Journal of Computational and Applied Mathematics 361 (2019) 426–443

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

An augmented matched interface and boundary (MIB) method for solving elliptic interface problem

Hongsong Feng [a,b], Guangqing Long [a], Shan Zhao [b,*]

[a] Department of Mathematics, Nanning Normal University, Nanning 530001, PR China
[b] Department of Mathematics, University of Alabama, Tuscaloosa AL, 35487, USA

ARTICLE INFO

Article history:
Received 25 September 2018
Received in revised form 8 February 2019

Keywords:
Elliptic interface problem
Immersed interface method (IIM)
Matched interface and boundary (MIB)
Fast Poisson solver
Schur complement

ABSTRACT

In this paper, a second order accurate augmented matched interface and boundary (MIB) is introduced for solving two-dimensional (2D) elliptic interface problems with piecewise constant coefficients. The augmented MIB seamlessly combines several key ingredients of the standard MIB, augmented immersed interface method (IIM), and explicit jump IIM, to produce a new fast interface algorithm. Based on the MIB, zeroth and first order jump conditions are enforced across an arbitrarily curved interface, which yields fictitious values on Cartesian nodes near the interface. By using such fictitious values, a simple procedure is proposed to reconstruct Cartesian derivative jumps as auxiliary variables and couple them with the jump-corrected Taylor series expansions, which allow us to restore the order of the central difference across the interface to two. Moreover, by using the Schur complement to disassociate the algebraic computation of auxiliary variables and function values, the discrete Laplacian can be efficiently inverted by using the fast Fourier transform (FFT). It is found in our numerical experiments that the iteration number in solving the auxiliary system weakly depends on the mesh size. As a consequence, the total computational cost of the augmented MIB is about $O(n^2 \log n)$ for a Cartesian grid with dimension $n \times n$ in 2D. Therefore, the augmented MIB outperforms the classical MIB in all cases by significantly reducing the CPU time, while keeping the same second order of accuracy in dealing with complicated interfaces.

1. Introduction

Elliptic interface problem has drawn a great amount of attention due to its wide application in many fields such as computational electromagnetics and optics [1,2], biomolecular electrostatics [3,4], and material science [5]. Its mathematical model is featured by a regular or mostly irregular interface, across which there exist discontinuous coefficients and singular source. A typical two-dimensional (2D) elliptic interface problem can be formulated through the Poisson equation

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

with the Dirichlet boundary condition

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

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

https://doi.org/10.1016/j.cam.2019.05.004
0377-0427/

For simplicity, we assume the domain Ω being a rectangular one in 2D. In the domain, there is an interface Γ separating Ω into two subdomains $\Omega = \Omega^+ \cup \Omega^-$. The interface is defined as $\Gamma = \Omega^+ \cap \Omega^-$. The coefficient $\beta(x, y)$ and source term $f(x, y)$ are smooth and continuous in each subdomain, but are discontinuous across the interface. 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^-. We also denote the source term $f(x, y)$ as $f^+(x, y)$ and $f^-(x, y)$ in Ω^+ and Ω^-, respectively. Across the interface Γ, the function values of u from different subdomains are associated by the jump conditions

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

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

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

Analytical solutions for elliptic interface problems are not readily available in the presence of irregular interfaces, while standard numerical methods may fail to produce accurate solutions because of the difficulty in enforcing the jump conditions into numerical discretization. For example, the finite difference method on Cartesian mesh will invoke a large error as the interface cuts through the grid lines near the irregular points. Besides, the process of jump conditions imposition may alter the condition number of numerical schemes, because it changes the structure of the coefficients matrix. It is crucial to appropriately address the issue from jump condition enforcement in order to design robust numerical schemes, especially when some iterative solver is utilized to solve the discretized system.

The finite element method (FEM) solution of elliptic interface problems dates back to 1970s by Babuška [6]. In continuous FEMs, complex interfaces are represented by using body-fitted unstructured grids, while the jump conditions can be satisfied in the variational formulations [7,8]. Recently, both discontinuous Galerkin (DG) [9,10] and weak Galerkin (WG) [11,12] type discontinuous FEMs have been constructed for solving elliptic interface problems. Immersed FEM [13–17] is an another popular FEM for elliptic equations, in which structured Cartesian meshes are employed so that the time-consuming mesh generation process can be avoided. To treat interfaces that cut through finite elements, the basis functions in cut-through elements can be modified to weakly satisfy jump conditions [13]. Other interface approaches, such as enforcing jump conditions via fictitious nodes [17] and Lagrange multipliers [15,16], have also been developed. Simple procedures, based on Delaunay triangulation [18] and Voronoi diagram [19] respectively, have been introduced to alter uniform grids locally to generate semi-structured interface-fitted meshes, which lead to effective virtual element [18] and finite volume [19] approaches.

The development of Cartesian grid finite difference methods for solving elliptic interface problems has received much attention in the past several decades. Peskin [20,21] proposed the immersed boundary method (IBM) to model blood flow in heart. In this method, singular forces are smeared out by discrete delta function. It proved to be a robust and efficient method, and it is typically first order in high dimension application. Fedkiw, Osher and coworkers [22,23] introduced the ghost fluid method (GFM) to treat contact discontinuities in the inviscid Euler equation. The principle lies in extending the piecewise function into the other subdomain to build a set of artificial values by the jump conditions. The extension of the first order GFM to second order has been reported in [24] recently. LeVeque and Li [25,26] have invented the immersed interface method (IIM) which introduces correction terms of jump conditions into finite difference discretization by Taylor expansion. As the first second-order accurate Cartesian grid method, the IIM has gained a great popularity in numerous applications involving elliptic equations and interfaces. By iteratively enforcing zeroth and first order jump conditions, a matched interface and boundary (MIB) method has been introduced in [27], which avoids the challenge of implementing high order jump conditions in constructing high order Cartesian grid methods. The MIB scheme is systematically carried out and can be made to arbitrarily high order in principle in the presence of straight interfaces. Orders up to 16 have been achieved numerically [27]. In treating smoothly curved interfaces, the MIB method can usually achieve the fourth order convergence [2,27]. Other effective Cartesian grid methods for elliptic interface problems include coupling interface method [28], piecewise-polynomial interface method [29], kernel free integral equation method [30], and virtual node method [31,32].

There is a great interest in developing fast interface algorithms to accelerate algebraic computations by means of fast Poisson solvers, which include geometric multigrid method with a complexity $O(N)$ and fast Fourier transform (FFT) with a complexity $O(N \log N)$, where N is the spatial degree of freedom. For FEMs, the discrete systems based on the interface-fitted mesh can be solved by using the algebraic or geometric multigrid solvers, see for example Ref. [33] with adaptive mesh refinement. For Cartesian grid methods, the formulation of the restriction and prolongation in a multigrid cycle is far away from trivial for interface problems. A major breakthrough in this direction is the multigrid IIM method developed by Adams and Li [34]. Later, multigrid solvers have also been applied in other Cartesian grid methods, including the piecewise-polynomial interface method [29] and virtual node method [31,32].

A significant achievement in the field is the augmented IIM (AIIM) [35,36] which introduces the jump in the normal derivative of the solution $[u_n]$ as an augmented variable. Thanks to the symmetric coefficient matrix derived from finite difference stencil, a fast Poisson solver associated with GMRES iterative method can be employed to solve the Schur complement system efficiently. The algebraic complexity of the AIIM primarily depends on the underlying fast solver, i.e., the FFT in case of a piecewise constant β [35] and the multigrid for piecewise variable coefficients [36]. Another effective approach for decomposing jump conditions was introduced in [37,38]. Instead of using $[u_n]$, jumps in Cartesian

derivatives, such as $[u_x]$, $[u_{xx}]$, $[u_y]$, $[u_{yy}]$ and even higher order ones, are calculated explicitly. Near the interface, jump corrected Taylor expansions have been proposed in [38], which lead to an explicit jump immerse interface method (EJIIM) for solving piecewise constant coefficient problems. For variable coefficient elliptic equations, a decomposed immersed interface method (DIIM) has been constructed using similar ideas [37]. A common feature of these three IIMs is that the Laplacian operator is approximated by the standard finite difference approximation, which results in a symmetric and diagonally dominant matrix for fast Poisson solver. The jump corrections are realized by means of auxiliary variables which can be solved either through the Schur complement method [35,36,38] or from the previous iterative step [37]. The robustness of these numerical schemes crucially depends on the numerical approximation to the jump conditions and may be affected when higher order jump conditions are included. On the other hand, the jump correction by using only zeroth and first order jump conditions in association with fast Poisson solvers has not been investigated before.

The goal of this paper is to introduce an augmented MIB method (AMIB) for solving elliptic interface problems with piecewise constant coefficients, which combines key features of AIIMs [35–38] and MIB method [2,27] in one framework. As in the regular MIB method, only zeroth and first order jump conditions will be employed to generate two layers of fictitious values on irregular nodes surrounding the interface, one inside and one outside. Unlike the MIB method, the fictitious values will not be directly used for modifying partial differential equation (PDE) discretization. Instead, jump corrected Taylor expansions introduced in the EJIIM [37,38] will be established based on the MIB fictitious values. Then, by treating the reconstructed Cartesian derivative jumps as auxiliary variables, an enlarged linear algebraic system will be formed, in a process quite similar to the AIIM [35,36]. Also, this system will be solved by the same Schur complement approach formulated in the AIIM, in which the discrete Laplacian is inverted by the FFT algorithm. The resulting AMIB scheme will be much more efficient than the classical MIB, while keeping the same second order accuracy. Like the original MIB, the AMIB is a PDE-independent approach, i.e., the interface treatment does not depend on the underlying PDE, which is a property not shared with the IIMs. The generation of the AMIB to other PDE interface problems and to higher order will be investigated elsewhere.

The rest of the paper is organized as follows. In Section 2, several key aspects of the proposed AMIB method will be presented. Section 3 will be dedicated to the implementation of our new approach in different complex interfaces and numerical convergence rate will be validated. A summary and future development will be discussed at the end of this paper.

2. Theory and algorithm

In this work, we focus on piecewise constant coefficient elliptic problem. Hence, our original PDE (1) can be rewritten as below after moving the coefficient β to the right hand side of the equation,

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

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

We restrict our discussion on an rectangular domain $[a,b] \times [c,d]$ in which a close interface Γ is embedded. The domain is partitioned into n_x and n_y equally spaced intervals in x- and y-directions respectively such that the mesh sizes are $h_x = (b-a)/n_x$ and $h_y = (d-c)/n_y$. We assume $h = h_x = h_y$ for simplicity. The grid nodes coordinates are defined as

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

In this way, Cartesian grids are generated across the domain.

Assume that the interface is defined by a level set function $\Gamma = \{(x,y), \varphi(x,y) = 0\}$, with $\varphi(x,y) > 0$ in Ω^- and $\varphi(x,y) < 0$ in Ω^+. Two functions are defined at each grid point

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

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.

We have the standard second order finite difference for second order partial derivative of x with the truncation error $O(h^2)$ as follows:

$$\frac{\partial^2 u(x_i,y_j)}{\partial x^2} \approx \frac{u(x_{i-1},y_j) - 2u(x_i,y_j) + u(x_{i+1},y_j)}{h^2} \tag{6}$$

Similar stencil could be derived for y-direction partial derivatives. While the standard difference could be used for approximation at regular points, modification is required for irregular points as finite difference is not well-defined because the function may not be smoothly continuous across the interface.

2.1. Correcting finite difference for Laplacian

In the MIB method [2,27], the Laplacian will be approximated in a tensor product manner. Similarly, it is sufficient for us to discuss the ideas of the proposed AMIB method by focusing on one direction first. Assume the interface intersects the grid line $y = y_j$ at some point α between x_i and x_{i+1}. We first establish the Taylor expansion that relates the function values at (x_i, y_j) and (x_{i+1}, y_j) across the interface. Let us drop the function dependence on y at the moment, by denoting $u = u(x)$. This reduces the derivation into a one-dimensional (1D) problem.

Assume the solution of elliptic interface problems being a piecewise smooth function, i.e., the smoothness of the piecewise function is $u \in C^{l+1}$ in each side of the interface. Here we take the situation with $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$ for demonstration. It can be similarly defined if $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$ as long as $x_i \le \alpha < x_{i+1}$. If the Cartesian jumps are known, we have the corrected Taylor expansions at two irregular points [38]

$$u(x_{i+1}) = \sum_{k=0}^{l} \frac{h^k}{k!} u^{(k)}(x_i) + \sum_{k=0}^{l} \frac{(h^+)^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{7}$$

and

$$u(x_i) = \sum_{k=0}^{l} \frac{(-h)^k}{k!} u^{(k)}(x_{i+1}) - \sum_{k=0}^{l} \frac{(h^-)^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{8}$$

where $h^- = x_i - \alpha$, $h^+ = x_{i+1} - \alpha$ with $x_i \le \alpha < x_{i+1}$, and $[u^{(m)}]_\alpha = \lim_{x\to\alpha^+} u^{(m)}(x) - \lim_{x\to\alpha^-} u^{(m)}(x)$. Based on the jump-corrected Taylor expansions (7) and (8), it is easy to derive following differences at irregular points.

Jump-corrected difference. Suppose $u \in C^4[x_i - h, \alpha) \cap C^4(\alpha, x_{i+1} + h]$ and $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$, where derivatives extend continuously up to α. The following approximations hold to $O(h^2)$ when $m = 3$ [38]:

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

$$u_{xx}(x_{i+1}) \approx \frac{u(x_{i+2}) - 2u(x_{i+1}) + u(x_i)}{h^2} + \frac{1}{h^2} \sum_{k=0}^{m} \frac{(h^-)^k}{k!} [u^{(k)}]. \tag{10}$$

In two dimension (2D), the geometry of a curved interface could be complicated. For a fast changing curvature, one grid line may cut the interface twice within a short distance. If the distance between two intersection points is less than h, the numerical resolution is too coarse and cannot sense the interface. If such a distance is larger than $2h$, these two interface points can be treated independently. Nevertheless, if such a distance is in between h and $2h$, we are facing a so-called corner node. For example, see point P_2 in Fig. 2 along y direction. The case with one grid line cutting the interface twice near a corner point is also called multiple interfaces in 1D. Multiple corrections have to be applied for the irregular point whenever multiple interfaces occur. The formula for multiple corrections is derived in x-direction for demonstration as following:

Multiple corrections. Let $x_{i-1} \le \alpha_1 < x_i \le \alpha_2 < x_{i+1}$, $h_1^- = x_{i-1} - \alpha_1$, $h_1^+ = x_i - \alpha_1$, $h_2^- = x_i - \alpha_2$, and $h_2^+ = x_{i+1} - \alpha_2$. For example, we may have the situation where $x_i \in \Omega^-$ and $x_{i-1}, x_{i+1} \in \Omega^+$. Assume $u \in C^4[x_{i-1}, \alpha) \cap C^4(\alpha_1, \alpha_2) \cap C^4(\alpha_2, x_{i+1}]$, with derivatives extending continuously up to the boundaries of the subintervals. The following second order approximation to second order derivative at x_i holds with truncation error $O(h^2)$ when $m = 3$ [38],

$$u_{xx}(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \frac{1}{h^2} \sum_{k=0}^{m} \frac{(h_1^-)^k [u^{(k)}]_{\alpha_1} - (h_2^+)^k [u^{(k)}]_{\alpha_2}}{k!}. \tag{11}$$

Remark 2.1. All the above corrected differences in the case of single interface or multiple interfaces are concerned with x-direction derivatives, which are also applicable to y-direction in the same manner. The combination of differences in the x- and y-direction enables us to approximate the Laplacian operator dimension by dimension. The corrected difference differs from standard difference by the summation of several jump quantities of derivatives, which is called a correction term. At irregular points, such correction term is of significance to address the discontinuity along the interface while correction term vanishes for regular points. On the other side, it helps maintain the symmetric and diagonally dominant properties for the standard second order discretization stencil in (9)–(11), which allow the use of the FFT algorithm.

Remark 2.2. From the corrected difference, it is easy to see that to achieve second order accuracy, we need Cartesian jump conditions up to the third order. Nevertheless, it has been proven theoretically and verified numerically in 1D [38] that it is sufficient to retain second order accuracy if we only use jump quantities up to the second order derivative with a truncation error $O(h)$ at irregular points, that is $m = 2$. Therefore, jump quantities up to second order will be adopted in the present study, and the second order global convergence can indeed be realized with first order local jump corrections.

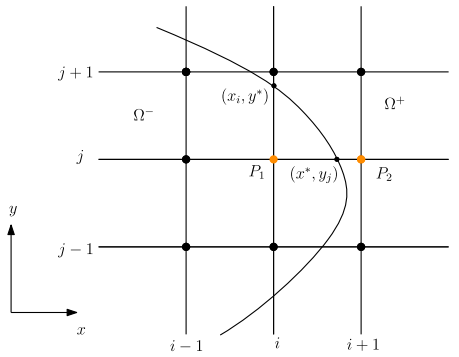


Fig. 1. Irregular points $P_1(i, j)$ and $P_2(i+1, j)$. The interface intersects $y = y_j$ at (x^*, y_j) and $x = x_i$ at (x_i, y^*).

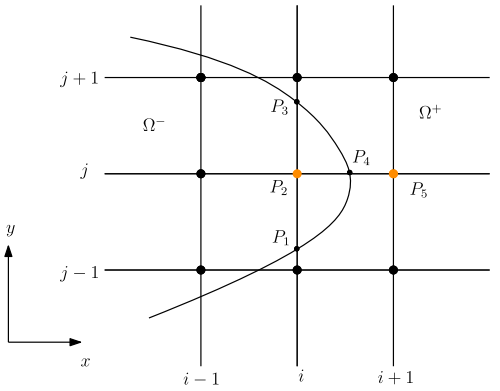


Fig. 2. Corner point $P_2(i, j)$. The interface intersects with $x = x_i$ at $P_1(x_i, y_1^*)$ and $P_3(x_i, y_2^*)$ while it intersects with $y = y_j$ at $P_4(x^*, y_j)$.

Bearing the above x-oriented corrected difference (9)–(11) in mind, we generalize the idea into 2D. Let us take a general case for illustration, in which we come across an irregular point with two of its adjacent points on the other side of the interface. Assume point $P_1(x_i, y_j)$ with irregularity stemming from the intersection of interface Γ with grid line $y = y_j$ on (x^*, y_j) and grid line $x = x_i$ on (x_i, y^*). See Fig. 1. Let $h_x^- = x_i - x^*$, $h_x^+ = x_{i+1} - x^*$, $h_y^- = y_j - y^*$ and $h_y^+ = y_{j+1} - y^*$. Corrected differences are needed to approximate Laplacian with (9) applied in both x- and y-direction [38]. Thus the approximation can be formulated as below with a local truncation error $O(h)$:

$$\triangle u(x_i, y_j) \approx \frac{u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j)}{h^2} - \frac{1}{h^2}\sum_{k=0}^{2} \frac{(h_x^+)^k}{k!}[\frac{\partial^k u}{\partial x^k}]|_{(x^*, y_j)} - \frac{1}{h^2}\sum_{k=0}^{2} \frac{(h_y^+)^k}{k!}[\frac{\partial^k u}{\partial y^k}]|_{(x_i, y^*)},$$

where Cartesian derivative jumps up to the second order ones are required. The correction for irregular point $P_2(x_{i+1}, y_j)$ is simpler in comparison with point $P_1(x_i, y_j)$ since it only involves one adjacent point on the other side of the interface. Formula (10) is employed for the x-partial derivative approximation in Laplacian operator while standard second finite difference is utilized in y-direction.

For the case of a corner point P_2 in Fig. 2, multiple corrected difference (11) is needed in y-direction while one correction (9) is applied in x-direction. In this case, the interface has three intersection points with x- and y-grid lines, i.e. $P_1(x_i, y_1^*)$, $P_3(x_i, y_2^*)$, and $P_4(x^*, y_j)$. Several signed distances are defined as $h_x^- = x_i - x^*$, $h_x^+ = x_{i+1} - x^*$, $h_{y_1}^- = y_1^* - y_{j-1}$, $h_{y_1}^+ = y_j - y_1^*$, $h_{y_2}^- = y_j - y_2^*$, and $h_{y_2}^+ = y_{j+1} - y_2^*$. The corrected difference to approximate Laplacian at P_2 is derived as: [38]

$$\triangle u(x_i, y_j) \approx \frac{u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j)}{h^2} - \frac{1}{h^2}\sum_{k=0}^{2} \frac{(h_x^+)^k}{k!}[\frac{\partial^k u}{\partial x^k}]|_{(x^*, y_j)} + \frac{1}{h^2}\sum_{k=0}^{2} \frac{(h_{y_1}^-)^k}{k!}[\frac{\partial^k u}{\partial y^k}]|_{(x_i, y_1^*)} - \frac{1}{h^2}\sum_{k=0}^{2} \frac{(h_{y_2}^+)^k}{k!}[\frac{\partial^k u}{\partial y^k}]|_{(x_i, y_2^*)}$$

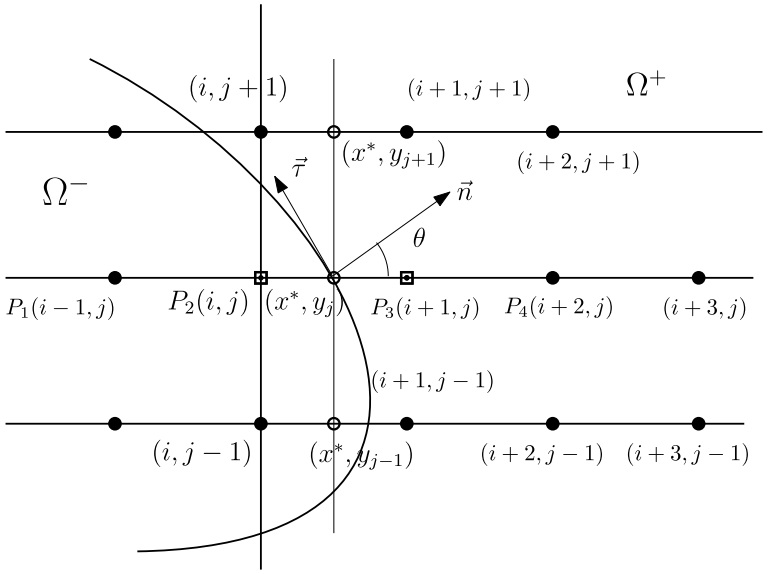


Fig. 3. The interface intersect $y = y_j$ at (x^*, y_j). At two irregular points $P_2(i, j)$, $P_3(i, j+1)$, a pair of fictitious values (empty squares) can be constructed. θ is the angle between positive x-direction and the normal vector $\vec{n}$.

Three special examples are given above. Formulas for other cases could be easily generalized depending on the location of irregular points to the interface using corrected difference (9) and (10).

2.2. Reconstructing the Cartesian derivative jumps

So far we have established the corrected differences with Cartesian derivative jumps undetermined. These jumps are solution dependent and their reconstruction is non-trivial so that it needs appropriate numerical approximation as one part of the solution. That is where fictitious values generated by the MIB method come in.

To maintain a first order convergence locally for the irregular points, an interpolation polynomial with third order accuracy is expected to approximate the interracial jump value. This guarantees the truncation error being $O(h)$ for second order derivatives. Such approximations will be realized in two stages in the proposed AMIB method: First, a pair of fictitious values are constructed along the Cartesian grid line as smooth extension for each piecewise function in the other subdomain. Second, the fictitious values are utilized in two interpolations from each subdomain to produce jump quantities of up to second order derivatives.

2.2.1. Fictitious values formulation

The MIB method [2,27] is employed to generate fictitious values in the vicinity of the interface with the aid of known jump conditions (3) and (4). One more jump condition could be analytically derived by differentiating Eq. (3) along the tangential direction τ of the interface as below,

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

For an interface point, the normal vector of the interface is $\vec{n} = (\cos\theta, \sin\theta)$ and the tangential direction is $\vec{\tau} = (-\sin\theta, \cos\theta)$ with $0 \le \theta \le 2\pi$ being the angle between positive x-direction and the vector $\vec{n}$ as shown in Fig. 3. The Cartesian forms of three known interface conditions can be given as [27],

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

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

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

for a point (x^*, y^*) on the interface.

Two equivalent approaches were proposed in [27] for generating two fictitious values in x-direction, by either canceling u_y^- or u_y^+ in (13), (14), and (15). Likewise, fictitious value simulation in y-direction can be completed in a similar fashion by canceling either u_x^- or u_x^+.

For simplicity, we only illustrate the case with the cancellation of u_x^- and u_y^-. This yields the following formulations for x- and y-direction fictitious values respectively:

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

$$[\![\beta u_n]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ u_x^+ - C_x^- u_x^- + C_y^+ u_y^+, \tag{17}$$

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

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

$$[\![\beta u_n]\!] + \beta^- \cot\theta [\![u_\tau]\!] = D_x^+ u_x^+ + D_y^+ u_y^+ - D_y^- u_y^-, \tag{19}$$

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

To illustrate how to obtain a pair of fictitious values along x-direction, formulation (16) and (17) are discretized by the finite difference, yielding

$$[\![u]\!] = w_{0,i}^+ \hat{u}_{i,j} + w_{0,i+1}^+ u_{i+1,j} + w_{0,i+2}^+ u_{i+2,j} - (w_{0,j-1}^- u_{i-1,j} + w_{0,i}^- u_{i,j} + w_{0,i+1}^- \hat{u}_{i+1,j}) \tag{20}$$

$$\begin{aligned} [\![\beta u_n]\!] - \beta^- \tan\theta [\![u_\tau]\!] =& C_x^+ (w_{1,i}^+ \hat{u}_{i,j} + w_{1,i+1}^+ u_{i+1,j} + w_{1,i+2}^+ u_{i+2,j}) \\ & - C_x^- (w_{1,i-1}^- u_{i-1,j} + w_{1,i}^- u_{i,j} + w_{1,i+1}^- \hat{u}_{i+1,j}) \\ & + C_y^+ (p_{1,j}^+ u_{\Gamma,j} + p_{1,j+1}^+ u_{\Gamma,j+1} + p_{1,j+2}^+ u_{\Gamma,j+2}), \end{aligned} \tag{21}$$

where $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ stand for the unknown fictitious values at point (x_i, y_j) and (x_{i+1}, y_j) on each side of the interface. Here w and p denote the finite difference weights [39] for approximation along x- and y-direction. It could be automatically obtained through a call to Fortran subroutine [39]. The superscripts $-$ and $+$ in w and p signify the Ω^- and Ω^+ domain separated by the interface while i denotes the order of derivative and j represents the location along the grid line. Note that three auxiliary function values $u_{\Gamma,j-1}$, $u_{\Gamma,j}$, $u_{\Gamma,j+1}$ at points (x^*, y_{j-1}), (x^*, y_j), (x^*, y_{j+1}) in the y-direction are deployed to approximate u_y^+, see Fig. 3. Additionally, each of these three auxiliary values are interpolated or extrapolated by three function values in Ω^+. For example, in Fig. 3, $u_{\Gamma,j+1}$ is interpolated by function values $u_{i,j+1}$, $u_{i+1,j+1}$ and $u_{i+2,j+1}$ while both $u_{\Gamma,j}$ and $u_{\Gamma,j-1}$ are extrapolated by three function values from their right side in the Ω^+ domain.

With $u_{\Gamma,j}$, $u_{\Gamma,j+1}$ and $u_{\Gamma,j+2}$ appropriately approximated, (20) and (21) give 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^*). A general form of $\hat{u}_{i,j}$ is given as

$$\begin{aligned} \hat{u}_{i,j} &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 [\![u]\!] + W_1 [\![u_\tau]\!] + W_2 [\![\beta u_n]\!] \\ &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 \phi(x^*, y^*) + W_1 \rho(x^*, y^*) + W_2 \psi(x^*, y^*) \end{aligned} \tag{22}$$

where $\mathbb{S}_{i,j}$ represents a set of surrounding nodes involved in (20) and (21). Similarly, $\hat{u}_{i+1,j}$ could be represented by the functions values on the set $\mathbb{S}_{i+1,j}$ and known jump values with different weights $W_{I,J}$. Pairs of fictitious values in the y-direction could be generated with Eqs. (18) and (19) following a similar procedure as above.

The MIB fictitious value representation typically involves points other than immediate neighbors of (x_i, y_j) as shown in (22). The underlying extrapolation is designed so that a second order accurate approximation is maintained in estimating these fictitious values. Moreover, no stability issue is experienced for the present elliptic interface problems. For time-dependent problems, a different extrapolation strategy has been studied in the MIB literature [40,41]. Non-orthogonal local coordinates were introduced to minimize the distance between extrapolation point and stencil center [40], and to simplify extrapolation in three-dimensions [41].

2.2.2. Approximation to derivative jumps

We next consider how to reconstruct Cartesian jumps based on fictitious values. This procedure essentially bridges the MIB method with the AIIM method, and has never been studied before in the finite difference literature.

In order to provide jump quantities in the corrected difference, we need to build two Lagrange polynomials of degree two on each subdomain to give limiting derivatives from left (below) and right (above), which will result in the approximated jumps for up to second order derivative. Take a jump approximation along x-direction for an illustration here. As shown in Fig. 3, we combine real values $u_{i-1,j}$ and $u_{i,j}$ at points (x_{i-1}, y_j) and (x_i, y_j) together with fictitious value $\hat{u}_{i+1,j}$ at $u_{i+1,j}$ for the Lagrange polynomial on the left-side subdomain. Meanwhile, with aid of fictitious value $\hat{u}_{i,j}$ at point (x_i, y_j) combined with real value $u_{i+1,j}$ and $u_{i+2,j}$ located at points (x_{i+1}, y_j) and (x_{i+2}, y_j), we obtain another Lagrange polynomial for the right-side subdomain.

By taking derivatives of first and second orders on both Lagrange polynomials, we can approximate the jump values at intersecting point (x^*, y_j) with regard to x-partial derivatives as below

$$\begin{aligned}[\frac{\partial u}{\partial x}] &= \lim_{x\to(x^*)^+}\frac{\partial u}{\partial x}(x,y_j) - \lim_{x\to(x^*)^-}\frac{\partial u}{\partial x}(x,y_j)\\ &= w^+_{1,1}\hat{u}_{i,j} + w^+_{1,2}u_{i+1,j} + w^+_{1,3}u_{i+2,j} - w^-_{1,1}u_{i-1,j} - w^-_{1,2}u_{i,j} - w^-_{1,3}\hat{u}_{i+1,j} + O(h^2),\\ [\frac{\partial^2 u}{\partial x^2}] &= \lim_{x\to(x^*)^+}\frac{\partial^2 u}{\partial x^2}(x,y_j) - \lim_{x\to(x^*)^-}\frac{\partial^2 u}{\partial x^2}(x,y_j)\\ &= w^+_{2,1}\hat{u}_{i,j} + w^+_{2,2}u_{i+1,j} + w^+_{2,3}u_{i+2,j} - w^-_{2,1}u_{i-1,j} - w^-_{2,2}u_{i,j} - w^-_{2,3}\hat{u}_{i+1,j} + O(h),\end{aligned}$$

which are equivalent to

$$w^-_{1,1}u_{i-1,j} + w^-_{1,2}u_{i,j} - w^+_{1,2}u_{i+1,j} - w^+_{1,3}u_{i+2,j} - w^+_{1,1}\hat{u}_{i,j} + w^-_{1,3}\hat{u}_{i+1,j} + [\frac{\partial u}{\partial x}] = 0, \tag{23}$$

$$w^-_{2,1}u_{i-1,j} + w^-_{2,2}u_{i,j} - w^+_{2,2}u_{i+1,j} - w^+_{2,3}u_{i+2,j} - w^+_{2,1}\hat{u}_{i,j} + w^-_{2,3}\hat{u}_{i+1,j} + [\frac{\partial^2 u}{\partial x^2}] = 0, \tag{24}$$

with local truncation error $O(h^2)$ and $O(h)$, respectively.

In addition to the two approximations to derivatives by (23) and (24), a third equation of the function jump in the correction term is needed. Instead of a numerical approximation, the function jump across the interface from left to right is explicitly obtained by the analytical jump condition (3).

$$[u] = \lim_{x\to(x^*)^+} u(x,y_j) - \lim_{x\to(x^*)^-} u(x,y_j) = c[\![u]\!] = c\phi(x^*,y_j), \tag{25}$$

where $\lim_{x\to(x^*)^+}$ or $\lim_{x\to(x^*)^-}$ stands for the right or left limit, and constant c is either 1 or -1 depending on the desired values from the given interface conditions. The analytical jump condition instead of numerical approximation guarantees the exact jump quantity.

Note that $w^+_{i,j}$ and $w^-_{i,j}$ denote the weights on each function value after taking certain derivative of Lagrange polynomial at the intersecting point (x^*, y_j). The subscript i stands for the order of derivative while subscript j signifies the order of function value from the left. Especially, the signs $-$ and $+$ in w and u here represent the value from the left and right subdomain of the interface respectively, rather than the sign definition in the Ω subdomain. This is how the constant c is ensuring the correct sign of function jump $[u]$ in the correction term regulated by given jump condition (3). This clarification applies for the below and above subdomain in the study of y-direction derivative jumps.

In the following, we denote a vector $\tilde{Q} = \left([u], [\frac{\partial u}{\partial x}], [\frac{\partial^2 u}{\partial x^2}]\right)^T$ as the jump quantities in correction term for one irregular point. Eqs. (23)–(25) can be rewritten into a matrix–vector form for the jump quantities formulation

$$\tilde{C}\tilde{U} + I\tilde{Q} = \tilde{\Phi} \tag{26}$$

where $\tilde{U}$ includes all function values contained in $\mathbb{S}_{i,j}$ and $\mathbb{S}_{i+1,j}$, $\tilde{C}$ is a coefficient matrix containing the corresponding finite difference weights for $\tilde{U}$. And $\tilde{\Phi}$ is composed of terms after moving the known interface quantities $\phi(x^*, y^*)$, $\psi(x^*, y^*)$ and $\rho(x^*, y^*)$ in representation of fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ to the right-hand side. For $[u]$, we simply take corresponding row of $\tilde{C}$ as zero and corresponding entry of $\tilde{\Phi}$ as $c\phi$ in (26).

2.3. Augmented system

2.3.1. Formulation of the augmented system

Let $U_{i,j}$ indicate the discrete solution while $u(x_i, y_j)$ is continuous solution at (x_i, y_j). Based on the corrected difference analysis, the problem (5) is discretized in all interior nodes as

$$L_hU_{i,j} + C_{i,j} = -\frac{f_{i,j}}{\beta_{i,j}},\ 1 \le i \le n_x - 1,\ 1 \le j \le n_y - 1, \tag{27}$$

where $C_{i,j}$ is the correction term, and $L_hU_{i,j}$ is the standard five point central difference scheme

$$L_hU_{i,j} := \frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2}.$$

The degree of freedom of this system is $N = (n_x - 1)(n_y - 1)$.

Note that correction term only exists at irregular points but vanishes at regular points. Via Eq. (27), a relation between numerical solution and jump values of x- or y-partial derivatives is obtained as

$$AU + BQ = F \tag{28}$$

where A is a N-by-N symmetric and diagonally dominant matrix, and the N-vector F is modified from the right-hand side of (27) after homogenizing the boundary conditions. Variable U stands for the numerical solution for all grid nodes.

Similar to $\tilde{Q}$ introduced in (26) for local approximation to a single set of correction jump quantities, here Q is a vector of all auxiliary variables $[u]_i$, $[\frac{\partial u}{\partial x}]_i$, $[\frac{\partial^2 u}{\partial x^2}]_i$, $i = 1, 2, \ldots$, and $[u]_j$, $[\frac{\partial u}{\partial y}]_j$, $[\frac{\partial^2 u}{\partial y^2}]_j$, $j = 1, 2, \ldots$ at all intersection points between the interface and mesh lines. The total number of auxiliary variables, denoted as M here, is the triple of that of all intersection points, while the latter is usually proportional to n_x or n_y. In other words, M is usually one-dimensional smaller than $N = (n_x - 1)(n_y - 1)$. The matrix B with dimension N-by-M contains coefficients from correction terms. In particular, matrix B is a sparse matrix, since the correction terms only have impact on the irregular points which account for a small portion in the whole computation grid points.

In matrix–vector form, the approximation to all the jump quantities in correction terms could be represented as

$$CU + IQ = \Phi \tag{29}$$

where the notations in (29) share the same meaning as their local counterparts in Eq. (26). Here the matrix C with dimension M-by-N is also a sparse matrix, and I is a M-by-M identity matrix. An augmented equation system is therefore obtained by combining (28) and (29),

$$KW = R, \tag{30}$$

where

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

2.3.2. A Schur complement procedure

We are concerned with solving the linear system of equations efficiently. The symmetric coefficient matrix A enables us to take advantage of the fast Fourier transform (FFT) algorithm in the implementation. Eliminating U in (30) gives a Schur complement system for Q,

$$(I - CA^{-1}B)Q = \Phi - CA^{-1}F, \tag{31}$$

which has a much smaller dimension. To solve for Q in equation system (31), biconjugate gradient method could be utilized. Once Q is determined, U could be solved by one more FFT on

$$AU = (F - BQ). \tag{32}$$

Solving the Schur complement system (31) will consume the majority of computational cost. Thus it needs a careful treatment. It can be explained in the following steps.

1. With F determined initially, we can calculate $\hat{F} = \Phi - CA^{-1}F$ easily after performing FFT on F, together with simple multiplication and subtraction on matrix and vector.
2. To start the biconjugate gradient iteration in solving for Q from (31), we give an initial guess $Q = (0, 0, \ldots, 0)^T$. The terminating tolerance in biconjugate gradient iteration can be set based on the expected second convergence requirement of solutions.
3. The left hand multiplication of matrix by vector $(I - CA^{-1}B)Q$ is achieved by $IQ - CA^{-1}BQ$. To be more clear, the multiplication by vector Q is first finished on matrix B and I to avoid explicitly forming matrices B and C in multiplication of $I - CA^{-1}B$ in the process of programming. Otherwise, it will take a great deal of computer storage. Once BQ is obtained, FFT and simple multiplication on vector is utilized again for $CA^{-1}(BQ)$. The transpose multiplication of $(I - CA^{-1}B)^T Q$ is done basically by following the same strategy on $IQ - B^T A^{-1} C^T Q$, due to its equivalence to original transpose multiplication.
4. At last, Q will be updated to start a new iteration until the tolerance is reached.

The complexity of the AMIB method crucially depends on the iteration number consumed in solving (31), whose dimension is $M \times M$. In general, such an iteration number could be linearly proportional to M. Nevertheless, as to be shown in our numerical experiments, this number does not depend or weakly depends on M in the AMIB method. Similar finding has been observed in the AIIM [35,36]. Consequently, the complexity of the AMIB is primarily determined by the cost of matrix–vector product in (31) or the cost of FFT for inverting A. Recall that $N = (n_x - 1)(n_y - 1)$. Asymptotically, let us denote $n = O(n_x) = O(n_y)$, $M = O(n)$, and $N = O(n^2)$. Under the assumption that the iteration number is almost independent of n, the complexity of the AMIB algorithm will be $O(n^2 \log(n^2)) = O(n^2 \log n)$. Such a speed will be much faster than the regular MIB method, whose complexity usually scales as $O(n^3)$.

The total number of auxiliary variables M is the triple of that of all intersection points in the proposed AMIB method. This could be reduced by one-third if the known jumps of $[u]$ are moved to the right hand side. We have tested this approach and found that this does not change the computational efficiency. In particular, the iteration numbers are the same. Since the current implementation is simpler, we focus only on this approach in numerical studies. We also note that a direct method has been proposed recently for the augmented IIM [42]. The acceleration of the AMIB method by means of different auxiliary variables or direct algebraic solutions will be explored in near future.

3. Numerical experiments

In this section, we examine the performance of the proposed augmented MIB (AMIB) method for two dimensional elliptic problem with various interfaces. A square domain with equally spaced nodes is used such that there are $n_x - 1$ and $n_y - 1$ interior grids in x and y direction, respectively. By taking $n = n_x = n_y$, we define $N = (n-1)^2$. Also the length of the domain is chosen as a non-integer so that the intersection points between the interface and grid lines locate at random places. The numerical accuracy and convergence will be tested by comparing the numerical solutions with exact solutions under the standard maximum norm and L_2 norm defined as

$$L_\infty = \max_{1\le i,j\le n-1} |u(x_i, y_j) - u_h(x_i, y_j)|$$

and

$$L_2 = \sqrt{\frac{1}{N}\sum_{i,j} |u(x_i, y_j) - u_h(x_i, y_j)|^2}$$

where $u(x_i, y_j)$, $u_h(x_i, y_j)$ are numerical and analytical solution, respectively. And the convergence rate of the scheme will be examined by the formula

$$\text{order} = | \frac{\log(\|E_n\|/\|E_{2n}\|)}{\log(2)} |,$$

where $\|E_n\|$ is the error under either of the above two defined norms on n by n mesh.

Additionally, relative error is defined here,

$$L_R = \frac{\max_\Gamma |v(x,y) - \bar{v}(x,y)|}{\max_\Gamma |v(x,y)|},$$

where function $v(x, y)$ is the real value at the intersection of grid line with the interface Γ, and $\bar{v}(x, y)$ is the approximate value at the same point. This norm will be used in measuring the accuracy of the approximate jump quantities.

All the experiments were carried out on personal laptop Lenovo Thinkpad L440 with 8.00 RAM and Intel Core i5-4200M @ 2.50 GHz.

3.1. Numerical accuracy

In this section, several comparisons on numerical accuracy between MIB method and augmented MIB method are studied through Examples 1 to 5 The tolerance for biconjugate gradient solvers in both AMIB and MIB are set to be $1.0e^{-10}$. Iteration number in solving the Schur complement system of the AMIB scheme is reported in all examples.

Example 1. A two dimensional Poisson equation

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

defined in a square $[-1.00001, 1.00001] \times [-1.00001, 1.00001]$ with a circular interface defined by $r^2 := x^2 + y^2 = \frac{1}{4}$. The exact solution to this problem is

$$u(x,y) = \begin{cases} x^2 + y^2, & r \le 0.5, \\ \frac{1}{4}(1 - \frac{1}{8b} - \frac{1}{b}) + (\frac{r^4}{4} + r^2)/b, & \text{otherwise}, \end{cases} \tag{33}$$

with the diffusion coefficient

$$\beta = \begin{cases} 2, & r \le 0.5, \\ b, & \text{otherwise}. \end{cases}$$

Here we call β as β^+ when it is outside of Γ, and β^- when it is inside of the interface Γ. It follows the same principle in below notations. The source term $q(x, y)$ is related to the above designated solution,

$$q(x,y) = \begin{cases} 8.0, & r \le 0.5, \\ 8(x^2 + y^2) + 4.0, & \text{otherwise}. \end{cases}$$

Selected parameter $b = 10$ leads to jump condition $[\![u]\!] = 0$ and $[\![\beta u_n]\!] = -0.75$ on the interface. A computed solution is plotted on interior nodes on a mesh with $n = 64$ in Fig. 4. The numerical errors from different meshes are listed in Table 1, where we can observe that this example shows convergence rate of second order of our approach.

Example 2. We want to solve Laplace equation $u_{xx}+u_{yy} = 0$, defined in square $[-1.00001, 1.00001]\times[-1.00001, 1.00001]$ with a circular interface defined by $r^2 := x^2 + y^2 = \frac{1}{4}$. The constructed analytical solution are prescribed as

$$u(x,y) = \begin{cases} e^x \cos(y), & r \le 0.5, \\ 0, & \text{otherwise}. \end{cases}$$

Table 1
Example 1 — Numerical error analysis. Here $\beta^+ = 10, \beta^- = 2$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		Iter no.
	Error	Order	Error	Order	
[32, 32]	1.053E−4	–	4.526E−5	–	14
[64, 64]	2.024E−5	2.38	8.334E−6	2.44	18
[128, 128]	5.909E−6	1.78	1.506E−6	2.47	20
[256, 256]	1.041E−6	2.50	3.529E−7	2.09	27
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[32, 32]	2.299E−3	–	4.518E−4	–	
[64, 64]	4.345E−4	2.40	8.788E−5	2.36	
[128, 128]	1.569E−4	1.47	2.435E−5	1.85	
[256, 256]	3.416E−5	2.20	5.300E−6	2.20	

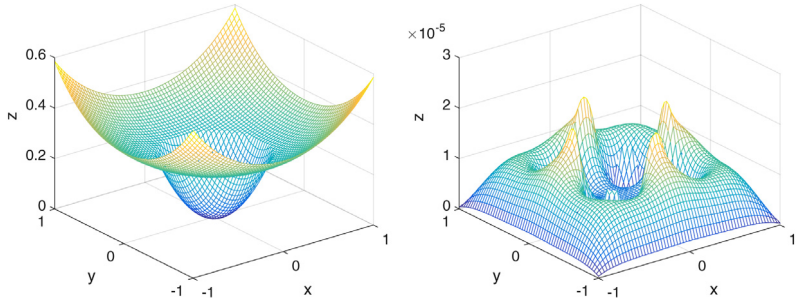


Fig. 4. The computed solution (left), and error (right) in Example 1 on a mesh with $n = 64$.

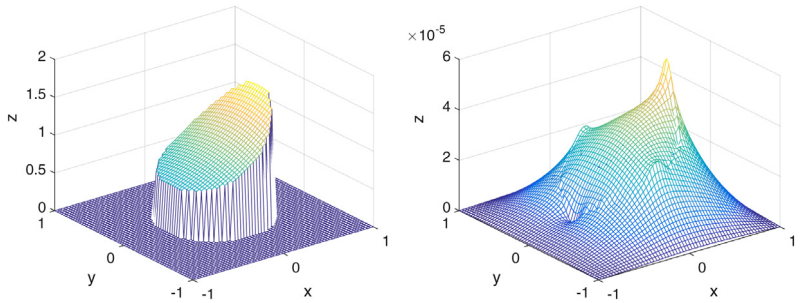


Fig. 5. The computed solution (left), and error (right) in Example 2 on a mesh with $n = 64$. Here β is equal to 1 throughout the domain.

A computed solution is plotted with $n = 64$ in Fig. 5. Without a jump in the PDE coefficient, this example is actually an irregular domain problem. The second order convergence of the AMIB method is again validated by the tests shown in Table 2.

Example 3. In this example, we consider Poisson equation with an elliptic interface Γ, which is defined as

$$(\frac{x}{20/27})^2 + (\frac{y}{10/27})^2 = 1$$

with exact solution

$$u(x, y) = \begin{cases} e^x \cos(y) & \text{inside } \Gamma, \\ 5e^{-x^2 - \frac{y^2}{2}} & \text{otherwise,} \end{cases}$$

on a square domain $[-1.00001, 1.00001] \times [-1.00001, 1.00001]$. In this example, two cases of β will be studied. In case A, β^- is equal to 10 and for case B, β^- is equal to 1000. β^+ is equal to 1 in both cases. It can be seen that both AMIB and MIB methods produce larger errors when β^- is increased from 10 to 1000. Furthermore, the iteration number of

Table 2
Example 2 — Numerical error analysis.

$[N_x, N_y]$	AMIB				
	L_∞		L_2		Iter no.
	Error	Order	Error	Order	
[64, 64]	5.799E−5	–	1.758e−5	–	13
[128, 128]	1.205E−5	2.27	3.834E−6	2.20	14
[256, 256]	2.533E−6	2.25	8.709E−7	2.14	14
[512, 512]	6.315E−7	2.00	2.364E−7	1.88	15
$[N_x, N_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	5.770E−5	–	1.700E−5	–	
[128, 128]	1.203E−5	2.26	3.767E−6	2.17	
[256, 256]	2.532E−6	2.25	8.647E−7	2.12	
[512, 512]	6.312E−7	2.00	2.356E−7	1.88	

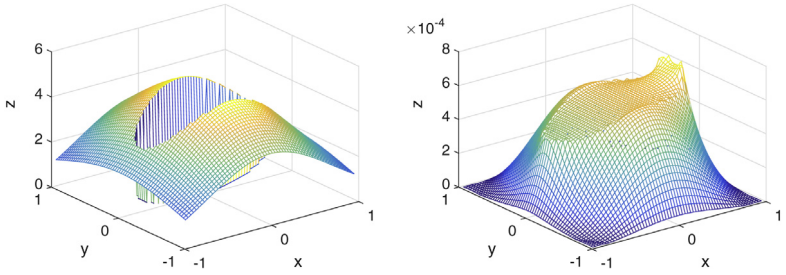


Fig. 6. The computed solution (left), and error (right) in example 3A on a mesh with $n = 64$. Here $\beta^+ = 1$, $\beta^- = 10$.

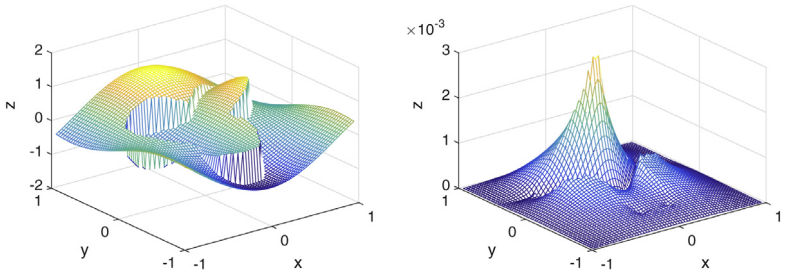


Fig. 7. The computed solution (left), and error (right) in Example 4 (case A) with $\beta^+ = 1000$ and $\beta^- = 1$ on a mesh with $n = 64$.

the AMIB computation also becomes larger. However, the AMIB method still performs robustly in dealing with interface problems of high β ratio contrast. In fact, despite of a lower precision, the AMIB method still achieves second order of convergence. Moreover, the increment slope of the iteration number with respect to the mesh size n is still very small. Consequently, the AMIB method is still much more efficient than the MIB method. A computed solution of Example 3A is plotted in Fig. 6.

Example 4. Poisson equation $\beta \triangle u = f(x, y)$ is studied here with an interface Γ

$$r = 0.5(1 + 0.5\sin(3\theta)),$$

and two exact solutions of Poisson equation designated by

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

on a square domain $[-\frac{\pi}{3.2}, \frac{\pi}{3.2}] \times [-\frac{\pi}{3.2}, \frac{\pi}{3.2}]$. The parameter k is set to be 2. The source terms associated with the two solutions are

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

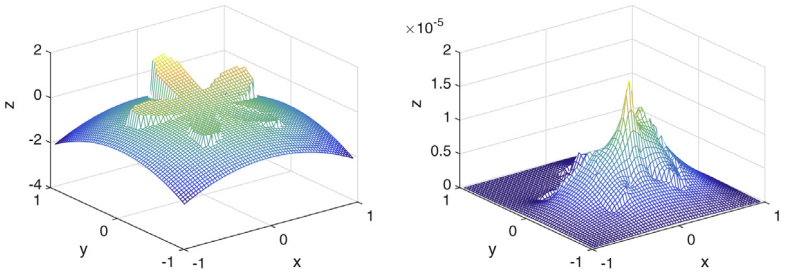


Fig. 8. The computed solution (left), and error (right) in Example 5 on a mesh with $n = 64$.

Table 3
Example 3A — Numerical error analysis. Here $\beta^- = 10$, $\beta^+ = 1$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		Iter no.
	Error	Order	Error	Order	
[64, 64]	7.019E−4	–	3.644E−4	–	30
[128, 128]	2.184E−4	1.68	9.981E−5	1.87	34
[256, 256]	4.347E−5	2.33	2.162E−5	2.20	41
[512, 512]	1.002E−5	2.12	5.036E−6	2.10	43
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	7.882E−4	–	4.292E−4	–	
[128, 128]	2.473E−4	1.67	1.178E−4	1.87	
[256, 256]	4.960E−5	2.32	2.576E−5	2.19	
[512, 512]	1.187E−5	2.06	6.284E−6	2.04	

Table 4
Example 3B — Numerical error analysis. Here $\beta^- = 1000$, $\beta^+ = 1$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		Iter no.
	Error	Order	Error	Order	
[64, 64]	3.025E−2	–	1.883E−2	–	89
[128, 128]	9.261E−3	1.71	5.714E−3	1.72	72
[256, 256]	1.463E−3	2.66	9.001E−4	2.67	114
[512, 512]	4.206E−4	1.80	2.583E−4	1.80	111
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	3.704E−2	–	2.243E−2	–	
[128, 128]	1.215E−2	1.61	7.404E−3	1.60	
[256, 256]	2.175E−3	2.48	1.332E−3	2.47	
[512, 512]	6.416E−4	1.76	3.938E−4	1.76	

Two sets of parameter values for β^+ and β^- are chosen. Tables 5 and 6 show the numerical results with $\beta^+ = 1000$, $\beta^- = 1$ (Case A) and $\beta^+ = 1$, $\beta^- = 1000$ (Case B), respectively. From the two tables, the AMIB method again yields a second order of accuracy. A comparison on the two tests demonstrates that our methods are robust enough in dealing with changing jump ratio $\rho = \beta^+/\beta^-$, with both $\rho \gg 1$ and $\rho \ll 1$ (see Fig. 7).

Example 5. Our approach is also tested on a complicated geometric interface problem. The interface Γ is defined as

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

with two exact solutions of Poisson equation designated by

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

Table 5
Example 4A — Numerical error analysis. Here $\beta^+ = 1000$, $\beta^- = 1$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		Iter no.
	Error	Order	Error	Order	
[64, 64]	2.213E−3	–	4.300E−4	–	52
[128, 128]	5.321E−4	2.03	1.025E−4	2.07	100
[256, 256]	1.304E−4	2.12	2.470E−5	2.05	120
[512, 512]	3.285E−5	1.99	6.300E−6	1.97	123
[1024, 1024]	8.943E−6	1.88	1.712E−6	1.88	172
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	2.092E−3	–	4.004E−4	–	
[128, 128]	5.330E−4	1.97	9.906E−5	2.02	
[256, 256]	1.292E−4	2.04	2.451E−5	2.01	
[512, 512]	3.229E−5	2.00	6.191E−6	1.99	
[1024, 1024]	8.704E−6	1.89	1.671E−6	1.90	

Table 6
Example 4B — Numerical error analysis. Here $\beta^+ = 1$, $\beta^- = 1000$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		Iter no.
	Error	Order	Error	Order	
[64, 64]	3.080E−2	–	1.987E−2	–	47
[128, 128]	5.463E−3	2.49	3.484E−3	2.51	65
[256, 256]	3.370E−3	0.70	2.177E−3	0.68	74
[512, 512]	1.029E−3	1.71	6.647E−4	1.71	79
[1024, 1024]	2.914E−4	1.82	1.882E−4	1.82	112
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	3.427E−2	–	2.147E−2	–	
[128, 128]	5.000E−3	2.78	3.128E−3	2.78	
[256, 256]	3.336E−3	0.58	2.138E−3	0.55	
[512, 512]	1.003E−3	1.73	6.449E−4	1.73	
[1024, 1024]	2.856E−4	1.81	1.840E−4	1.81	

on a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The diffusion coefficient β is defined as $\beta^+ = 100$ and $\beta^- = 1$. The AMIB errors are slightly larger than those of the MIB, while a second order of convergence is still achieved for the AMIB scheme (see Fig. 8).

3.2. Cartesian jump reconstruction

In this part, we want to test the accuracy of the AMIB approximation of the jump quantities in the correction terms. Only the approximated jump quantities of first and second Cartesian derivatives will be tested here. We note that the zeroth order jump quantity is given analytically in our discretization. Here L_2 and L_R norms will be used in the measurement. With the same parameters in Examples 1 and 3, we test the accuracy of the jump quantities for these two examples. Table 8 shows that the convergence rate for the first and second order jumps follows second and first order, respectively. Therefore, we conclude that the correction term is at least first order accurate as a whole.

Remark 3.1. Through the above several examples, we can see that the our method is a second order accurate scheme in spite of the local first order accuracy around the interface. Compared to the MIB method, the AMIB gives us quite accurate solutions as we can see in Tables 1–7, there is no much difference in the numerical errors from these two methods. Our jump quantities approximation from fictitious points proves to be reliable in terms of accuracy, and is verified to be at least first order accurate from Table 8.

3.3. Computational efficiency

Here we carry out a few comparisons between our augmented MIB method (AMIB) with regular MIB method (MIB) in terms of the computational efficiency. With the same parameter set in the proceeding five examples, we conclude that our

Table 7
Example 5 — Numerical error analysis. Here $\beta^+ = 100$, $\beta^- = 1$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		Iter no.
	Error	Order	Error	Order	
[64, 64]	1.806E−5	–	3.277E−6	–	59
[128, 128]	2.208E−6	3.03	4.311E−7	2.93	75
[256, 256]	6.691E−7	1.72	1.250E−7	1.79	93
[512, 512]	1.717E−7	1.96	3.181E−8	1.97	116
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	3.406E−6	–	3.804E−7	–	
[128, 128]	4.722E−7	2.85	6.357E−8	2.58	
[256, 256]	8.000E−8	2.56	1.591E−8	2.00	
[512, 512]	3.746E−8	1.09	5.400E−9	1.56	

Table 8
Reconstructed Cartesian derivative jumps for Examples 1 and 3A.

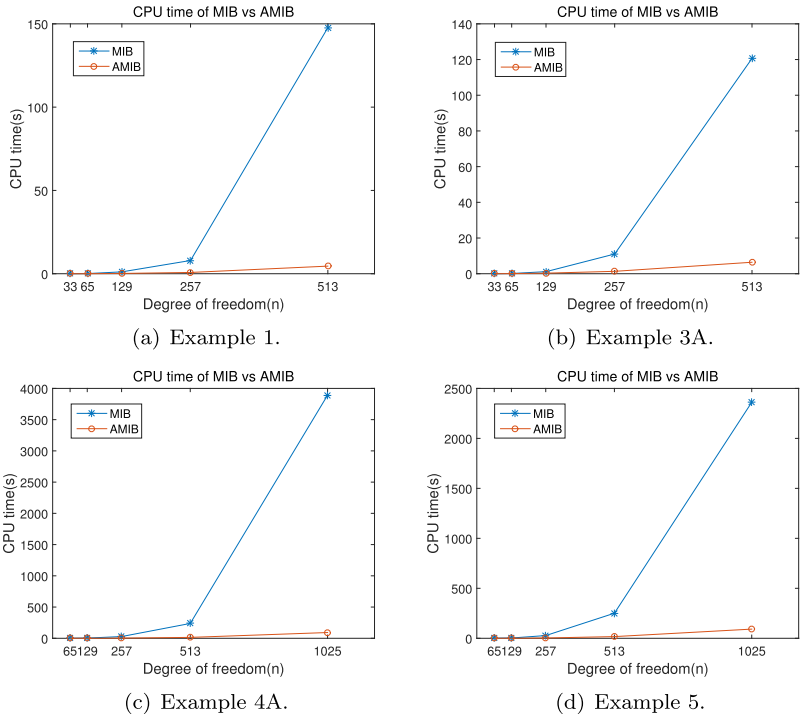
Example 1								
$[n_x, n_y]$	First order derivative jumps				Second order derivative jumps			
	L_R		L_2		L_R		L_2	
	Error	Order	Error	Order	Error	Order	Error	Order
[32, 32]	2.299E−3	–	4.518E−4	–	3.187E−2	–	1.253E−2	–
[64, 64]	4.345E−4	2.40	8.788E−5	2.36	9.357E−3	1.77	4.323E−3	1.54
[128, 128]	1.569E−4	1.46	2.435E−5	1.85	6.890E−3	0.44	2.631E−3	0.72
[256, 256]	3.416E−5	2.20	5.300E−6	2.20	3.240E−3	1.09	1.318E−3	1.00
Example 3A								
$[n_x, n_y]$	First order derivative jumps				Second order derivative jumps			
	L_R		L_2		L_R		L_2	
	Error	Order	Error	Order	Error	Order	Error	Order
[32, 32]	4.095E−3	–	5.170E−3	–	0.156	–	0.214	–
[64, 64]	8.467E−4	2.27	9.702E−4	2.41	6.175E−2	1.34	0.102	1.07
[128, 128]	2.855E−4	1.56	2.870E−4	1.76	2.848E−2	1.12	4.875E−2	1.07
[256, 256]	8.976E−5	1.67	6.773E−5	2.08	1.452E−2	0.97	2.494E−2	0.97

method significantly saves the computational time after comparing the CPU time of AMIB to that of MIB. Fig. 9 covering four examples indicates that AMIB method is particularly efficient in calculation of solution on very dense meshes, which gives a persuasive evidence that AMIB is a successful solver.

In our previous examples, the iteration numbers needed by the biconjugate gradient method in solving the Schur complemented system (31) have been reported. Indeed, this number grows slowly when n is increased, where n basically represents the grid number in each direction for a square domain. As discussed above, if impact of iteration number is negligible, the complexity of the AMIB method is essentially $O(n^2 \log n)$. To verify this complexity of flops, we plot the CPU time against n again, but now in log–log scale. See Fig. 10. In particular, we will conduct a least squares fitting in log scale. In this fitting, it is convenient to drop $\log n$ and only concern on the flops order r in the form of $O(n^r)$. It can be seen in Fig. 10 the flops order of the AMIB method is around two in all cases, while that of the regular MIB is about three. In practice, this means the AMIB is much more efficient than the MIB, while preserving the second order accuracy in dealing with various complex interfaces.

4. Summary

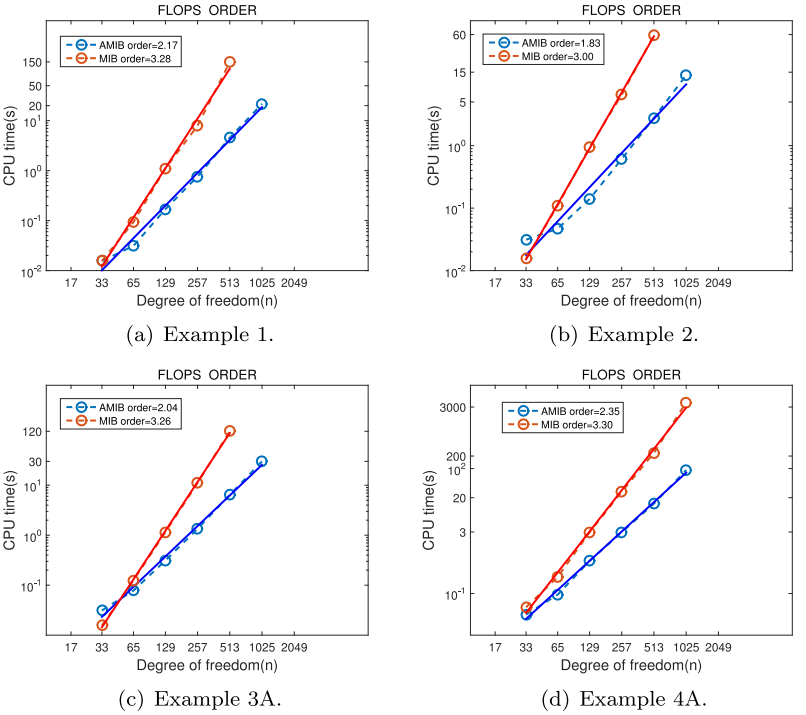
In this work, we proposed a fast and second order accurate method to solve 2D elliptic interface problem based on the MIB method. The proposed augmented MIB (AMIB) method seamlessly combines several key gradients of the MIB, AIIM, and EJIIM schemes. The fictitious values generated by the MIB give a very straightforward way along the Cartesian grid line to approximate the jump quantities needed in the correction terms for the corrected difference. The augmented system in the approach provides an efficient way to solve the elliptic interface problem. The method is shown to be second order convergent and stable via the numerical experiments for various complex interfaces. The AMIB has a very nice property that the iteration number of the biconjugate gradient solver in solving complemented system grows very slowly compared with the growth of the mesh refinement. Consequently, the complexity of our AMIB follows $O(n^2 \log n)$



(a) Example 1. (b) Example 3A.

(c) Example 4A. (d) Example 5.

Fig. 9. CPU time in seconds for both MIB and AMIB methods. The horizontal line is the degree of freedom in one dimension, i.e., $n+1$.



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

(c) Example 3A. (d) Example 4A.

Fig. 10. Flops order in CPU time is examined for several examples. The horizontal line is the degree of freedom in one dimension, i.e., $n+1$.

for n grids in each direction of a square domain. This indicates that our AMIB is a quite effective and efficient algorithm in solving elliptic interface problem.

Different from the existing fast interface algorithms for elliptic equations, this approach has a potential to be generalized in solving parabolic/hyperbolic interface problems due to the MIB treatment in jump quantities approximation. The future plan is to extend the approach for solving time-dependent problems and generalize it into even higher orders.

Acknowledgments

This research is partially supported by the Simons Foundation award 524151, the National Science Foundation (NSF) of USA under grant DMS-1812930, the Natural Science Foundation of China under grant 11461011, and the key project of Guangxi Provincial Natural Science Foundation of China under grant 2017GXNSFDA198014 and 2018GXNSFDA050014.

References

[1] G.R. Hadley, High-accuracy finite difference equations for dielectrix waveguide analysis I: Uniform regions and dielectric interfaces, J. Lightwave Technol. 20 (2002) 1210–1218.
[2] 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.
[3] N.A. Baker, Poisson–Boltzmann methods for biomolecular electrostatics, Meth. Enzymol. 383 (2004) 94–118.
[4] 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.
[5] 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.
[6] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, Computing 5 (1970) 207–213.
[7] Z. Chen, J. Zhou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numer. Math. 79 (1998) 175–202.
[8] E. Burman, P. Hansbo, Interior-penalty-stabilized Lagrange multiplier methods for the finite-element solution of elliptic interface problems, SIAM J. Numer. Anal. 30 (2010) 870–885.
[9] M. Dryjaa, J. Galvisb, M. Sarkisb, BDDC methods for discontinuous Galerkin discretization of elliptic problems, J. Complexity 23 (2007) 715–739.
[10] L.N.T. Huynh, N.C. Nguyen, J. Peraire, B.C. Khoo, A high-order hybridizable discontinuous Galerkin method for elliptic interface problems, Internat. J. Numer. Methods. Engrg. 93 (2013) 183–200.
[11] 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.
[12] 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.
[13] R.E. Ewing, Z.L. Li, T. Lin, Y.P. Lin, The immersed finite volume element methods for the elliptic interface problems, Math. Comput. Simulation 50 (1999) 63–76.
[14] A. Hansbo, P. Hansbo, An unfitted finite element method, Comput. Methods Appl. Mech. Engrg. 191 (2002) 5537–5552.
[15] S.M. Hou, X.-D. Liu, A numerical method for solving variable coefficient elliptic equation with interfaces, J. Comput. Phys. 202 (2005) 411–445.
[16] 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.
[17] K.N. Xia, M. Zhan, G.-W. Wei, MIB Galerkin method for elliptic interface problems, J. Comput. Appl. Math. 272 (2014) 195–220.
[18] 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.
[19] 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.
[20] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[21] C.S. Peskin, Lectures on mathematical aspects of physiology, Lect. Appl. Math. 19 (1981) 69–107.
[22] 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.
[23] X.D. Liu, R.P. Fedkiw, M. Kang, Boundary condition capturing mehtod for Poisson's equation on irregular domain, J. Comput. Phys. 160 (2000) 151–178.
[24] 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.
[25] 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.
[26] Z.L. Li, K. Ito, Maximum principle preserving schemes for interface problems with discontinuous coefficients, SIAM J. Sci. Comput. 23 (2001) 339–361.
[27] 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.
[28] I.-L. Chern, Y.-C. Shu, A coupling interface method for elliptic interface problems, J. Comput. Phys. 225 (2007) 2138–2174.
[29] T. Chen, J. Strang, Piecewise-polynomial discretization and Krylov-accelerated multigrid for elliptic interface problems, J. Comput. Phys. 227 (2008) 7503–7542.
[30] W.J. Ying, W.C. Wang, A kernel-free boundary integral method for implicitly defined surfaces, J. Comput. Phys. 252 (2013) 606–624.
[31] 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.
[32] 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.
[33] H. Wei, L. Chen, Y. Huang, B. Zheng, Adaptive mesh refinement and superconvergence for two-dimensional interface problems, SIAM J. Sci. Comput. 36 (2014) A1478–A1499.
[34] L. Adams, Z.L. Li, The immersed interface/multigrid methods for interface problems, SIAM J. Sci. Comput. 24 (2002) 463–479.
[35] Z.L. Li, A fast iterative algorithm for elliptic interface problem, SIAM J. Numer. Anal. 35 (1998) 230–254.
[36] 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.
[37] P.A. Berthelsen, A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions, J. Comput. Phys. 197 (2004) 364–386.

[38] 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.
[39] Bengt Fornberg, Calculation of weights in the finite difference formuas, SIAM Rev. 40 (1998) 685–691.
[40] S. Zhao, A matched alternating direction implicit (ADI) method for solving the heat equation with interfaces, J. Sci. Comput. 63 (2015) 118–137.
[41] Z. Wei, C. Li, S. Zhao, A spatially second order alternating direction implicit (ADI) method for three dimensional parabolic interface problems, Comput. Math. Appl. 75 (2018) 2173–2192.
[42] X. Chen, X. Feng, Z. Li, A direct method for accurate solution and gradient computations for elliptic interface problems, Numer. Algorithms (2008) http://dx.doi.org/10.1007/s11075-018-0503-5.