e$ is taken as $(n+5)$ by $(n+5)$. The same domain partition is used in the MIB4 scheme. For the AMIB2 scheme, the mesh size is taken as $(n+1)$ by $(n+1)$ for $\Omega^- \cup \Omega^+$, because the anti-symmetric property is not required for second order FFT Poisson solver [18].

By considering $\beta^+ = 100$ and $\beta^- = 1$, the numerical errors of the AMIB2, AMIB4, and MIB4 are reported in Table 2. Again, the desired orders of convergence are numerically attained in three schemes and the accuracies of the AMIB4 are close to those of MIB4. By comparing the iteration numbers of the AMIB4 and AMIB2 in this example with those in Example 1, we found that such number does not change for the AMIB2, while increases a little bit for the AMIB4. This is because the AMIB4 scheme now contains two interfaces Γ_1 and Γ_2. In other words, the AMIB4 involves more auxiliary variables at Γ_2, on which Dirichlet boundary conditions are imposed. Nevertheless, such increment in comparing with Example 1 is so little such that the efficiency of the AMIB4 is almost the same. This is consistent with our previous study in [19], i.e., the iteration number of the AMIB4 in solving Poisson's problem on a rectangular domain with Dirichlet boundaries essentially does not increase as the mesh is refined. Overall, the proposed AMIB4 scheme is much more efficient than the AMIB2 and MIB4.

We next increase the diffusion coefficient to be $\beta^+ = 1000$ and $\beta^- = 1$, while keeping the other parameters unchanged. In Table 3, the numerical results from the AMIB4, AMIB2, and MIB4 are displayed. By comparing with Table 2, we found that the changes in both accuracies and efficiency are very minor.

Following the studies in [4], we further explore the impact of highly contrasted coefficients, by changing β^+ ranging from 10^{-4} to 10^4 with $\beta^- = 1$. The mesh size is chosen as $n = 508$, $n = 512$, and $n = 508$, respectively, for the AMIB4, AIMB2, and MIB4. As discussed above, two extrapolation procedures, i.e., MIB-PLUS and MIB-MINUS, will be utilized according to β^+ and β^-. In particular, our experiments show that the iteration number will be reduced if the MIB-PLUS is used when $\beta^- \geq \beta^+$, while the MIB-MINUS for the case with $\beta^- < \beta^+$.

In Fig. 7, the L_∞, L_2 errors, iteration number, and CPU time of the three methods are plotted against the β^+ value. Thanks to the selection strategy, the iteration number and CPU time of two AMIB methods are almost flat at both ends of β^+. This indicates that the condition number of the AMIB is not influenced by the high contrast in diffusion coefficients. The same result has been observed in [4] by using a similar selection scheme. The numerical errors are also flat when β^+ goes to the infinity, while they become larger as β^+ is approaching zero. In particular, at $\beta^+ = 10^{-4}$, the MIB4 fails to converge, while the AMIB4 still produces a decent accuracy. In short, these results demonstrate the effectiveness of our selection scheme and the robustness of the AMIB method.

Example 3. Consider a 2D Poisson's equation

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

over a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with an ellipse interface defined by

$$\Gamma : \frac{x^2}{0.5^2} + \frac{y^2}{0.3^2} = 1.$$

The exact solution to this problem is

$$u(x, y) = \begin{cases} \sin(kx)\cos(ky), & \text{inside } \Gamma, \\ \cos(kx)e^y, & \text{otherwise}, \end{cases} \tag{33}$$

Table 1
Example 1 -$\beta^+ = 100, \beta^- = 1$; Circle interface.

$[n, n]$	AMIB4					
	Solution					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[32, 32]	8.203E-5	–	2.469E-5	–	27	2.068E-2
[64, 64]	3.031E-6	4.76	9.484E-7	4.70	36	5.937E-2
[128, 128]	2.411E-7	3.65	5.544E-8	4.01	38	0.138
[256, 256]	1.481E-8	4.03	3.975E-9	3.80	57	0.566
[512, 512]	9.821E-10	3.91	3.099E-10	3.68	69	2.766
	Gradient					
	L_∞		L_2			
	Error	Order	Error	Order		
[32, 32]	3.416E-4	–	9.900E-5	–		
[64, 64]	2.067E-5	4.05	5.311E-6	4.22		
[128, 128]	3.629E-6	2.51	4.050E-7	3.71		
[256, 256]	1.486E-7	4.61	2.433E-8	4.06		
[512, 512]	1.702E-8	3.13	1.605E-9	3.92		
$[n, n]$	AMIB2					
	Solution					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[32, 32]	4.427E-3	–	1.910E-3	–	19	7.290E-3
[64, 64]	1.235E-3	1.84	5.023E-4	1.93	25	2.115E-2
[128, 128]	3.589E-4	1.78	1.310E-4	1.94	24	6.380E-2
[256, 256]	8.339E-5	2.10	3.165E-5	2.05	37	0.320
[512, 512]	1.892E-5	2.14	7.442E-6	2.10	39	1.488
[1024, 1024]	4.851E-6	1.96	1.893E-6	1.98	47	8.544
	Gradient					
	L_∞		L_2			
	Error	Order	Error	Order		
[32, 32]	1.636E-2	–	7.231E-3	–		
[64, 64]	4.849E-3	1.75	1.841E-3	1.97		
[128, 128]	1.720E-3	1.50	4.725E-4	1.96		
[256, 256]	3.615E-4	2.25	1.152E-4	2.04		
[512, 512]	8.488E-5	2.09	2.801E-5	2.04		
[1024, 1024]	2.278E-5	1.90	7.029E-6	2.00		
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[32, 32]	8.425E-5	–	2.604E-5	–	176	2.448E-2
[64, 64]	2.160E-6	5.29	7.019E-7	5.21	477	0.109
[128, 128]	3.422E-7	2.67	9.619E-8	2.87	799	0.492
[256, 256]	1.754E-8	4.29	4.833E-9	4.31	2311	5.793
[512, 512]	1.005E-9	4.14	3.236E-10	4.52	5527	58.35

with the diffusion coefficient

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

and the parameter k selected to be 5. The source term $q(x, y)$ is related to the above designated solution,

$$q(x, y) = \begin{cases} -2k^2\beta^- \sin(kx)\cos(ky), & \text{inside } \Gamma, \\ \beta^+(1-k^2)\cos(kx)e^y, & \text{otherwise}. \end{cases}$$

By considering Dirichlet boundary conditions, the numerical results of the AMIB4 and MIB4 are listed Table 4. In terms of accuracy and efficiency, the present results are very close to those in Example 2, and the AMIB4 is much faster than the MIB4. The numerical solution and error of the AMIB4 scheme are depicted in Fig. 8. It can be seen that the L_∞ error occurs around the ellipse interface Γ_1, while the error at Γ_2 is almost the same as regular points away from the interfaces. We note that a fourth order convergence is guaranteed in treating both interfaces Γ_1 and Γ_2. Errors near Γ_2 are small, simple

Table 2
Example 2a -$\beta^+ = 100$, $\beta^- = 1$; Circle interface.

$[n, n]$	AMIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[28, 28]	8.924E-5	–	2.580E-5	–	28	2.051E-2
[60, 60]	1.879E-6	5.48	6.137E-7	5.31	30	5.525E-2
[124, 124]	1.336E-7	3.75	4.368E-8	3.75	52	0.181
[252, 252]	9.553E-9	3.75	2.332E-9	4.16	58	0.621
[508, 508]	1.967E-9	2.24	2.919E-10	2.95	74	3.920
$[n, n]$	AMIB2					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[32, 32]	2.715E-3	–	1.132E-3	–	17	4.746E-3
[64, 64]	9.506E-4	1.51	3.321E-4	1.77	21	1.948E-2
[128, 128]	2.032E-4	2.23	7.505E-5	2.15	23	6.593E-2
[256, 256]	4.522E-5	2.17	1.718E-5	2.13	33	0.301
[512, 512]	1.150E-5	1.98	4.339E-6	1.99	44	1.643
[1024, 1024]	3.122E-6	1.88	1.144E-6	1.92	47	8.381
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[28, 28]	9.182E-5	–	2.588E-5	–	198	2.90E-2
[60, 60]	2.110E-6	4.95	6.245E-7	4.89	413	0.122
[124, 124]	1.362E-7	3.77	4.362E-8	3.67	1079	0.877
[252, 252]	1.018E-8	3.66	2.430E-9	4.07	2507	6.32
[508, 508]	1.141E-9	3.12	8.086E-11	4.85	8176	85.91

Table 3
Example 2b -$\beta^+ = 1000$, $\beta^- = 1$; Circle interface.

$[n, n]$	AMIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[28, 28]	9.060E-5	–	2.615E-5	–	28	2.128E-2
[60, 60]	1.906E-6	5.07	6.210E-7	4.91	30	5.150E-2
[124, 124]	1.357E-7	3.64	4.410E-8	3.64	56	0.181
[252, 252]	9.746E-9	3.72	2.358E-9	4.13	63	0.65
[508, 508]	7.400E-10	3.68	1.536E-10	3.90	75	3.12
$[n, n]$	AMIB2					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[32, 32]	2.744E-3	–	1.138E-3	–	17	6.660E-3
[64, 64]	9.600E-4	1.52	3.340E-4	1.77	22	2.127E-2
[128, 128]	2.051E-4	2.23	7.544E-5	2.15	23	6.895E-2
[256, 256]	4.569E-5	2.17	1.727E-5	2.13	34	0.334
[512, 512]	1.161E-5	1.98	4.358E-6	1.99	44	1.715
[1024, 1024]	3.153E-6	1.88	1.150E-6	1.92	50	9.34
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[28, 28]	9.322E-5	–	2.623E-5	–	201	3.014E-2
[60, 60]	2.141E-6	4.95	6.321E-7	4.89	361	0.113
[124, 124]	1.382E-7	3.77	4.413E-8	3.67	865	0.669
[252, 252]	1.041E-8	3.66	2.481E-9	4.07	2495	6.60
[508, 508]	1.106E-9	3.20	1.576E-10	3.93	8037	83.90

because Γ_2 is a straight interface. In general, it is much more difficult to design high order schemes in treating a curved interface over a Cartesian grid.

We also want to investigate the performance of the AMIB4 for treating other types of boundary conditions. Like the original AMIB4 scheme [19], the present AMIB4 method is able to handle any type of general boundary conditions. Moreover,

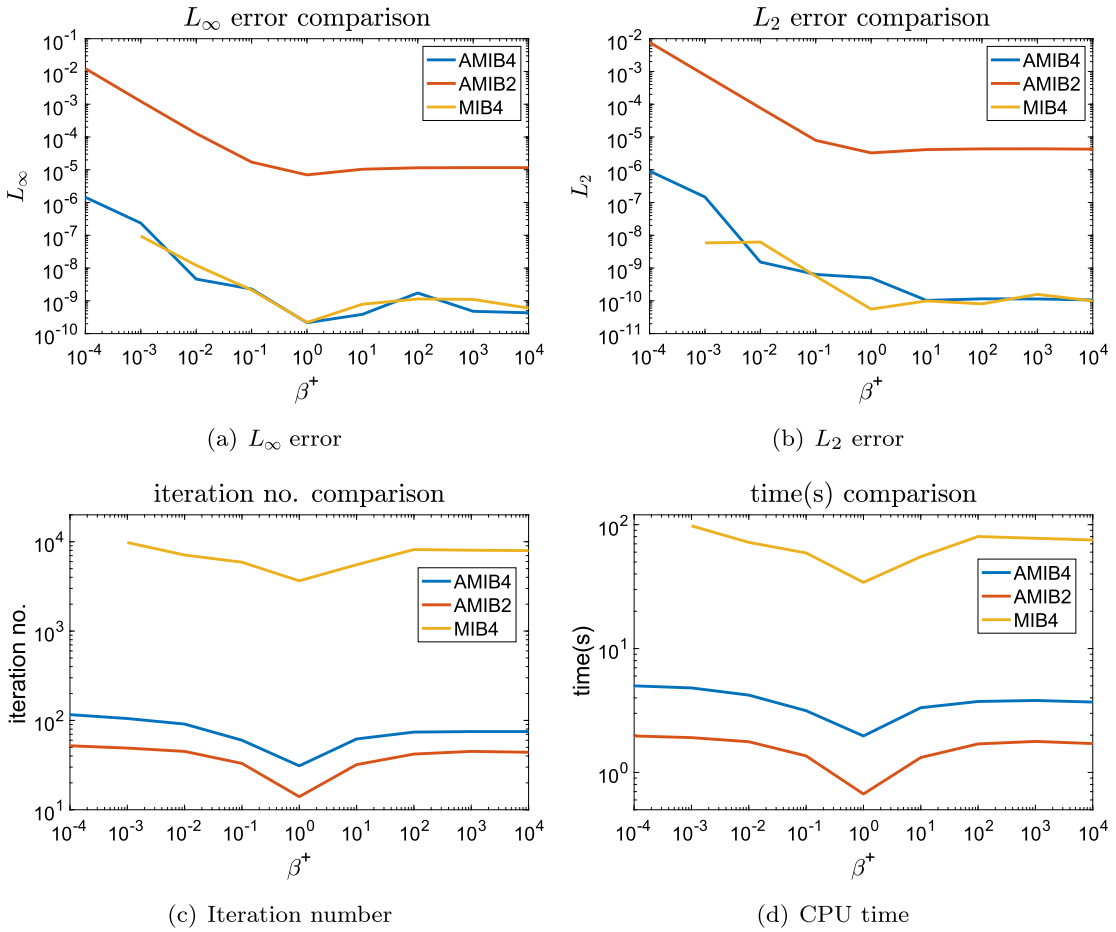


Fig. 7. Impact of highly contrasted coefficients to the MIB and AMIB methods for the circle interface problem. With a fixed $\beta^- = 1$, β^+ ranges from 10^{-4} to 10^4. Here, the mesh size for the AMIB4, AMIB2, and MIB4, respectively, is taken as $n = 508$, $n = 512$ and $n = 508$.

Table 4
Example 3a -$\beta^+ = 1, \beta^- = 10$; Ellipse interface; Dirichlet boundary condition.

$[n, n]$	AMIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	9.417E-4	–	3.966E-4	–	30	3.303E-2
[124, 124]	6.587E-5	3.66	2.743E-5	3.68	40	0.110
[252, 252]	3.528E-6	4.13	1.409E-6	4.19	52	0.489
[508, 508]	2.486E-7	3.78	1.040E-7	3.72	59	2.274
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	6.747E-4	–	2.769E-4	–	415	0.108
[124, 124]	7.157E-5	3.09	3.007E-5	3.06	1244	0.833
[252, 252]	3.019E-6	4.46	1.177E-6	4.57	3294	7.828
[508, 508]	2.388E-7	3.62	9.958E-8	3.52	7135	73.94

any mixed combination can be applied at the both ends of a Cartesian direction. In the present study, we consider Robin boundary conditions along the boundaries specified as

$$u + \frac{\partial u}{\partial v} = g(x, y),$$

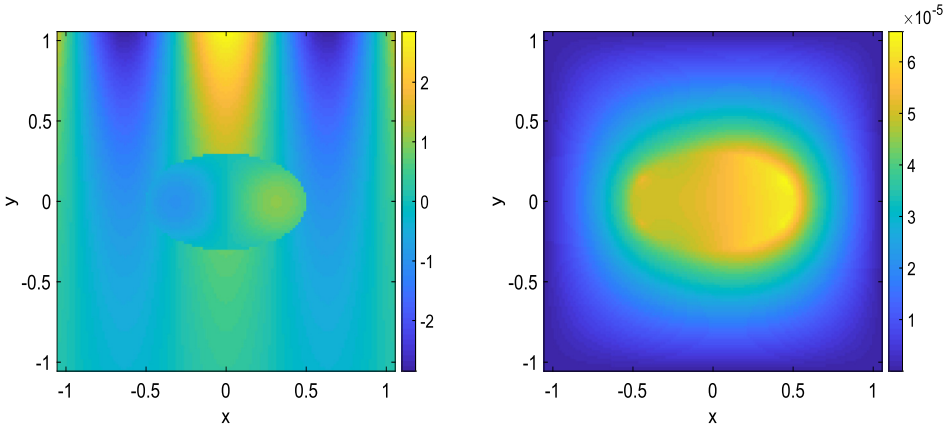


Fig. 8. The numerical solution (left) and error (right) of Example 3a on a mesh with $n = 124$.

Table 5
Example 3b -$\beta^+ = 1, \beta^- = 10$; Ellipse interface; Robin boundary condition.

$[n, n]$	AMIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	1.647E-3	–	1.019E-3	–	39	3.967E-2
[124, 124]	1.132E-4	3.69	6.923E-5	3.70	62	0.171
[252, 252]	5.973E-6	4.15	3.559E-6	4.19	98	0.915
[508, 508]	4.264E-7	3.77	2.612E-7	3.73	141	5.387
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	1.190E-3	–	7.315E-4	–	509	0.115
[124, 124]	1.232E-4	3.12	7.559E-5	3.13	1286	0.803
[252, 252]	5.093E-6	4.49	3.000E-6	4.55	3366	8.47
[508, 508]	4.096E-7	3.60	2.506E-7	3.54	7569	79.53

where $\vec{v}$ denotes the outer normal direction towards the rectangular domain. The numerical results are presented in Table 5. The comparison of new results with those of Dirichlet boundary conditions shows that the accuracy and order of convergence are almost the same. However, the iteration number of the AMIB4 scheme becomes larger. Such an increment is mainly due to the difficult nature of the Robin boundary condition, and has also been observed in the original AMIB4 scheme [19]. We note that the AMIB4 is still much more efficient than the MIB4 in solving this challenging problem. On a mesh of 513 by 513, the AMIB4 is about 15 times faster than the MIB4.

Example 4. We consider a three-leaf shaped interface. A 2D Poisson's equation

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

is studied here with an interface Γ

$$r = 0.6 + 0.1\sin(3\theta),$$

and two exact solutions of Poisson equation are designated by

$$u(x, y) = \begin{cases} \cos(kx)e^y & \text{inside } \Gamma, \\ e^x(x^2\sin(y) + y^2) & \text{otherwise}, \end{cases}$$

on a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The corresponding source terms are

$$q(x, y) = \begin{cases} \beta^-(1 - k^2)\cos(kx)e^y & \text{inside } \Gamma, \\ \beta^+e^x(2 + y^2 + (4x + 2)\sin(y)) & \text{otherwise}. \end{cases}$$

In this example, the parameters k is set to be 5, and the diffusion coefficients β^+ and β^- are chosen to be 20 and 1, respectively. Dirichlet boundary conditions are assumed.

Table 6
Example 4 -$\beta^+ = 20, \beta^- = 1$; Three-leaf shaped interface.

$[n, n]$	AMIB4					
	Solution					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[124, 124]	1.970E-5	–	6.649E-6	–	48	0.178
[252, 252]	1.104E-6	4.06	3.571E-7	4.12	58	0.659
[508, 508]	6.261E-8	4.09	2.113E-8	4.03	73	3.141
[1020, 1020]	3.611E-9	4.09	1.268E-9	4.04	103	19.74
	Gradient					
	L_∞		L_2			
	Error	Order	Error	Order		
[124, 124]	1.131E-4	–	2.094E-5	–		
[252, 252]	7.137E-6	3.90	1.132E-6	4.11		
[508, 508]	5.617E-7	3.63	6.623E-8	4.05		
[1020, 1020]	6.024E-8	3.20	3.898E-9	4.06		
$[n, n]$	MIB4					
	Solution					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	3.227E-4	–	1.137E-4	–	663	0.169
[124, 124]	1.567E-5	4.17	5.055E-6	4.29	1325	1.007
[252, 252]	8.770E-7	4.07	3.118E-7	3.93	4071	11.36
[508, 508]	5.276E-8	4.01	1.798E-8	4.07	13760	135.15

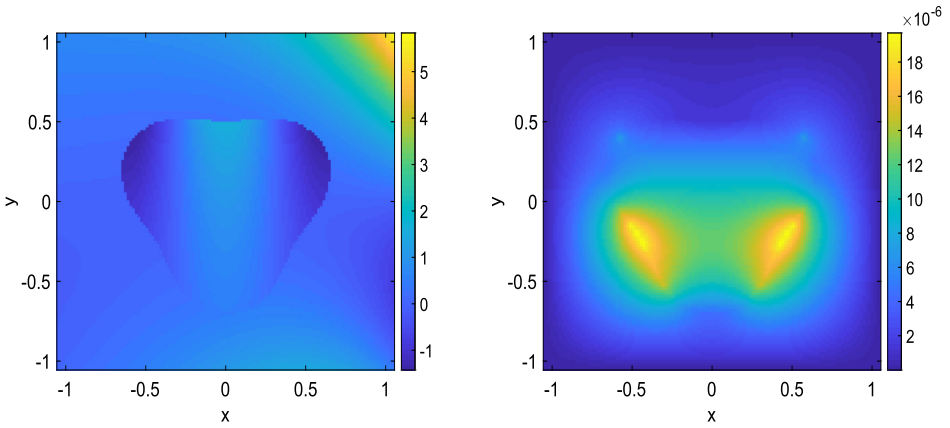


Fig. 9. The numerical solution (left) and error (right) of Example 4 in on a mesh with $n = 124$.

The numerical results are presented in Table 6. Obviously, the AMIB4 can achieve fourth order convergence in both solution and gradient. With Dirichlet boundary conditions, the iteration number of the AMIB4 grows slowly as mesh is refined, and the AMIB4 is much faster than the MIB4. The numerical solution and error of the AMIB4 are plotted in Fig. 9. Large errors are clustered around the interface Γ_1, particularly when the normal direction is not aligned with x or y direction. For the other part of Γ_1, such as the top portion, the interface is roughly parallel to a grid line, so that a less discretization error is invoked.

Example 5. We consider a five-leaf shaped interface. A 2D Poisson's equation

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

is studied here with an interface Γ

$$r = 0.5(1 + 0.2\sin(5\theta)),$$

and two exact solutions of Poisson equation designated by

Table 7
Example 5 -$\beta^+ = 1, \beta^- = 20$; Five-leaf shaped interface.

$[n, n]$	AMIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	1.078E-3	–	5.211E-4	–	43	4.442E-2
[124, 124]	5.885E-5	4.14	2.721E-5	4.13	70	0.223
[252, 252]	2.973E-6	4.06	1.229E-6	4.04	84	0.841
[508, 508]	9.272E-8	4.08	3.497E-8	4.08	100	3.744
[1020, 1020]	9.460E-9	2.76	2.263E-9	2.83	116	19.47
	Gradient					
	L_∞		L_2			
	Error	Order	Error	Order		
[60, 60]	4.824E-3	–	1.328E-3	–		
[124, 124]	2.300E-4	4.19	6.851E-5	4.08		
[252, 252]	3.221E-5	2.77	3.933E-6	4.03		
[508, 508]	8.286E-7	5.22	1.214E-7	4.96		
[1020, 1020]	3.586E-7	1.20	9.969E-9	3.59		
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	1.050E-3	–	5.003E-4	–	645	0.167
[124, 124]	5.782E-5	3.99	2.619E-5	4.06	1914	1.482
[252, 252]	1.462E-6	5.19	4.879E-7	5.61	3698	9.810
[508, 508]	9.594E-8	3.86	4.065E-8	3.54	11278	125.99

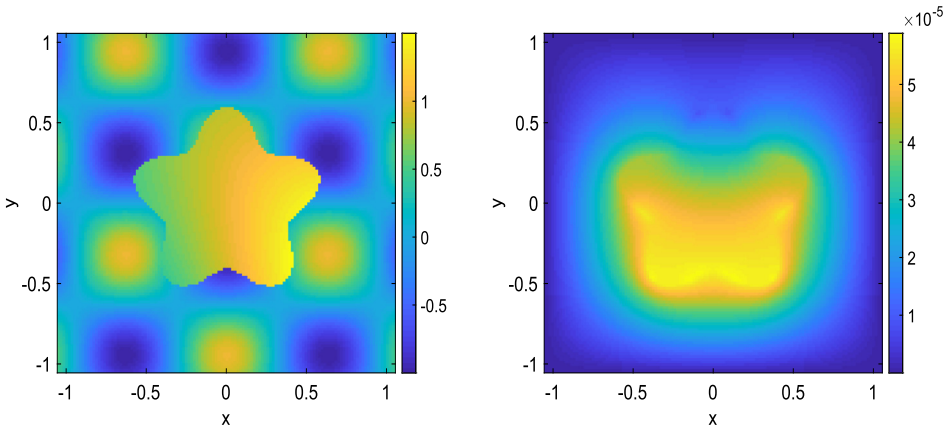


Fig. 10. The numerical solution (left) and error (right) of Example 5 on a mesh with $n = 124$.

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

on a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The corresponding source terms are

$$q(x, y) = \begin{cases} -\beta^- e^x(\cos(x + y) + 2\sin(x + y)) & \text{inside } \Gamma, \\ -2k^2\beta^+ \cos(kx)\sin(ky) & \text{otherwise}. \end{cases}$$

In this example, the parameter k is set to be 5, and the diffusion coefficients β^+ and β^- are chosen to be 1 and 20, respectively. Dirichlet boundary conditions are assumed.

The numerical results of both AMIB4 and MIB4 shown in Table 7 are comparable to those in the Example 4. The fourth order convergence is attained for both methods and for both solution and gradient. Also, the AMIB4 is much faster than the MIB4. The iteration numbers of the AMIB4 are larger than the previous example, mainly because the interface is more complicated. The fact that β contrast ratio is reversed may be a factor too. Nevertheless, the grow rate of iteration number with respect to mesh size n is still moderate, so that the efficiency of the AMIB4 is not reduced. The numerical error of the AMIB4 plotted in Fig. 10 shows that errors are large in four out of five leafs, but it is small for the top leaf. We believe

Table 8
Example 6 -multi-domain problem with two ellipse interfaces; Dirichlet boundary condition.

$[n, n]$	AMIB4					
	Solution					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	7.859E-3	–	2.937E-3	–	53	8.541E-2
[124, 124]	6.329E-4	3.47	2.379E-4	3.46	56	0.239
[252, 252]	2.805E-5	4.39	1.093E-5	4.34	82	1.022
[508, 508]	1.692E-6	4.01	6.538E-7	4.02	101	4.58
	Gradient					
	L_∞		L_2			
	Error	Order	Error	Order		
[60, 60]	3.505E-2	–	9.516E-3	–		
[124, 124]	2.995E-3	3.39	7.608E-4	3.48		
[252, 252]	1.351E-4	4.37	3.442E-5	4.37		
[508, 508]	8.190E-6	4.00	2.059E-6	4.02		
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	7.000E-3	–	2.607E-3	–	848	0.200
[124, 124]	6.210E-4	3.24	2.326E-4	3.33	2036	1.34
[252, 252]	2.810E-5	4.37	1.096E-5	4.31	5044	13.11
[508, 508]	1.706E-6	4.00	6.588E-7	4.01	12753	144.15

this is due to the solution, see Fig. 10 left. The solution changes rapidly near the interface for the other four leafs, but such change is relatively slower for the top leaf.

Example 6. We next study a problem with multiple subdomains. Consider a 2D Poisson's equation

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

defined in a square domain $\Omega = [-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. To test the robustness of the AMIB4, we divide Ω into three regions by two ellipse shape interfaces:

$$\Gamma_L: \left(\frac{x+\frac{\pi}{6}}{\frac{\pi}{12}}\right)^2 + \left(\frac{y+\frac{\pi}{8}}{\frac{\pi}{9}}\right)^2 = 1$$
$$\Gamma_R: \left(\frac{x-\frac{\pi}{6}}{\frac{\pi}{9}}\right)^2 + \left(\frac{y-\frac{\pi}{8}}{\frac{\pi}{12}}\right)^2 = 1$$

Three pieces of solutions are defined in each subdomain

$$u(x, y) = \begin{cases} \cos(kx)\sin(ky) & \text{inside } \Gamma_L, \\ e^{-x-0.5y^2} & \text{inside } \Gamma_R, \\ \sin(kx)\cos(ky) & \text{otherwise.} \end{cases}$$

The diffusion coefficients are also defined respectively as

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

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

The numerical solutions are summarized in Table 8. The fourth order convergences are again confirmed. The efficiency of the AMIB4 over MIB4 is further validated. In particular, based on a mesh 513 by 513, the execution time of the AMIB4 is 4.58 seconds, while the MIB4 costs up to 144.15 second, which is much more time-consuming. For the numerical error shown in Fig. 11, we see that interface Γ_L incurs a much larger error than Γ_R, due to two factors. First, β contrast is 10 times larger for Γ_L. Second, a rapid function variation is presented inside Γ_L, changing from positive to negative, while inside Γ_R solution is always positive.

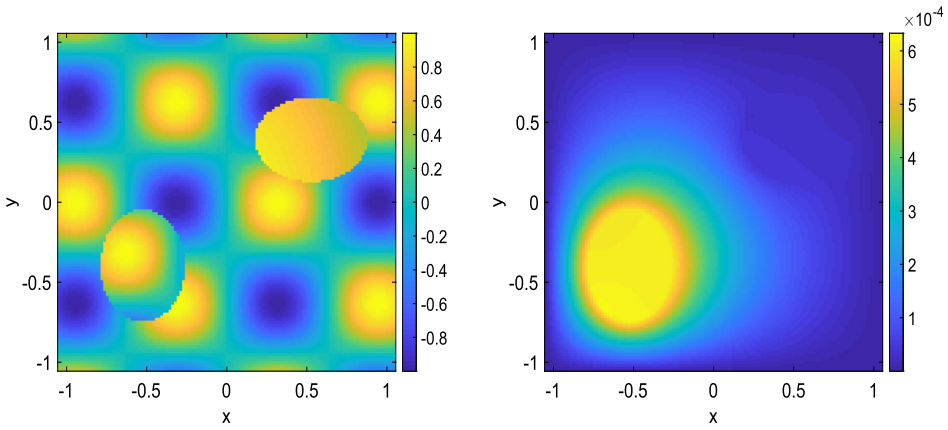


Fig. 11. The numerical solution (left) and error (right) of Example 6 on a mesh with $n = 124$.

Table 9
Example 7 multi-domain problem with two ellipse interfaces; Robin boundary condition.

$[n, n]$	AMIB4					
	Solution					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	2.807E-4	–	5.172E-5	–	43	6.044E-2
[124, 124]	1.260E-5	4.28	3.541E-6	3.69	56	0.188
[252, 252]	9.545E-7	3.64	2.764E-7	3.60	82	0.904
[508, 508]	5.813E-8	3.99	1.716E-8	3.96	121	4.74
	Gradient					
	L_∞		L_2			
	Error	Order	Error	Order		
[60, 60]	5.449E-3	–	2.669E-4	–		
[124, 124]	5.313E-4	3.20	1.388E-5	4.07		
[252, 252]	1.281E-5	5.25	8.255E-7	3.98		
[508, 508]	2.672E-6	2.23	5.222E-8	3.94		
	Fluxes					
	L_∞		L_2			
	Error	Order	Error	Order		
[60, 60]	28.73	–	2.088	–		
[124, 124]	2.396	3.42	0.108	4.08		
[252, 252]	0.128	4.12	7.420E-3	3.78		
[508, 508]	9.354E-3	3.73	4.644E-4	3.95		
$[n, n]$	MIB4					
	L_∞		L_2		Iter no.	CPU time (s)
	Error	Order	Error	Order		
[60, 60]	2.689E-4	–	5.068E-5	–	587	0.152
[124, 124]	1.288E-5	4.19	3.658E-6	3.62	1487	1.178
[252, 252]	9.486E-7	3.68	2.787E-7	3.63	3654	10.31
[508, 508]	5.725E-8	4.00	1.710E-8	3.98	9912	96.88

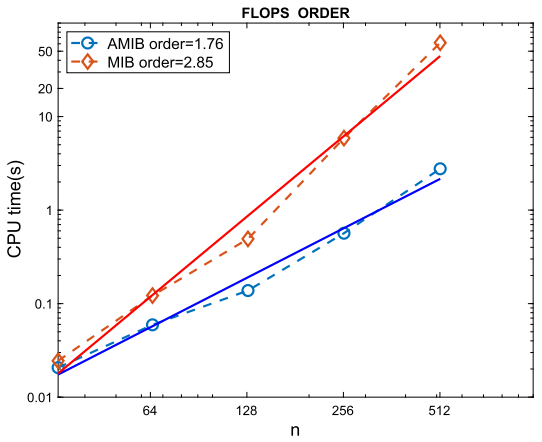
Example 7. In this last example, we further test the performance of our algorithm for a Helmholtz type problem

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

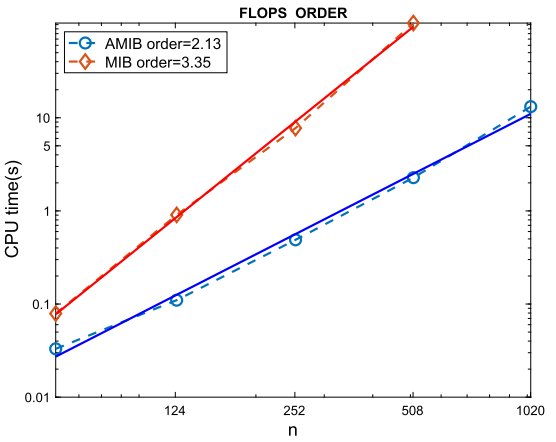
The domain, interface, and solution of Example 6 are re-used. The diffusion coefficients are now defined respectively as

$$\beta = \begin{cases} 1, & \text{inside } \Gamma_L, \\ 10, & \text{inside } \Gamma_R. \\ 10000, & \text{otherwise.} \end{cases}$$

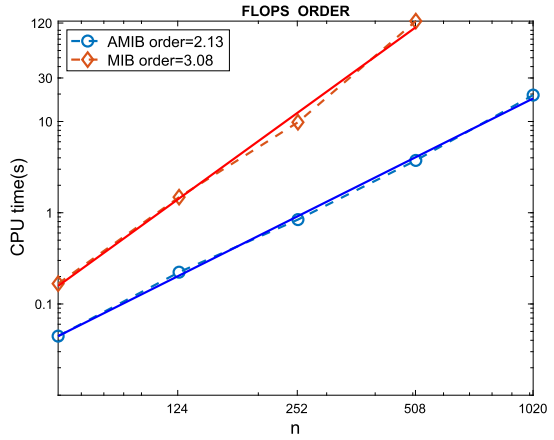
On the boundaries of Ω, Robin boundary conditions are assumed



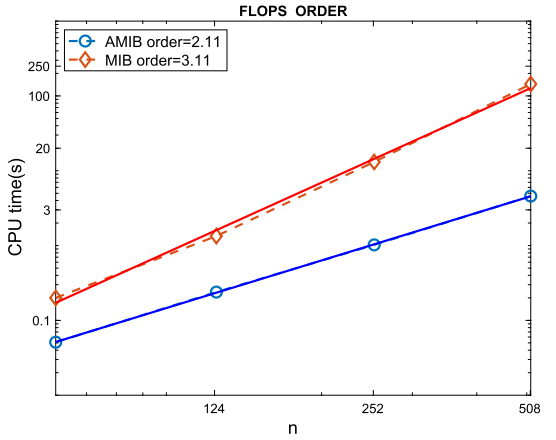
(a) Example 1.



(b) Example 3a.



(c) Example 5.



(d) Example 6.

Fig. 12. Flops order in CPU time is examined for several examples. Here n represents the degree of freedom in each direction.

$$10u + \frac{\partial u}{\partial v} = g(x, y),$$

where $\vec{v}$ denotes the outer normal direction. The source term $q(x, y)$ and boundary date $g(x, y)$ are determined by the analytical solution.

Similar to the previous study, the impact of Robin boundary conditions can be clearly seen in Table 9. The iteration number of the AMIB4 becomes larger when Dirichlet boundary conditions are replaced by Robin boundary conditions, but the growth of iteration number with respect to n is still moderate. Overall, the AMIB4 is still much more efficient than the MIB4. Besides the recovery of gradient, the flux approximation has also drawn some attentions in the literature [15]. In Table 9, the flux approximation by the AMIB4 method is also investigated. In particular, each component of fluxes $(\beta u_x, \beta u_y)$ is approximated with formula (31) or its $\frac{\partial u_h}{\partial y}$ form multiplied by the corresponding β in each subdomain. Then we compare such numerical approximations to the true solution fluxes under the L_∞ and L_2 norms. It is observed that the flux approximations also preserve the fourth order convergence. The fact that the magnitude of the numerical fluxes is larger than that of gradients is because a large diffusion coefficient $\beta = 10000$ is used.

3.2. Computational efficiency

In all numerical examples above, we have reported the computational time of the AMIB4 and MIB4 methods. The AMIB4 is found to be much faster in all cases. In this subsection, we would like to numerically quantify the complexity of both methods. For this purpose, Fig. 12 compares the computational efficiency of AMIB4 and MIB4 for four examples by plotting CPU time versus n in a log-log manner. Recall the mesh size is $(n+1)$ by $(n+1)$ in Example 1, while in all other examples, it is $(n+5)$ by $(n+5)$. For simplicity, we treat the degree of freedom in each direction as n, and we plot CPU time against n

in Fig. 12. A least squares fitting is conducted in each case to express the CPU time in the form of n^r, and the corresponding order r is reported in the legend of each subfigure. It is clear that the complexity of the MIB4 is at least $O(n^3)$. For the AMIB4 scheme, such r values are usually slightly above 2, which meets the expected complexity of $O(n^2 \log n)$ for our designed algorithm.

4. Summary

In this work, an efficient fourth order finite difference algorithm is proposed to solve two-dimensional elliptic interface problems with help of FFT. The proposed AMIB method is built based on our previous developments, including the fourth order MIB [58,56] and second order AMIB [18] for treating smoothly curved interfaces and the fourth order AMIB [19] for handling various boundary conditions. A systematic approach is constructed at both interfaces and boundaries to correct the fourth order central difference discretization of Laplacian operator. A uniform augmentation framework is utilized to accelerate the computation at both interfaces and boundaries. In the Schur complement solution of the augmented linear system, the AMIB method is well conditioned such that the iteration number of the biconjugate gradient solver only weakly depends on the mesh size n. Thus, the FFT-AMIB4 can deliver a computational complexity of $O(n^2 \log n)$ on a $n \times n$ mesh. Therefore, for the first time in the literature, a Cartesian grid finite difference algorithm is able to not only achieve a fourth order of accuracy, but also maintain the $O(n^2 \log n)$ efficiency of FFT or multigrid fast Poisson solvers, in solving elliptic interface problems with curved interfaces and any boundary conditions. Moreover, the proposed AMIB method is found to produce a fourth order of convergence in approximating solution gradient.

In the future, we will investigate potential approaches to control the condition number of the linear system resulting from discretization so that the efficiency of the AMIB method could be further improved. The extension of the proposed AMIB method to more challenging interface problems will also be explored. A second order MIB scheme has been developed in the literature to deal with sharp-edged interfaces, such as cusps and tips [54]. Based on fictitious values generated by such a MIB scheme, the FFT-AMIB scheme could be generalized to handle non-smooth interfaces. The extension of the present AMIB scheme to handle three dimensional interfaces is also tractable, and will be reported elsewhere. However, the application of the FFT-AMIB scheme to biomolecular electrostatics [42,21] faces additional challenges when a nonlinear term is involved in the Poisson-Boltzmann equation. This difficulty may be overcome by developing an iterative nonlinear relaxation algorithm, which will be investigated in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

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

References

[1] L. Adams, Z.L. Li, The immersed interface/multigrid methods for interface problems, SIAM J. Sci. Comput. 24 (2002) 463–479.
[2] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, Computing 5 (1970) 207–213.
[3] J. Bedrossian, J.H. von Brecht, S.W. Zhu, E. Sifakis, J.M. Teran, A finite element method for interface problems in domains with smooth boundaries and interfaces, J. Comput. Phys. 229 (2010) 6405–6426.
[4] D. Bochkov, F. Gibou, Solving elliptic interface problems with jump conditions on Cartesian grids, J. Comput. Phys. 407 (2020) 109269.
[5] J. Bramble, J. King, A finite element method for interface problems in domains with smooth boundaries and interfaces, Adv. Comput. Math. 6 (1996) 109–138.
[6] O. Bruno, M. Lyon, High-order unconditionally stable FC-AD solvers for general smooth domains, I: basic elements, J. Comput. Phys. 229 (2010) 2009–2033.
[7] E. Burman, P. Hansbo, Interior-penalty-stabilized Lagrange multiplier methods for the finite-element solution of elliptic interface problems, IMA J. Numer. Anal. 30 (2010) 870–885.
[8] L. Chen, H. Wei, M. Wen, An interface-fitted mesh generator and virtual element methods for elliptic interface problems, J. Comput. Phys. 334 (2017) 327–348.
[9] T. Chen, J. Strang, Piecewise-polynomial discretization and Krylov-accelerated multigrid for elliptic interface problems, J. Comput. Phys. 227 (2008) 7503–7542.
[10] Z. Chen, J. Zou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numer. Math. 79 (1998) 175–202.
[11] I.-L. Chern, Y.-C. Shu, A coupling interface method for elliptic interface problems, J. Comput. Phys. 225 (2007) 2138–2174.
[12] 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.
[13] John Dolbov, Isaac Harari, An efficient finite element method for embedded interface problems, Int. J. Numer. Methods Eng. 78 (2009) 229–252.
[14] M. Dryjaa, J. Galvisb, M. Sarkisb, BDDC methods for discontinuous Galerkin discretization of elliptic problems, J. Complex. 23 (2007) 715–739.
[15] R. Egan, F. Gibou, xGFM: recovering convergence of fluxes in the ghost fluid method, J. Comput. Phys. 409 (2020) 109351.
[16] R.E. Ewing, Z.L. Li, T. Lin, Y.P. Lin, The immersed finite volume element methods for the elliptic interface problems, Math. Comput. Simul. 50 (1999) 63–76.
[17] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), J. Comput. Phys. 152 (1999) 457–492.

[18] H. Feng, G. Long, S. Zhao, An augmented matched interface and boundary (MIB) method for solving elliptic interface problem, J. Comput. Appl. Math. 361 (2019) 426–433.
[19] H. Feng, S. Zhao, FFT-based high order central difference schemes for the three-dimensional Poisson equation with various types of boundary conditions, J. Comput. Phys. 410 (2020) 109391.
[20] T.-P. Fries, T. Belytschko, The intrinsic XFEM: a method for arbitrary discontinuities without additional unknowns, Int. J. Numer. Methods Eng. 68 (13) (2006) 1358–1385.
[21] W. Geng, S. Zhao, A two-component Matched Interface and Boundary (MIB) regularization for charge singularity in implicit solvation, J. Comput. Phys. 351 (2017) 25–39.
[22] A. Guittet, M. Lepilliez, S. Tanguy, F. Gibou, Solving elliptic problems with discontinuities on irregular domains – the Voronoi interface method, J. Comput. Phys. 298 (2015) 747–765.
[23] H. Guo, X. Yang, Gradient recovery for elliptic interface problem, II: immersed finite element methods, J. Comput. Phys. 338 (2017) 606–619.
[24] G.R. Hadley, High-accuracy finite difference equations for dielectric waveguide analysis, I: uniform regions and dielectric interfaces, J. Lightwave Technol. 20 (2002) 1210–1218.
[25] A. Hansbo, P. Hansbo, An unfitted finite element method, Comput. Methods Appl. Mech. Eng. 191 (2002) 5537–5552.
[26] A. Hansbo, P. Hansbo, A finite element method for the simulation of strong and weak discontinuities in solid mechanics, Comput. Methods Appl. Mech. Eng. 193 (2004) 3523–3540.
[27] J.L. Hellrung Jr., L.M. Wang, E. Sifakis, 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.
[28] T.Y. Hou, Z.L. Li, S. Osher, H. Zhao, A hybrid method for moving interface problems with application to the Hele-Shaw flow, J. Comput. Phys. 134 (1997) 236–252.
[29] S.M. Hou, X.-D. Liu, A numerical method for solving variable coefficient elliptic equation with interfaces, J. Comput. Phys. 202 (2005) 411–445.
[30] S.M. Hou, P. Song, L.Q. Wang, H.K. Zhao, A weak formulation for solving elliptic interface problems without body fitted grid, J. Comput. Phys. 249 (2013) 80–959.
[31] L.N.T. Huynh, N.C. Nguyen, J. Peraire, B.C. Khoo, A high-order hybridizable discontinuous Galerkin method for elliptic interface problems, Int. J. Numer. Methods Eng. 93 (2013) 183–200.
[32] H. Ji, J. Dolbow, On strategies for enforcing interfacial constraints and evaluating jump conditions with the extended finite element method, Int. J. Numer. Methods Eng. 61 (14) (2004) 2508–2535.
[33] R.J. LeVeque, Z.L. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.
[34] J. Li, J.M. Melenk, B. Wohlmuthc, J. Zou, Optimal a priori estimates for higher order finite elements for elliptic interface problems, Appl. Numer. Math. 60 (2010) 19–37.
[35] Z.L. Li, A fast iterative algorithm for elliptic interface problem, SIAM J. Numer. Anal. 35 (1998) 230–254.
[36] Z.L. Li, K. Ito, Maximum principle preserving schemes for interface problems with discontinuous coefficients, SIAM J. Sci. Comput. 23 (2001) 339–361.
[37] Z.L. Li, K. Ito, The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains, SIAM, 2006.
[38] Z.L. Li, H. Ji, X. Chen, Accurate solution and gradient computation for elliptic interface problems with variable coefficients, SIAM J. Numer. Anal. 55 (2017) 670–697.
[39] C. Li, Z. Wei, G. Long, C. Campbell, S. Ashlyn, S. Zhao, Alternating direction ghost-fluid methods for solving the heat equation with interfaces, Comput. Math. Appl. 80 (2020) 714–732.
[40] C. Liu, C. Hu, A second order ghost fluid method for an interface problem of the Poisson equation, Commun. Comput. Phys. 22 (2017) 965–996.
[41] X. Liu, R. Fedkiw, M. kang, A boundary condition capturing method for Poisson's equation on irregular domains, J. Comput. Phys. 160 (2000) 151–178.
[42] B.Z. Lu, Y.C. Zhou, M.J. Holst, J.A. McCammon, Recent progress in numerical methods for the Poisson-Boltzmann equation in biophysical applications, Commun. Comput. Phys. 3 (2008) 973–1009.
[43] M. Lyon, O. Bruno, High-order unconditionally stable FC-AD solvers for general smooth domains, II: elliptic, parabolic and hyperbolic PDEs; theoretical considerations, J. Comput. Phys. 229 (2010) 3358–3381.
[44] L. Mu, J. Wang, G.W. Wei, X. Ye, S. Zhao, Weak Galerkin methods for second order elliptic interface problems, J. Comput. Phys. 250 (2013) 106–125.
[45] L. Mu, J. Wang, X. Ye, S. Zhao, A new weak Galerkin finite element method for elliptic interface problems, J. Comput. Phys. 325 (2016) 157–173.
[46] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[47] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in Fortran: the Art of Scientific Computing, 2nd edition, Cambridge University Press, 1992.
[48] D.B. Stein, R.D. Guy, B. Thomases, Immersed boundary smooth extension: a high-order method for solving PDE on arbitrary smooth domains using Fourier spectral methods, J. Comput. Phys. 304 (2016) 252–274.
[49] F. Tong, W. Wang, X. Feng, J. Zhao, Z. Li, How to obtain an accurate gradient for interface problems?, J. Comput. Phys. 405 (2020) 109070.
[50] A. Wiegmann, K.P. Bube, The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth solutions, SIAM J. Numer. Anal. 37 (2004) 827–862.
[51] K. Xia, M. Zhan, G.W. Wei, MIB Galerkin method for elliptic interface problems, J. Comput. Appl. Math. 272 (2014) 195–220.
[52] Y. Xie, W. Ying, A fourth-order kernel-free boundary integral method for the modified Helmholtz equation, J. Sci. Comput. 78 (2019) 1632–1658.
[53] W. Ying, W.C. Wang, A kernel-free boundary integral method for implicitly defined surfaces, J. Comput. Phys. 252 (2013) 606–624.
[54] S. Yu, Y. Zhou, G.W. Wei, Matched interface and boundary (MIB) method for elliptic problems with sharp-edged interfaces, J. Comput. Phys. 224 (2010) 729–756.
[55] 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.
[56] S. Zhao, G.W. Wei, Matched interface and boundary (MIB) for the implementation of boundary conditions in high order central finite differences, Int. J. Numer. Methods Eng. 77 (2009) 1690–1730.
[57] 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.
[58] Y.C. Zhou, S. Zhao, M. Feig, 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.