AUGMENTED MATCHED INTERFACE AND BOUNDARY(AMIB) METHOD FOR SOLVING INTERFACE AND BOUNDARY VALUE PROBLEMS

by

HONGSONG FENG

SHAN ZHAO, COMMITTEE CHAIR
DAVID HALPERN
WEI ZHU
MOJDEH RASOULZADEH
XIAOWEN WANG

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Mathematics
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2021

Copyright HONGSONG FENG 2021
ALL RIGHTS RESERVED

ABSTRACT

This dissertation is devoted to the development of the augmented matched interface and boundary(AMIB) method and its applications for solving interface and boundary value problems.

We start with a second order accurate AMIB introduced for solving two-dimensional (2D) elliptic interface problems with piecewise constant coefficients, which illustrates the theory of AMIB illustrated in details. AMIB method is different from its ancestor matched interface and boundary (MIB) method in employing fictitious values to restore the accuracy of central differences for interface and boundary value problems by approximating the corrected terms in corrected central differences with these fictitious values. Through the augmented system and Schur complement, the total computational cost of the AMIB is about $O(N \log N)$ for degree of freedom N on a Cartesian grid in 2D when fast Fourier transform(FFT) based Poisson solver is used. The AMIB method achieves $O(N \log N)$ efficiency for solving interface and boundary value problems, which is a significant advance compared to the MIB method.

Following the theory of AMIB in chapter 2, chapter 3 to chapter 6 cover the development of AMIB for a high order efficient algorithm in solving Poisson boundary value problems and a fourth order algorithm for elliptic interface problems as well as efficient algorithm for parabolic interface problems. The AMIB adopts a second order FFT-based fast Poisson solver in solving elliptic interface problems. However, high order FFT-based direct Poisson solver is not available in the literature, which imposes a grand challenge in designing a high order efficient algorithm for elliptic interface problems. The AMIB method investigates efficient algorithm of Poisson boundary value problem (BVP) on rectangular and cubic domains by converting Poisson BVP to an immersed boundary problem, based on which a high order FFT algorithm is proposed. This naturally allows for fulfilling a fourth order fast algorithm for solving elliptic interface problems. Besides the FFT algorithm, a multigrid method is also considered to achieve high efficiency in solving parabolic

interface problems.

Extensive numerical results are included in each chapter of the concerned problem, and are used to show the robustness and efficiency of AMIB method.

Keyword: Elliptic interface problem; Parabolic interface problem; Corrected differences; High order accuracy; Fast Fourier transform(FFT); $O(N \log N)$ algorithm; Augmented Matched interface and boundary (AMIB); Augmented system; Schur complement; Multigrid.

DEDICATION

To my parents, mentors, friends and all those who have ever helped me in my life.

ACKNOWLEDGMENTS

First and foremost, I am gratefully indebted to my advisor Professor Shan Zhao, and wish to express my deepest appreciation to him for his persistent guidance, patience, and encouragement on my research over the past five years of my PhD study. His integrity, enthusiasm and professionalism inspire me to pursue academic advances, and to become an educator in the future.

I would like to acknowledge helpful suggestions from my committee members: Dr. David Halpern, Dr. Wei Zhu, Dr. Mojdeh Rasoulzadeh, and Dr. Xiaowen Wang. I thank them for their time, support and expertise to improve my dissertation work.

Many faculty and staff in the math department of The University of Alabama gave me great assistance and encouragement in various ways during the course of my PhD study. I really appreciate all I have learnt from the professors in classes. I am especially grateful to Professor David Halpern, Professor David Cruz-Uribe and Ms. Natalie Lau for their continuous assistance during the long time of my study at UA.

Especially, I would like to thank Professor Runchang Lin as my master advisor during my study at Texas A&M International University. His patient guidance led me to my academic life, and I appreciate his continuous support over these years since I came to USA.

Last but not least, I am grateful to my parents and sister for their blessings and selfless support in my path of academic pursuance and journey of life.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Elliptic interface problems have drawn a great amount of attention due to their wide applications in many fields such as computational electromagnetics and optics [33, 79], biomolecular electrostatics [5, 30], and material science [38]. Their mathematical models are 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

$$-\nabla \cdot (\beta \nabla u) = f(x, y), \quad (x, y) \in \Omega \tag{1.1}$$

with the Dirichlet boundary condition

$$u(x, y) = g(x, y), \quad (x, y) \in \partial\Omega. \tag{1.2}$$

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 so that $\Omega = \Omega^+ \cup \Omega^-$. 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 dissertation, 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

$$[\![u]\!] := u^+ - u^- = \phi(x, y), \tag{1.3}$$

$$[\![\beta u_n]\!] := \beta^+ \nabla u^+ \cdot \vec{n} - \beta^- \nabla u^- \cdot \vec{n} = \psi(x, y), \tag{1.4}$$

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. Equations (1.3) and (1.4) are called 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 a 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 of 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 [4]. In continuous FEMs, complex interfaces are represented by using body-fitted unstructured grids, while the jump conditions can be satisfied in the variational formulations [13, 16]. Recently, both discontinuous Galerkin (DG) [20, 39] and weak Galerkin (WG) [56, 57] type discontinuous FEMs have been constructed for solving elliptic interface problems. Immersed FEM [22, 34, 36, 37, 73] 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 [22]. Other interface approaches, such as enforcing jump conditions via fictitious nodes [73] and Lagrange

multipliers [36, 37], have also been developed. Simple procedures, based on Delaunay triangulation [14] and Voronoi diagram [32] respectively, have been introduced to alter uniform grids locally to generate semi-structured interface-fitted meshes, which lead to effective virtual element [14] and finite volume [32] approaches.

The development of Cartesian grid finite difference methods for solving elliptic interface problems has received much attention in the past several decades. Peskin [60, 61] proposed the immersed boundary method (IBM) to model blood flow in the heart. In this method, singular forces are smeared out by the discrete delta function. It proved to be a robust and efficient method, and it is typically first order in high dimension applications. Fedkiw, Osher and coworkers [23, 54] introduced the ghost fluid method (GFM) to treat contact discontinuities in the invisicid Euler equation. The principle lies in extending the piecewise function into the other subdomains 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 [51] recently. LeVeque and Li [42, 48] have invented the immersed interface method (IIM) which introduces correction terms of jump conditions into the 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 the zeroth and first order jump conditions (1.3) and (1.4) , a matched interface and boundary (MIB) method has been introduced in [85], 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 [85]. In treating smoothly curved interfaces, the MIB method can usually achieve fourth order convergence [79, 85]. Other effective Cartesian grid methods for elliptic interface problems include the coupling interface method [17], the piecewise-polynomial interface method [15], kernel free integral equation method [75], and the virtual node method [8, 35].

There is a great interest in developing fast interface algorithms to accelerate algebraic

computations by means of fast Poisson solvers, which include geometric multigrid methods with a complexity $O(N)$ and fast Fourier transforms (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. [70] 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 [2]. Later, multigrid solvers have also been applied in other Cartesian grid methods, including the piecewise-polynomial interface method [15] and virtual node method [8, 35].

A significant achievement in the field is the augmented IIM (AIIM) [45, 50] 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 the 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 [45] and the multigrid for piecewise variable coefficients [50]. Another effective approach for decomposing jump conditions was introduced in [6, 72]. 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 [72], 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 [6]. 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 the fast Poisson solver. The jump corrections are realized by means of auxiliary variables which can be solved either through the Schur complement method [45, 50, 72] or from the previous iterative step [6]. The robustness of these numerical schemes crucially depends on the numerical approximation of 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.

This dissertation discusses augmented MIB method (AMIB) for solving elliptic interface problems with piecewise constant coefficients, which combines key features of AIIMs [6, 45, 50, 72] and MIB method [79, 85] 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 the partial differential equation (PDE) discretization. Instead, jump corrected Taylor expansions introduced in the EJIIM [6, 72] 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 [45, 50]. 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.

With the progress achieved in fulfilling a second order fast algorithm for elliptic interface problems, combining higher order convergence with fast Poisson solvers could further improve the computational efficiency. This requires spending efforts in both designing high order accurate algorithm and advanced strategy to accomplish high computational speed.

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 [44] 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 [44] and DG [39] 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 [56, 57]. For Cartesian grid finite difference methods, a fourth order IIM could be constructed by computing high order jump conditions involving mixed derivatives [49]. By iteratively using zero and first order jump conditions, the MIB method [79, 85] normally achieves fourth order rate 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 [84], another fourth order IIM was developed in two-dimensions (2D) by iteratively using zero and first order jump conditions too.

With all fast interface algorithms mentioned above achieving second order accuracy and high order algorithms of order over two, 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 [11, 55, 64, 74]. 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 [44]. This brings extra difficulties for applying geometrical multigrid methods to fourth order FEMs. For fast IIMs including AIIM [45, 50] and EJIIM [72], the extension from second order to fourth order requires up to fourth order jump conditions involving mixed derivatives [49], which impose a big challenge in numerical implementation. We note that such a difficulty is simply bypassed in the fourth order MIB [85] by imposing zeroth and first order jump conditions repeatedly. Therefore, the combination of fourth order MIB [85] with the augmented MIB [24] seems promising.

With regard to this, AMIB for solving elliptic interface problems would attempt to achieve fourth order accuracy in dealing with interfaces and boundaries, but also to maintain the FFT complexity of $O(N \log N)$. Based on our previous second order AMIB [24], the present study addresses two major difficulties. In resolving the first difficulty, the fourth order MIB [85] is applied in the framework of the augmented approach to treat a smoothly curved interface. However, the combination of the fourth order MIB [85] with the AMIB [24] faces another grand difficulty: An FFT-based fourth order central difference scheme has never been reported for Poisson boundary value problems over cubic domains before our studies, considering that fact that MIB [85] adopts fourth central difference to fulfill high order accuracy.

To attack the aforementioned high order computational acceleration issue, an assumption of solution anti-symmetry across boundary is used in constructing an efficient algorithm for solving Poisson BVP on a cubic domain [26]. To generalize the anti-symmetry assumption to other general boundary conditions, artificial zero-padding zone is extended beyond the original domain. Besides, high order corrected differences with Cartesian derivative jumps are derived to account for the accuracy restoration across the original boundaries. MIB fictitious values are then deployed for approximation to the Cartesian derivative jumps. All these ingredients are then formulated in AMIB framework to obtain fast solution Poisson BVP with Dirichlet, Neumann, Robin or any combination of these boundary conditions, and arbitrarily high order in principle with $O(N \log N)$ complexity are accomplished. In this way, a high order central difference based FFT algorithm is proposed to efficiently solve Poisson BVP in [26]. As a further application [25] of AMIB, Poisson BVP on staggered boundary conditions is attacked by AMIB due to the popular utilization of staggered grid in Poisson related problems.

Such efficiency strategy by AMIB in [26] naturally allows us to consider high order efficient algorithm for elliptic interface problems. By introducing auxiliary variables both at interior interfaces and exterior boundaries, 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 $O(N \log N)$ [27].

In addition to efficient algorithms for elliptic interface problem as modeled by (1.1)-(1.4), interests are also attracted toward fast algorithm for solving parabolic interface problem, which has piece-wise discontinuous coefficient diffusion equation across the interface as governing equation, and has time-dependent boundary and interface conditions. FFT is not readily applicable to such parabolic interface problem due to the non-constant coefficient of the reaction term in the Helmholtz equation. We propose a new multigrid method based on the augmented matched interface and boundary (AMIB) method for solving parabolic interface problems. In designing efficient algorithm, attempts have been made in several directions. On one hand, alternating direction implicit (ADI) algorithms have been designed [43, 46, 47, 52, 53, 71, 71, 80] for parabolic interface problems. The core philosophy of ADI methods [19, 59] is to reduce a multidimensional system from discretizing parabolic equation to a sequence of independent one-dimensional (1D) sub-systems of tridiagonal structure, which can be solved efficiently with Thomas algorithm. Compared with generic iterative solvers, ADI methods can be regarded as an exact solver due to the fact that computation is completed within a fixed number of steps. Recently, ADI schemes have been developed to cope with parabolic interface problems with desired ADI efficiency $O(N)$ achieved. On the other hand, multigrid method is also utilized for algorithm design for such problem. Based on structured or unstructured meshes, approaches for constructing interpolation, smoothing operators and coarse gird points selection would require special designs. Progresses have been made by IIM-based multigrid method [1, 2] and other methods [18, 67]. Despite the great success achieved by the aforementioned multigrid methods, a new multigrid method has to be designed again when a new interface algorithm is under consideration. Because each interface algorithm has a different procedure to capture boundary or interface conditions, sophisticated treatments are demanded to define interpolation with knowledge of geometry from the interface and to define restriction and coarse grid operators. Due to the augmented approach adopted, the finite difference interface treatment and the multigrid solution procedure can be decoupled in the proposed AMIB method. This considerably simplifies the interpolation and restriction process for the coarse grid operators in the multigrid method.

Hence standard multigrid components can be applied in such augmented framework with the advantage that storage for a coarse grid matrix is avoided and the implementation is fairly simple. Second order corrected central differences are applied for spatial accuracy to compensate accuracy loss across the interface, and the Crank-Nicolson scheme is used for temporal discretization. This leads to second order accuracy in both space and time, while the multigrid approach accelerates the computational speed.

The rest of this dissertation is organized as follows. Chapter 2 focuses on the development of second AMIB method for solving elliptic interface problem, where the main theory of AMIB method is demonstrated in details. Following this, the development of AMIB method in solving different problems is shown from chapter three to chapter 6. A conclusion in included in chapter seven.

CHAPTER 2

SECOND ORDER AMIB FOR ELLIPTIC INTERFACE PROBLEM

The augmented matched interface and boundary(AMIB) method is developed based on MIB method. It was first explored to obtain second order fast solution of elliptic interface problem (1.1) subject to conditions (1.2), (1.3) and (1.4). The framework of AMIB method will be illustrated based on the discussion for solving such second order fast solution of elliptic interface problem (1.1). Then further development of AMIB is carried out through solving a few other problems, which will be shown in later chapters of this dissertation.

2.1 AMIB Theory and algorithm

We focus on piecewise constant coefficient elliptic problem. Hence, our original PDE (1.1) can be rewritten as below after moving the coefficient β to the right hand side of the equation,

$$\triangle u = -\frac{f(x,y)}{\beta}, \quad (x,y) \in (\Omega^- \cup \Omega^+) \setminus \Gamma \tag{2.1}$$

with the same Dirichlet boundary conditions (1.2) and interface jump conditions (1.3) and (1.4). 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

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

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:

$$\frac{\partial^2 u(x_i, y_j)}{\partial x^2} \approx \frac{u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)}{h^2} \tag{2.2}$$

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.

2.1.1 Correcting finite difference for Laplacian

In the MIB method [79, 85], 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 $x_i \leq \alpha < x_{i+1}$. If the Cartesian jumps are known, we have the corrected Taylor

expansions at two irregular points [72]

$$u(x_{i+1}) = \sum_{k=0}^{l} \frac{h^k}{k!} u^{(k)}(x_i) + \sum_{k=0}^{l} \frac{(h^+)^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{2.3}$$

and

$$u(x_i) = \sum_{k=0}^{l} \frac{(-h)^k}{k!} u^{(k)}(x_{i+1}) - \sum_{k=0}^{l} \frac{(h^-)^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{2.4}$$

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 (2.3) and (2.4), it is easy to derive following differences at irregular points.

Jump-corrected difference. Suppose $u \in C^4[x_j - h, \alpha) \cap C^4(\alpha, x_{j+1} + h]$ 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 $m = 3$ [72]:

$$u_{xx}(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - \frac{1}{h^2} \sum_{k=0}^{m} \frac{(h^+)^k}{k!} [u^{(k)}], \tag{2.5}$$

$$u_{xx}(x_{i+1}) \approx \frac{u(x_{i+2}) - 2u(x_{i+1}) + u(x_i)}{h^2} + \frac{1}{h^2} \sum_{k=0}^{m} \frac{(h^-)^k}{k!} [u^{(k)}]. \tag{2.6}$$

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. 2.2 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 $x_{j-1} \le \alpha_1 < x_j \le \alpha_2 < x_{j+1}, h_1^- = x_{j-1} - \alpha_1, h_1^+ = x_j - \alpha_1, h_2^- = x_j - \alpha_2$, and $h_2^+ = x_{j+1} - \alpha_2$. For example, we may have the situation where $x_i \in \Omega^-$ and $x_{i-1}, x_{i+1} \in \Omega^+$. Assume $u \in C^4[x_{j-1}, \alpha) \cap C^4(\alpha_1, \alpha_2) \cap C^4(\alpha_2, x_{j+1}]$, 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 $m = 3$ [72],

$$u_{xx}(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \frac{1}{h^2} \sum_{k=0}^{m} \frac{(h_1^-)^k [u^{(k)}]_{\alpha_1} - (h_2^+)^k [u^{(k)}]_{\alpha_2}}{k!}. \tag{2.7}$$

Remark 2.1.1. *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* (2.5)*,* (2.6) *and* (2.7)*, which allow the use of the FFT algorithm.*

Remark 2.1.2. *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 [72] 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.*

Bearing the above $x-$oriented corrected difference (2.5) (2.6) and (2.7) 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 $P_1(x_i, y_j)$

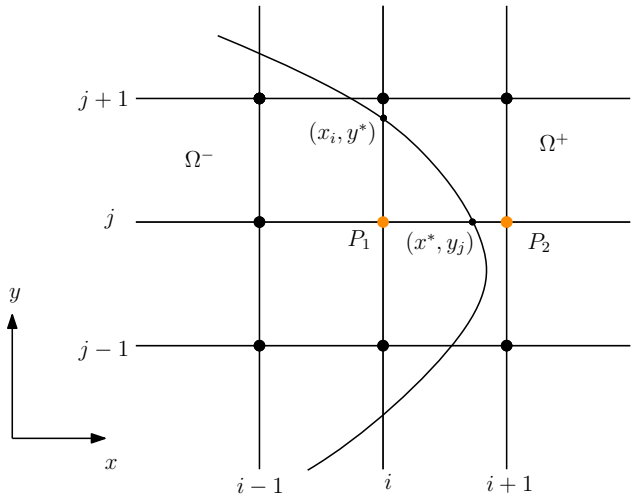

Figure 2.1: Irregular points $P_1(i, j)$ and $P_2(i+1, j)$. The interface intersects $y = y_j$ at (x^*, y_j) and $x = x_i$ at (x_i, y^*)

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. 2.1. 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 (2.5) applied in both $x-$ and $y-$direction [72]. Thus the approximation can be formulated as below with a local truncation error $O(h)$:

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

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 (2.6) 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. 2.2, multiple corrected difference (2.7) is needed in $y-$direction while one correction (2.5) 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

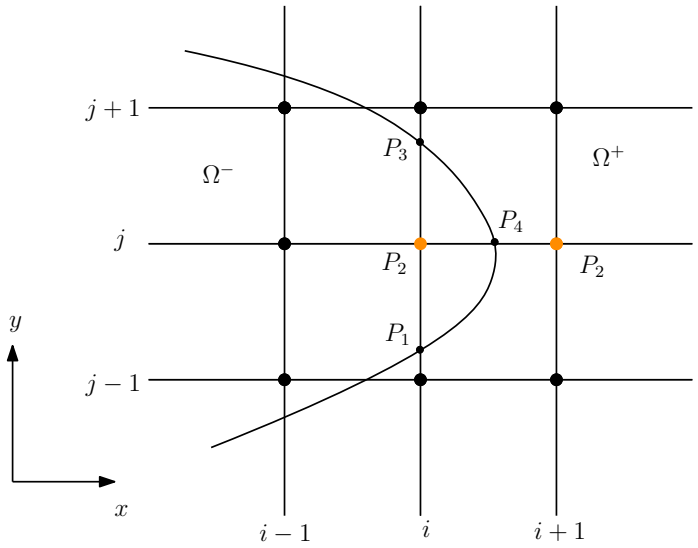

Figure 2.2: Corner point $P_2(i, j)$. The interface intersects with $x = x_i$ at $P_1(x_i, y_1^*)$ and $P_3(x_i, y_2^*)$ while it intersects with $y = y_j$ at $P_4(x^*, y_j)$.

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: [72]

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

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 (2.5) and (2.6).

2.1.2 Reconstructing the Cartesian derivative jumps

So far we have established the corrected differences with Cartesian derivative jumps undetermined, which are defined as

$$[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} = \lim_{x\to\alpha^+} \frac{\partial^k u}{\partial x^k} - \lim_{x\to\alpha^-} \frac{\partial^k u}{\partial x^k}, \tag{2.8}$$

where $k = 0, 1, 2$ indicate the order of derivative.

These jumps are solution dependent and their reconstruction is non-trivial so that it needs appropriate numerical approximation as one part of the solution. To maintain a first order convergence locally for the irregular points, an interpolation polynomial with third order accuracy is expected to approximate the interfacial 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.

For sake of accuracy and stability, it is better to approximate each one-sided derivative limit with central or pseudo-central difference. As shown in Fig.2.3, if one layer of fictitious values are available on each side of the interface $x = \alpha$, we can combine those with two real function values on the other side of the interface to yield the desired two Lagrange polynomials. The derivative jumps can be approximated as below:

$$[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} \approx (w^k_{i,j}\hat{u}_{i,j} + \sum_{l=1}^{2} w^k_{i+l,j}u_{i+l,j}) - (\sum_{l=1}^{2} w^k_{i-2+l,j}u_{i-2+l,j} + w^k_{i+1,j}\hat{u}_{i+1,j}), \tag{2.9}$$

where the linear combinations in the two parenthesis denote the derivatives of the two constructed Lagrange polynomials for approximating one-sided derivative limits at $x = \alpha$.

It is possible that only one real function value is available on one side of the interface due to sharp topology change of the interface. To deal with such a corner case, fictitious value is employed

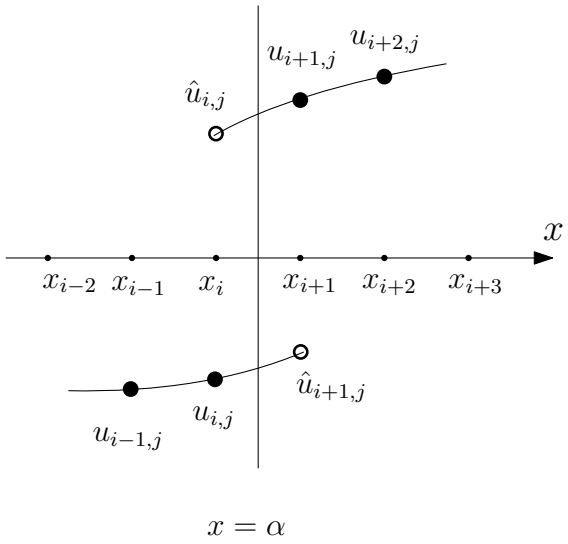

Figure 2.3: Black dots denote real function values, while empty circles indicate fictitious values. They are used in the polynomials for approximating derivative jump across the interface at $x = \alpha$.

where the needed real value is missing. Just consider the derivative jumps at $x = \alpha_2$ in the two scenarios shown in Fig.2.4. The fictitious value at $x = x_{i-1}$ is used for the case in the right figure in comparison with the case in the left figure where the real value at $x = x_{i-1}$ is adopted. Likewise, the discussions for the $y-$direction derivative jump can be formulated when the interface intersects with the x grid line. To apply the approximation (2.9), we need to construct fictitious values at all the irregular points. In following subsection, MIB method is employed to generate the needed fictitious values.

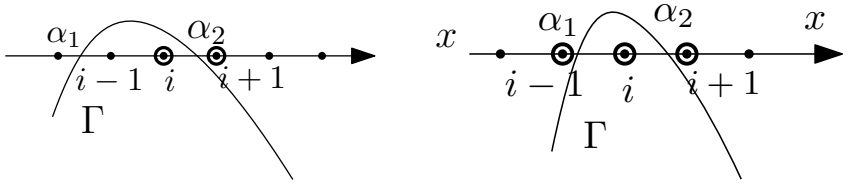

Figure 2.4: Derivative jump approximation in two scenarios. Empty circles denote the fictitious values at the grid points, while black dots indicate real function values.

Fictitious values formulation

The MIB method [79, 85] is employed to generate fictitious values in the vicinity of the interface with the aid of known jump conditions (1.3) and (1.4). One more jump condition could be

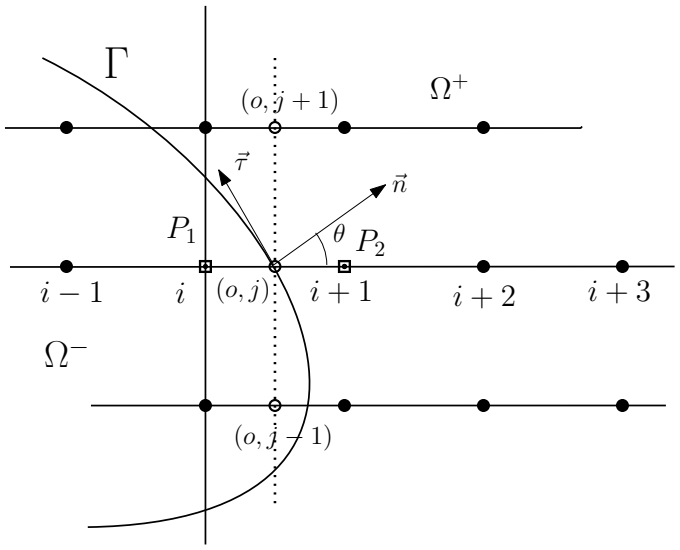

Figure 2.5: Consider the case where the interface intersects $y = y_j$ at (x_o, y_j). At the irregular points $P_1(i, j)$ and $P_2(i+1, j)$, fictitious values (in box) can be constructed. Here θ is the angle between positive $x-$ direction and the normal vector $\vec{n}$.

analytically derived by differentiating Eq. (1.3) along the tangential direction τ of the interface as below,

$$[\![u_\tau]\!] = u_\tau^+ - u_\tau^- = \rho(x, y) \tag{2.10}$$

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 $\vec{n} = (\cos\theta, \sin\theta)$, respectively. See Fig. 2.5 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 [85],

$$[\![u]\!] = u^+ - u^- = \phi(x^*, y^*), \tag{2.11}$$

$$[\![u_\tau]\!] = (-u_x^+ \sin\theta + u_y^+ \cos\theta) - (-u_x^- \sin\theta + u_y^- \cos\theta) = \rho(x^*, y^*), \tag{2.12}$$

$$[\![\beta u_n]\!] = \beta^+(u_x^+ \cos\theta + u_y^+ \sin\theta) - \beta^-(u_x^- \cos\theta + u_y^- \sin\theta) = \psi(x^*, y^*). \tag{2.13}$$

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^-. The following formulations (2.14) or (2.15) can be obtained by eliminating u_y^- or u_x^- from the four quantities. If u_y^- is eliminated, the following equations can be derived

$$[\![u]\!] = u^+ - u^-, \;\; [\![\beta u_n]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ u_x^+ - C_x^- u_x^- + C_y^+ u_y^+, \tag{2.14}$$

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

$$[\![u]\!] = u^+ - u^-, \;\; [\![\beta u_n]\!] + \beta^- \cot\theta [\![u_\tau]\!] = D_y^+ u_y^+ - D_y^- u_y^- + D_x^+ u_x^+, \tag{2.15}$$

where $D_x^+ = (\beta^+ - \beta^-)\cos\theta, D_y^+ = \beta^- \cos\theta \cot\theta + \beta^+ \sin\theta$ and $D_y^- = \beta^-(\cos\theta \cot\theta + \sin\theta)$. Eqns. (2.14) and (2.15) allows us to reduce the 2D jump conditions to 1D ones along Cartesian direction provided that u_y^+ and u_x^+ can be determined, respectively. Suppose the interface intersects with $y = y_j$ at (x_o, y_j) as shown in Fig. 2.5. To generate a pair of fictitious values $\hat{u}_{i,j}$ and $\hat{u}_{i+1,j}$ at points P_1 and P_2 on each side of the interface, discretization is carried out on Eqn.(2.14) as below,

$$[\![u]\!] = W_0^+ U^+ - W_0^- U^- \tag{2.16}$$

$$[\![\beta u_n]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ W_1^+ U^+ - C_x^- W_1^- U^- + C_y^+ P^+ U_o, \tag{2.17}$$

with the following vector representations

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

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

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

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

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 (2.14) 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_o 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 three auxiliary function values in U_o are at points $(x_o, y_{j-1}), (x_o, y_j)$,$(x_o, y_{j+1})$, see Fig 2.5. Additionally, each of these three auxiliary values are interpolated or extrapolated by three function values in Ω^+. For example, in Fig. 2.5, $u_{o,j+1}$ is interpolated by function values $u_{i,j+1}, u_{i+1,j+1}$ and $u_{i+2,j+1}$. Similarly, $u_{o,j}, u_{o,j-1}$ will be extrapolated by three function values from their right side in the Ω^+ domain.

With the function values in U_o appropriately approximated, (2.16) and (2.17) 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

$$\begin{aligned} \hat{u}_{i,j} &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 [\![u]\!] + W_1 [\![u_\tau]\!] + W_2 [\![\beta u_n]\!] \\ &= \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 \phi(x^*, y^*) + W_1 \rho(x^*, y^*) + W_2 \psi(x^*, y^*), \end{aligned} \tag{2.18}$$

where $\mathbb{S}_{i,j}$ represents a set of surrounding nodes involved in (2.16) and (2.17).

In this manner, the needed two layers of fictitious values can be generated along $x-$direction by

enforcing the jump conditions (2.14) or along $y-$direction by enforcing (2.15). Interested readers are referred to [85] for more details.

Remark 2.1.3. *The above MIB formulation utilizes extrapolations in Ω^+ only, i.e., u_y^+ and u_x^+ in* (2.14) *and* (2.15) *are perpendicularly approximated by using auxiliary values, and each auxiliary value will be extrapolated by using function values in Ω^+. Recall that Eqs.*(2.14) *and* (2.15) *are derived after u_y^- or u_x^- being eliminated from Eqs.*(2.11) - (2.13)*. On the other hand, if we remove u_y^+ or u_x^+ from Eqns* (2.11) - (2.13)*, 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^-. 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.*

Remark 2.1.4. *The accuracy of derivative jump reconstruction in Eq.* (2.9) *could be affected by the fictitious value generated by 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.*

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. 2.3, 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

$$[\frac{\partial u}{\partial x}] = \lim_{x\to(x^*)^+} \frac{\partial u}{\partial x}(x, y_j) - \lim_{x\to(x^*)^-} \frac{\partial u}{\partial x}(x, y_j)$$

$$= w^+_{1,1}\hat{u}_{i,j} + w^+_{1,2}u_{i+1,j} + w^+_{1,3}u_{i+2,j} - w^-_{1,1}u_{i-1,j} - w^-_{1,2}u_{i,j} - w^-_{1,3}\hat{u}_{i+1,j} + O(h^2),$$

$$[\frac{\partial^2 u}{\partial x^2}] = \lim_{x\to(x^*)^+} \frac{\partial^2 u}{\partial x^2}(x, y_j) - \lim_{x\to(x^*)^-} \frac{\partial^2 u}{\partial x^2}(x, y_j)$$

$$= w^+_{2,1}\hat{u}_{i,j} + w^+_{2,2}u_{i+1,j} + w^+_{2,3}u_{i+2,j} - w^-_{2,1}u_{i-1,j} - w^-_{2,2}u_{i,j} - w^-_{2,3}\hat{u}_{i+1,j} + O(h),$$

which are equivalent to

$$w^-_{1,1}u_{i-1,j} + w^-_{1,2}u_{i,j} - w^+_{1,2}u_{i+1,j} - w^+_{1,3}u_{i+2,j} - w^+_{1,1}\hat{u}_{i,j} + w^-_{1,3}\hat{u}_{i+1,j} + [\frac{\partial u}{\partial x}] = 0, \tag{2.19}$$

$$w^-_{2,1}u_{i-1,j} + w^-_{2,2}u_{i,j} - w^+_{2,2}u_{i+1,j} - w^+_{2,3}u_{i+2,j} - w^+_{2,1}\hat{u}_{i,j} + w^-_{2,3}\hat{u}_{i+1,j} + [\frac{\partial^2 u}{\partial x^2}] = 0, \tag{2.20}$$

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

In addition to the two approximations to derivatives by (2.19) and (2.20), 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 (1.3).

$$[u] = \lim_{x \to (x^*)^+} u(x, y_j) - \lim_{x \to (x^*)^-} u(x, y_j) = c[\![u]\!] = c\phi(x^*, y_j), \tag{2.21}$$

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 (1.3). This clarification applies for the below and above subdomain in the study of $y-$direction derivative jumps.

In the following, we denote a vector $\tilde{Q} = ([u], [\frac{\partial u}{\partial x}], [\frac{\partial^2 u}{\partial x^2}])^T$ as the jump quantities in correction term for one irregular point. Equations (2.19), (2.20) and (2.21) can be rewritten into a matrix-vector form for the jump quantities formulation

$$\tilde{C}\tilde{U} + I\tilde{Q} = \tilde{\Phi} \tag{2.22}$$

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 $c\phi(x^*, y_j)$ in (2.21).

2.1.3 Augmented system

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 (2.1) is discretized in all interior nodes as

$$L_h U_{i,j} + C_{i,j} = -\frac{f_{i,j}}{\beta_{i,j}}, \;\; 1 \le i \le n_x - 1, \;\; 1 \le j \le n_y - 1, \tag{2.23}$$

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

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

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.(2.23), a relation between numerical solution and jump values of $x-$or $y-$partial derivatives is obtained as

$$AU + BQ = F \tag{2.24}$$

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 (2.23) after homogenizing the boundary conditions. Variable U stands for the numerical solution for all grid nodes. Similar to $\tilde{Q}$ introduced in (2.22) for local approximation to a single set of correction jump quantities, here Q is a vector of all auxiliary variables $[u]_i, [\frac{\partial u}{\partial x}]_i, [\frac{\partial^2 u}{\partial x^2}]_i, i = 1, 2, \cdots$, and $[u]_j, [\frac{\partial u}{\partial y}]_j, [\frac{\partial^2 u}{\partial y^2}]_j, j = 1, 2, \cdots$ 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

$$CU + IQ = \Phi \tag{2.25}$$

where the notations in (2.25) share the same meaning as their local counterparts in Eq. (2.22). 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 (2.24) and (2.25),

$$KW = R, \tag{2.26}$$

where

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

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 (2.26) gives a Schur complement system for Q,

$$(I - CA^{-1}B)Q = \Phi - CA^{-1}F, \tag{2.27}$$

which has a much smaller dimension. To solve for Q in equation system (2.27), biconjugate gradient method could be utilized. Once Q is determined, U could be solved by one more FFT on

$$AU = (F - BQ). \tag{2.28}$$

Solving the Schur complement system (2.27) 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 (2.27), we give a initial guess $Q = (0, 0, \cdots, 0)^T$. The terminating tolerance in biconjugate gradient iteration can be set based on the expected 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 (2.27), 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 [45, 50]. Consequently, the complexity of the AMIB is primarily determined by the cost of matrix-vector product in (2.27) 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.

Remark 2.1.5. *In the standard MIB methods, fictitious values calculated above will be directly substituted into the central difference stencil to fulfill the approximation for Laplacian at irregular points. However, this leads to irregular matrix structures, which limit the computational efficiency. The AMIB method overcomes such efficiency limitation by preserving the standard central difference stencil. It consists of several key components that allow itself to be a fast solver: the augmented system, Schur complement procedure and fast Poisson solver via FFT. The corrected differences and the numerical approximation to the correction terms in the corrected difference by using MIB fictitious values formulate the augmented system. Fast Poisson solver is needed in the Schur complement in order to fulfill fast solution of the elliptic interface problem. When high order accuracy is expected in solving elliptic interface related problem, high order corrected differences and high order fast Poisson solver are needed. In following chapters, applications of AMIB method will be explored, and the algorithms are all built on the augmentation framework.*

Remark 2.1.6. *The efficiency of the second 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 $O(N \log N)$. Fortunately, the iteration is for solving* (2.27)*, whose dimension $M \ll N$. 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 $O(N \log(N))$ for the AMIB algorithm. In our numerical experiments, which will be presented in the next section, we have found that the iteration number weakly depends the mesh size.*

2.2 Reducing condition number

Remark 2.1.4 discusses an iteration reducing strategy in solving the auxiliary variables, which can eventually save computational time. The iteration number in one iterative computing process is largely determined by the condition number of the coefficient matrix. In our AMIB method, the generation of fictitious values can significantly affect the resulting coefficient matrix, as it contributes to the sparse matrix structure and coefficients. In this section, two ways of fictitious values generation are investigated, and the resulting condition numbers are compared after their representations are distributed into the finalized coefficient matrix.

We restrict our discussion for the condition number issue to 1D elliptic interface problem. The 1D elliptic interface problem is modeled as

$$\beta u_{xx} = f, \tag{2.29}$$

with jump conditions

$$\begin{cases} [u] = \phi(x), x \in \Gamma \\ [\beta u_n] = \psi(x), x \in \Gamma \end{cases}$$

where

$$\beta = \begin{cases} \beta^-, x \in [a, \Gamma] \\ \beta^+, x \in [\Gamma, b] \end{cases}$$

and

$$f(x) = \begin{cases} f^-(x), x \in [a, \Gamma] \\ f^+(x), x \in [\Gamma, b] \end{cases}$$

The interval $[a, b]$ is partitioned into N subintervals. Supposed that the interface Γ lies in between x_i and x_{i+1}. It is observed in the previous numerical experiment that the interface location and the ration between β^+ and β^- can affect the condition number of the resulting discretization matrix.

2.2.1 Traditional MIB

The traditional MIB method adopts the four adjacent nodes to generate two layers of fictitious values at x_i and x_{i+1}. Plugging the two fictitious values into the second order central difference stencil gives the matrix of MIB2. However, the matrix tends to become ill-conditioned for two cases:

1. Γ is quite close to x_i and $\beta^+ >> \beta^-$.

2. Γ is quite close to x_{i+1} and $\beta^- >> \beta^+$.

Now set $N = 400$ such that $x_j = j\Delta x$ for $j = 0, 1, \cdots, N$ are formed. Let $i = 199$, and $\Gamma = x_i + \alpha\Delta x$ for $\alpha = 0.01, 0.02, \cdots, 0.99$.

We compare the condition number of MIB2 to that of standard second order discretization matrix, and illustrate their ratios corresponding to $\alpha = 0.01, 0.02, \cdots, 0.99$ in the Fig.2.6.

Let $\beta^- = 1$ and $\beta^+ = 1000$. The left figure in Fig.2.6 shows that the MIB2 matrix becomes ill-conditioned when Γ gets close to x_i.

Let $\beta^- = 1000$ and $\beta^+ = 1$. The right figure in Fig.2.6 shows that the MIB2 matrix becomes ill-conditioned when Γ gets close to x_{i+1}.

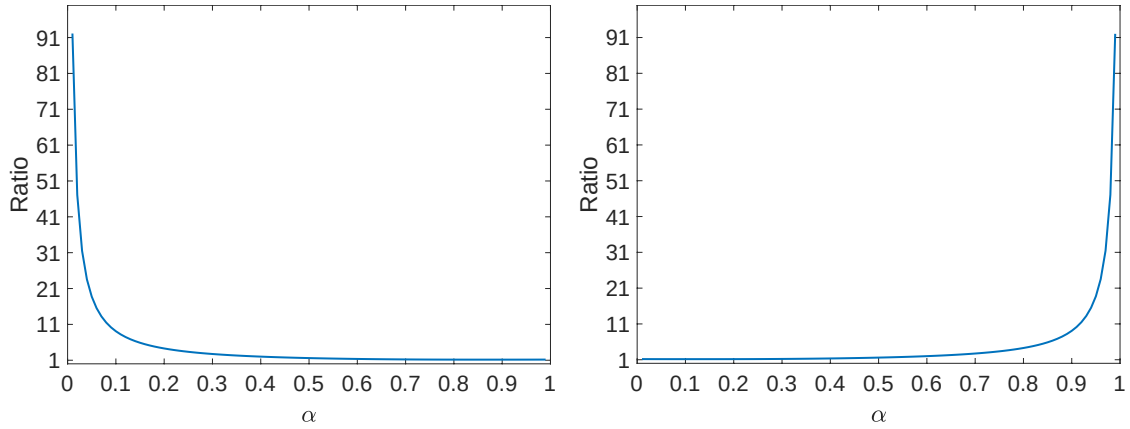

Figure 2.6: The ratios for $\beta^+ >> \beta^-$ (left) and The ratios for $\beta^+ << \beta^-$(right).

2.2.2 Shifting fictitious point

To remedy the issues in the above two cases in MIB, we propose the shifting fictitious point(SFP) approach.

For case 1, we use x_{i-1}, x_i, x_{i+1} and x_{i+2} to generate the two fictitious values at x_{i-1} and x_{i+1}.

For case 2, we use x_{i-1}, x_i, x_{i+1} and x_{i+2} to generate the two fictitious values at x_i and x_{i+2}.

The above approach deals with the above two cases when the interface gets close to x_i or x_{i+1}. In my numerical experiment, I use this approach when $\alpha < 0.1$ for case 1, and $\alpha > 0.9$ for case 2. Otherwise, traditional MIB is deployed.

For case 1, once the fictitious values at x_{i-1} and x_{i+1} are generated, they are plugged into the modified finite difference stencil approximating u_{xx} at the irregular points.

For case 2, once the fictitious values at x_i and x_{i+2} are generated, they are plugged into the modified finite difference stencil approximating u_{xx} at the irregular points.

We compare the condition number of MIB2-SFP to that of stand second order discretization matrix, and illustrate their ratios corresponding to $\alpha = 0.01, 0.02, \cdots, 0.99$ in the Fig.2.7.

The β values are selected as in the last experiment. The figures in Fig.2.7 shows that the ratio shown in Fig.2.5 for each case is not reduced at all. Printing out the coefficients for the rows for irregular points from MIB-SFP shows that they are identical to those of traditional MIB.

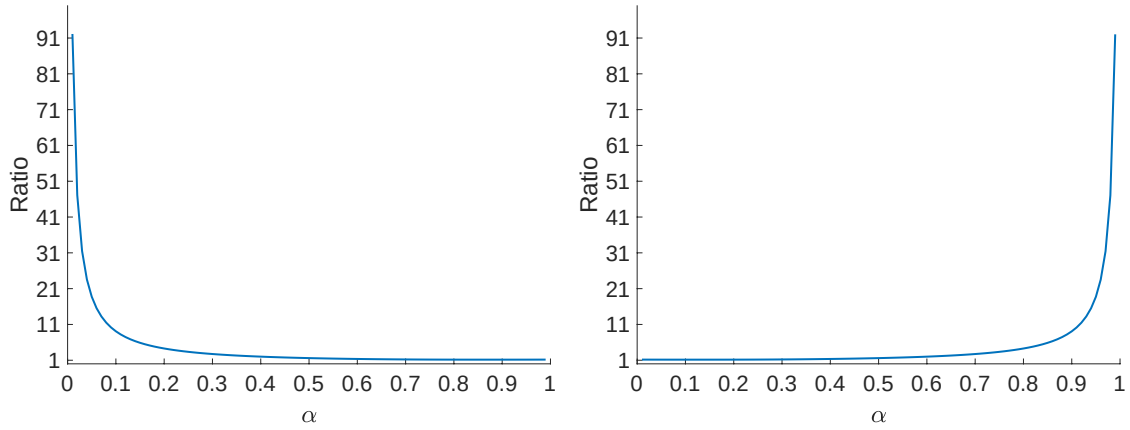

Figure 2.7: The ratios for $\beta^+ >> \beta^-$ (left) and The ratios for $\beta^+ << \beta^-$ (right).

2.2.3 Shifting real point

To remedy the issues in the above two cases in MIB, we propose the shifting real point(SRP) approach.

For case 1, we use $x_{i-2}, x_{i-1}, x_{i+1}$ and x_{i+2} to generate the two fictitious values at x_i and x_{i+1}. Note

that the real point x_i is skipped in this case.

For case 2, we use x_{i-1}, x_i, x_{i+2} and x_{i+3} to generate the two fictitious values at x_i and x_{i+1}. Note that the real point x_{i+1} is skipped in this case.

The above approach deals with the above two cases when the interface gets close to x_i or x_{i+1}.

In my first numerical experiment, I consider using the approach when $\alpha < 0.9$ for case 1, and $\alpha > 0.1$ for case 2. Otherwise, traditional MIB is deployed.

We compare the condition number of MIB2-SRP to that of stand second order discretization matrix, and illustrate their ratios corresponding to $\alpha = 0.01, 0.02, \cdots, 0.99$ in the Fig.2.8.

The β values are selected as in the last experiment. The figures in Fig.2.8 indicates that the ratio shown in Fig.2.6 for each case is significantly reduced. However, the selection for such α is not practical considering that the topology of corner point can be encountered especially for multidimensional problem. So we select different α in the following experiment.

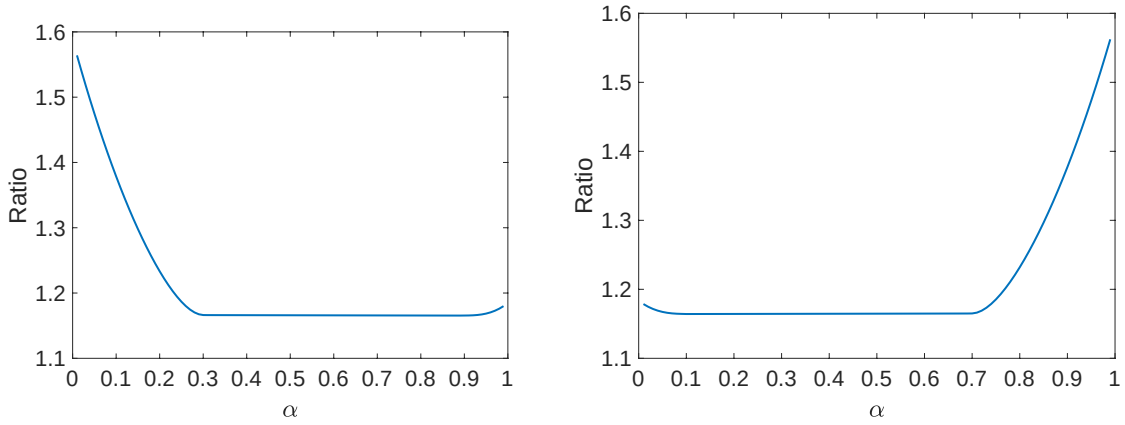

Figure 2.8: The ratios for $\beta^+ >> \beta^-$ (left) and The ratios for $\beta^+ << \beta^-$(right).

In my second numerical experiment, I consider using the approach when $\alpha < 0.1$ for case 1, and $\alpha > 0.9$ for case 2. Otherwise, traditional MIB is deployed.

We compare the condition number of MIB2-SRP to that of stand second order discretization matrix, and illustrate their ratios corresponding to $\alpha = 0.01, 0.02, \cdots, 0.99$ in the Fig.2.9.

The β values are selected as in the last experiment. The figures in Fig.2.9 indicates that the ratio shown in Fig.2.6 for each case is well reduced.

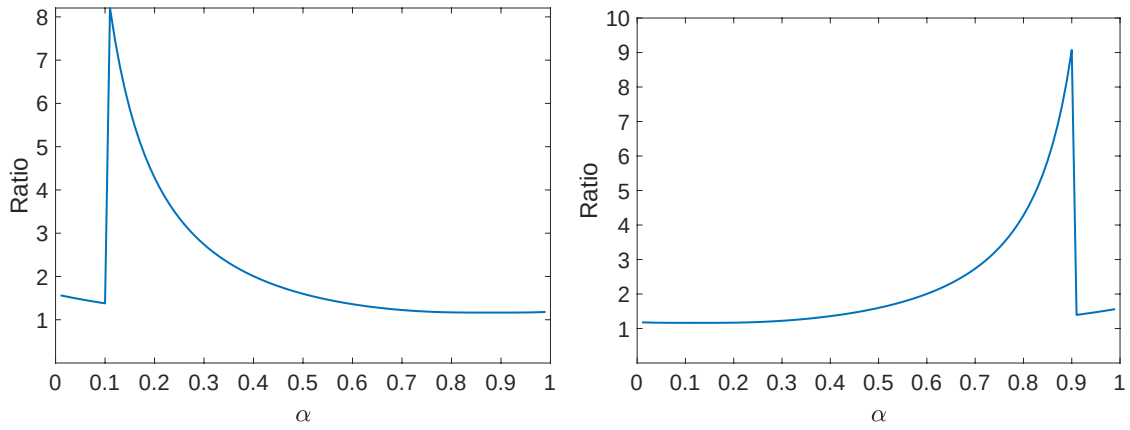

Figure 2.9: The ratios for $\beta^+ >> \beta^-$ (left) and The ratios for $\beta^+ << \beta^-$ (right).

Remark 2.2.1. *The coefficient matrix arising form traditional MIB may suffer from ill-condition issue as specified in the two cases in subsection 2.2.1. Shifting fictitious point and shifting real point are proposed above in order to remedy such issues. As discussed above, shifting real point provides a promising approach in this regard. It is natural to generalize such idea from 1D to multidimensional discussion. With such guideline, fictitious values are be generated for Cartesian direction MIB and Recasting MIB or for high order MIBs in the future studies.*

2.3 Matrix structure from 1D MIB

We would like to carry out further investigation on the coefficient matrix structure arising from MIB discretization for Laplacian operator since it is critical to the effectiveness and efficiency of algorithm. We still concern ourselves with 1D elliptic interface problem with homogeneous interface jump conditions. The reason for considering homogeneous interface conditions is due to the fact that non-homogeneous terms only contribute the right hand side of resulting linear system. To be clear, we are talking about following 1D problem:

$$\beta u_{xx} = f, \tag{2.30}$$

with jump conditions

$$\begin{cases} [u] = 0, x \in \Gamma \\ [\beta u_n] = 0, x \in \Gamma \end{cases}$$

where

$$\beta = \begin{cases} \beta^-, x \in [a, \Gamma] \\ \beta^+, x \in [\Gamma, b] \end{cases}$$

and

$$f(x) = \begin{cases} f^-(x), x \in [a, \Gamma] \\ f^+(x), x \in [\Gamma, b] \end{cases}$$

We start with second order accurate fictitious values using only one point interpolation ($L = 1$) from each side of interface (MIB1), and then we study third order accurate fictitious values using two points interpolation ($L = 2$) on each side of interface (MIB2).

2.3.1 Coefficients derivation for Case $L = 1$

Assume interface point x_θ is between x_0, and x_1. Then $x_\theta = x_0 + \theta h, 0 \leq \theta < 1$, where $h = x_1 - x_0$. For simplicity, just denote $x_\theta = x$. Then we can have interpolation as below

$$u^- = u_0 \frac{x - x_1}{x_0 - x_1} + \hat{u}_1 \frac{x - x_0}{x_1 - x_0} = u_0(1 - \theta) + \hat{u}_1 \theta$$

and

$$u^+ = \hat{u}_0 \frac{x - x_1}{x_0 - x_1} + u_1 \frac{x - x_0}{x_1 - x_0} = \hat{u}_0(1 - \theta) + u_1 \theta$$

with corresponding derivatives

$$u_x^- = u_0 \frac{1}{x_0 - x_1} + \hat{u}_1 \frac{1}{x_1 - x_0} = -u_0 \frac{1}{h} + \hat{u}_1 \frac{1}{h}$$

and

$$u_x^+ = \hat{u}_0 \frac{1}{x_0 - x_1} + u_1 \frac{1}{x_1 - x_0} = -\hat{u}_0 \frac{1}{h} + u_1 \frac{1}{h}$$

From the jump conditions

$$\begin{cases} u^+ - u^- = 0 \\ \beta^+ u_x^+ - \beta^- u_x^- = 0, \end{cases} \tag{2.31}$$

we can get the following two equation system

$$\begin{cases} (1-\theta)\hat{u}_0 - \theta\hat{u}_1 = u_0(1-\theta) - u_1\theta \\ \beta^+\hat{u}_0 + \beta^-\hat{u}_1 = \beta^- u_0 + \beta^+ u_1, \end{cases} \tag{2.32}$$

Solving above equations lead to the expression of $\hat{u}_0$ and $\hat{u}_1$

$$\hat{u}_0 = \frac{\beta^-}{(1-\theta)\beta^- + \theta\beta^+} u_0 + \frac{(\beta^+ - \beta^-)\theta}{(1-\theta)\beta^- + \theta\beta^+} u_1$$

and

$$\hat{u}_1 = \frac{(1-\theta)(\beta^- - \beta^+)}{(1-\theta)\beta^- + \theta\beta^+} u_0 + \frac{\beta^+}{(1-\theta)\beta^- + \theta\beta^+} u_1.$$

2.3.2 Coefficients derivation for Case $L = 2$

For $L = 2$, we need 4 points to construct 2 fictitious points,ie. x_0, x_1, x_2, x_3, where $x_i = x_0 + ih, i = 0, 1, 2, 3$, and $h = x_1 - x_0$. These four points are equally spaced.

Assume interface point x_θ is between x_1, and x_2. Then $x_\theta = x_0 + (1+\theta)h, 0 \le \theta < 1$, where $h = x_1 - x_0$. For simplicity, just denote $x_\theta = x$. Then we can have interpolation as below

$$\begin{aligned} u^- &= u_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + u_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + \hat{u}_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \\ &= u_0 \frac{\theta(\theta - 1)}{2} - u_1(\theta + 1)(\theta - 1) + \hat{u}_2 \frac{\theta(\theta + 1)}{2} \end{aligned}$$

and

$$
\begin{aligned}
u^+ &= \hat{u}_1 \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} - u_2 \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + u_3 \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} \\
&= \hat{u}_1 \frac{(\theta-1)(\theta-2)}{2} - u_2 \theta(\theta-2) + u_3 \frac{\theta(\theta-1)}{2}
\end{aligned}
$$

with corresponding derivatives

$$
\begin{aligned}
u_x^- &= u_0 \frac{2x-(x_1+x_2)}{2h^2} - u_1 \frac{2x-(x_0+x_2)}{h^2} + \hat{u}_2 \frac{2x-(x_0+x_1)}{2h^2} \\
&= u_0 \frac{2\theta-1}{2h} - u_1 \frac{2\theta}{h} + \hat{u}_2 \frac{2\theta+1}{2h}
\end{aligned}
$$

and

$$
\begin{aligned}
u_x^+ &= \hat{u}_1 \frac{2x-(x_2+x_3)}{2h^2} - u_2 \frac{2x-(x_1+x_3)}{h^2} + u_3 \frac{2x-(x_1+x_2)}{2h^2} \\
&= \hat{u}_1 \frac{2\theta-3}{2h} - u_2 \frac{2\theta-2}{h} + u_3 \frac{2\theta-1}{2h}
\end{aligned}
$$

From the jump conditions

$$
\begin{cases}
u^+ - u^- = 0 \\
\beta^+ u_x^+ - \beta^- u_x^- = 0,
\end{cases}
\tag{2.33}
$$

we can get the following two equation system

$$
\begin{aligned}
&\beta^+(2\theta-3)\hat{u}_1 - \beta^-(2\theta+1)\hat{u}_2 \\
&\qquad = \beta^-(2\theta-1)u_0 - \beta^- 4\theta u_1 + u_2(4\theta-4)\beta^+ - (2\theta-1)\beta^+ u_3 = R_1 \\
&(\theta-1)(\theta-2)\hat{u}_1 - \theta(\theta+1)\hat{u}_2 \\
&\qquad = \theta(\theta-1)u_0 - 2(\theta-1)(\theta+1)u_1 + 2\theta(\theta-2)u_2 - \theta(\theta-1)u_3 = R_2.
\end{aligned}
$$

Solving above equations gives rise to the expression of $\hat{u}_0$, and $\hat{u}_1$. with R_1 and R_2.

$$\hat{u}_1 = \frac{R_1(\theta^2+\theta) - \beta^- R_2(2\theta+1)}{\beta^+(2\theta-3)(\theta^2+\theta) - \beta^-(\theta-1)(\theta-2)(2\theta+1)}$$

and

$$\hat{u}_2 = \frac{\beta^+\hat{u}_1(2\theta-3) - R_1}{\beta^-(2\theta+1)}.$$

Plugging R_1, R_2 into above $\hat{u}_1, \hat{u}_2$ expression, we obtain

$$\begin{aligned}\hat{u}_1 = \frac{2\theta^2\beta^-}{D1}u_0 - \frac{2(\theta+1)^2\beta^-}{D1}u_1 + \frac{(\theta^2+\theta)(4\theta-4)\beta^+ - 2(2\theta+1)\theta(\theta-2)\beta^-}{D1}u_2 \\ + \frac{-(2\theta-1)(\theta^2+\theta)\beta^+ + (2\theta+1)\theta(\theta-1)\beta^-}{D1}u_3,\end{aligned}$$

and

$$\begin{aligned}\hat{u}_2 =& \frac{(\beta^-)^2(\theta-1)(\theta-2)(2\theta+1)(2\theta-1) - \beta^+\beta^-(2\theta-3)(2\theta+1)\theta(\theta-1)}{D2}u_0 \\ &+ \frac{\beta^+\beta^-(2\theta-3)(2\theta+1)2(\theta^2-1) - (\beta^-)^2(\theta-1)(\theta-2)(2\theta+1)4\theta}{D2}u_1 \\ &- \frac{2(2\theta+1)(\theta-2)^2}{D2}u_2 + \frac{\beta^+\beta^-2(2\theta+1)(\theta-1)^2}{D2}u_3,\end{aligned}$$

where $D1 = \beta^+(2\theta-3)(\theta^2+\theta) - \beta^-(\theta-1)(\theta-2)(2\theta+1)$, and $D2 = \beta^-(2\theta+1)D1$.

Remark 2.3.1. *The MIB1 and MIB2 can yield to fictitious values of different orders. Due to different number of real function values used to generate fictitious values, it can cause different coefficient matrix structure, which directly impact the condition number and stability of designed numerical scheme. Recently, MIB1 and MIB2 are respectively coupled with ADI method to solve parabolic interface problem [43]. Comparisons between these two approaches in affecting the scheme stability were made. The MIB1 approach can provide better stability despite lower order accuracy.*

2.3.3 Coefficients redistribution

For the elliptic problem

$$\beta u_{xx} = f,$$

with jump conditions

$$[u] = \phi, [\beta u_n] = \varphi.$$

The second order central difference approximation to the Laplacian equation is formulated as

$$\beta_i \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f_i,$$

which is equivalent to

$$u_{i-1} - 2u_i + u_{i+1} = f_i \frac{h^2}{\beta_i}.$$

We will incorporate the coefficients from the two cases mentioned above. We can discuss the first case first.

Case $L = 1$

When u_{i+1} is $\hat{u}_{i+1}$, let's distribute its coefficient to corresponding constructing nodes.

At x_{i-1}, the new coefficient is still 1.

At x_i, the new coefficient is

$$-2 + \frac{(\beta^- - \beta^+)(1-\theta)}{(1-\theta)\beta^- + \theta\beta^+} = -1 - \frac{\beta^+}{(1-\theta)\beta^- + \theta\beta^+}.$$

At x_{i+1}, the new one is

$$\frac{\beta^+}{(1-\theta)\beta^- + \theta\beta^+}.$$

When u_i is $\hat{u}_i$, let's distribute its coefficient to corresponding constructing nodes.

At x_i, the new coefficient is

$$\frac{\beta^-}{(1-\theta)\beta^- + \theta\beta^+}.$$

At x_{i+1}, the new one is

$$-2 + \frac{\theta(\beta^+ - \beta^-)}{(1-\theta)\beta^- + \theta\beta^+} = -1 - \frac{\beta^-}{(1-\theta)\beta^- + \theta\beta^+}.$$

At x_{i+2}, the new one remains to be 1.

Stability analysis

The FD stencil coefficients at x_i is

$$(c_1, c_2, c_3) = (1, -1 - \frac{\beta^+}{(1-\theta)\beta^- + \theta\beta^+}, \frac{\beta^+}{(1-\theta)\beta^- + \theta\beta^+}),$$

through which we can see that

$$|c_2| - c_1 - c_3 = 0.$$

Meanwhile, the FD stencil coefficients at x_{i+1} is

$$(d_1, d_2, d_3) = (\frac{\beta^-}{(1-\theta)\beta^- + \theta\beta^+}, -1 - \frac{\beta^-}{(1-\theta)\beta^- + \theta\beta^+}, 1),$$

which tells us that

$$|d_2| - d_1 - d_3 = 0.$$

In view of the relation between the coefficients, the FD stencil is diagonally dominant. Such diagonal dominance property has the potential to construct stable interface algorithm.

Case $L = 2$

We now proceed to discuss the second case with $L = 2$.

When u_{i+1} is $\hat{u}_{i+1}$, let's distribute its coefficients to corresponding constructing nodes.

At x_{i-1}, the new coefficient is

$$\begin{aligned}
&1+\frac{(\beta^-)^2(\theta-1)(\theta-2)(2\theta+1)(2\theta-1)-\beta^+\beta^-(2\theta-3)(2\theta+1)\theta(\theta-1)}{D2}\\
=&\frac{\beta^-\beta^+[-2\theta(-4\theta^2+4\theta+3)]+(\beta^-)^2[-4\theta^3+10\theta^2-2\theta-4]]}{D2}\\
=&\frac{\beta^-\beta^+[2\theta(2\theta+1)(2\theta-3)]-(\beta^-)^2[2(\theta-1)(\theta-2)(2\theta+1)]}{D2}>0.
\end{aligned}$$

At x_i, the new coefficient is

$$\begin{aligned}
&-2+\frac{\beta^+\beta^-(2\theta-3)(2\theta+1)2(\theta^2-1)-(\beta^-)^2(\theta-1)(\theta-2)(2\theta+1)4\theta}{D2}\\
=&\frac{-2\beta^-\beta^+(2\theta-3)(2\theta+1)(\theta+1)+(\beta^-)^2[2(\theta-1)(\theta-2)(2\theta+1)])}{D2}<0.
\end{aligned}$$

At x_{i+1}, the new one is

$$-\frac{\beta^+\beta^-2(2\theta+1)(\theta-2)^2}{D2}>0.$$

At x_{i+2}, the coefficient is

$$\frac{\beta^+\beta^-2(2\theta+1)(\theta-1)^2}{D2}\leq 0.$$

Remark 2.3.2. *The sum of the four coefficient is equal to one. However, due to the non-positive coefficient for position x_{i+2}, the diagonal dominance property can not be guaranteed.*

When u_i is $\hat{u}_i$, let's distribute its coefficient to corresponding constructing nodes.

At x_{i-1}, the new coefficient is

$$\frac{2\theta^2\beta^-}{D1}\leq 0.$$

At x_i, the new coefficient is

$$-\frac{2(\theta+1)^2\beta^-}{D1}>0.$$

At x_{i+1}, the new one is

$$\begin{aligned} -2+&\frac{(\theta^2+\theta)(4\theta-4)\beta^+ - 2(2\theta+1)\theta(\theta-2)\beta^-}{D1} \\ =&\frac{2\beta^+\theta(\theta+1) - 2\beta^-(\theta-2)(2\theta+1)}{D1} < 0. \end{aligned}$$

At x_{i+2}, the new one is

$$\begin{aligned} 1+&\frac{-(2\theta-1)(\theta^2+\theta)\beta^+ + (2\theta+1)\theta(\theta-1)\beta^-}{D1} \\ =&\frac{-2\beta^+\theta(\theta+1) + 2\beta^-(2\theta+1)(\theta-1)}{D1} > 0. \end{aligned}$$

Here

$$D1 = \beta^+(2\theta-3)(\theta^2+\theta) - \beta^-(\theta-1)(\theta-2)(2\theta+1) < 0,$$

and

$$D2 = \beta^-(2\theta+1)D1 < 0.$$

Remark 2.3.3. *The sum of the four coefficients is equal to one. However, due to the non-positive coefficient for position x_{i-1}, the diagonal dominance property can not be guaranteed.*

2.4 Alternative to jump-corrected differences

In the above section, two interpolation approaches are used to generate fictitious values. The accuracy of fictitious values depends on interpolation points, i.e. $L = 1$ (MIB1) or $L = 2$ (MIB2). The interpolation error of each type of fictitious values is of order $O(h^2)$ and $O(h^3)$ respectively for MIB1 and MIB2 in light of Lagrange interpolation remainder. A brief analysis can be found in [86]. To be clear, we should have following error estimation. Assuming that the interface Γ lies in between grid points x_i and x_{i+1}, then fictitious values $\hat{u}(x_i)$ and $\hat{u}(x_{i+1})$are related to real function values at these two grid points as following for each case.

L=2 :

$$\hat{u}(x_i) = u(x_i) + O(h^2)$$

$$\hat{u}(x_{i+1}) = u(x_{i+1}) + O(h^2).$$

L=3 :

$$\hat{u}(x_i) = u(x_i) + O(h^3)$$

$$\hat{u}(x_{i+1}) = u(x_{i+1}) + O(h^3).$$

Traditional MIB2 replaces real function values with fictitious values when the standard second order central differences cut through the interface. For the two irregular points x_i and x_{i+1}, we have following stencil with the same assumption that $u \in C^4[x_j - h, \alpha) \cap C^4(\alpha, x_{j+1} + h]$ and $x_i \in \Omega^-$ and $x_{i+1} \in \Omega^+$, where derivatives extend continuously up to α in the jump-corrected difference.

$$\begin{aligned}
u_{xx}(x_i) =& \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + O(h^2) \\
=& \frac{\hat{u}(x_{i+1}) + O(h^3) - 2u(x_i) + u(x_{i-1})}{h^2} + O(h^2) \\
=& \frac{\hat{u}(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + O(h), \qquad (2.34) \\
u_{xx}(x_{i+1}) =& \frac{u(x_{i+2}) - 2u(x_{i+1}) + u(x_i)}{h^2} \\
=& \frac{\hat{u}(x_{i-1}) + O(h^3) - 2u(x_i) + u(x_{i+1})}{h^2} + O(h^2) \\
=& \frac{\hat{u}(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} + O(h), . \qquad (2.35)
\end{aligned}$$

As a consequence, (2.34) and (2.35) can provide locally first order approximation to Laplacian operator with the first order truncation error.

The approximation (2.34) and (2.35) can be equivalently reformulated as

$$\begin{aligned} u_{xx}(x_i) \approx & \frac{\hat{u}(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} \\ \approx & \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \frac{1}{h^2}(\hat{u}(x_{i+1}) - u(x_{i+1})), \end{aligned} \tag{2.36}$$

$$\begin{aligned} u_{xx}(x_{i+1}) \approx & \frac{\hat{u}(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} \\ \approx & \frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} + \frac{1}{h^2}(\hat{u}(x_{i-1}) - u(x_{i-1})) \end{aligned} \tag{2.37}$$

Similar to the augmentation approach in above AMIB method based on jump-corrected differences, we may consider introduce the terms in the parenthesis in (2.34) and (2.35) as auxiliary variables. Suppose the fictitious values of $\hat{u}(x_i)$ or $\hat{u}(x_{i+1})$ are approximated with four real functions at four grid points adjacent to the interface point α, it is simply represented as below for $\hat{u}(x_{i+1})$

$$\hat{u}(x_{i+1}) = \sum_{j=1}^{4} w_j u(x_{i-2+j}) + w_5\phi(\alpha) + w_6\psi(\alpha) \tag{2.38}$$

Let $Q = \hat{u}_{i+1} - u_{i+1}$, then following equation is true:

$$-\sum_{j=1}^{4} w_j u(x_{i-2+j}) + u_{i+1} + Q = w_5\phi(\alpha) + w_6\psi(\alpha).$$

To be concise, it can also have a vector form as

$$CU + Q = \Phi.$$

With the Laplacian operator approximated by

$$AU + BQ = F,$$

augmented system below can be constructed after taking corrected fictitious values as auxiliary

variables,

$$KW = R, \tag{2.39}$$

where

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

Remark 2.4.1. *The augmented system* (2.39) *based on correcting fictitious values gives an alternative to the one by introducing jump derivative as auxiliary variables. It allow AMIB to has relatively smaller number of auxiliary variables totally compared to corrected difference case. For each interface point, two fictitious values need to be corrected along the Cartesian direction while at least two derivative jumps have to be introduced for corrected differences. However, the fictitious values can be repeatedly used from two directions. This explains why corrected fictitious value approach owns smaller number of auxiliary variables. The numerical performance of the two methods will be compared. The current approach is based on corrected fictitious value(CFP), and the previous one is by corrected finite difference(CFD). The development of AMIB method in solving problems, we mainly adopt the CFD approach, and we call it as AMIB method. We would also compare the performance of the two different approaches for AMIB, when we will use AMIB-CFD or AMIB -CFP to indicate which approach is being used.*

2.5 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 in this section were carried out on personal laptop Lenovo Thinkpad L440 with 8.00 RAM and Intel Core i5-4200M @ 2.50GHz.

2.5.1 Numerical accuracy

In this section, several comparisons on numerical accuracy between MIB method and augmented MIB method are studied through example 1 to example 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 also 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

$$u(x, y) = \begin{cases} x^2 + y^2, & r \le 0.5, \\ \frac{1}{4}(1 - \frac{1}{8b} - \frac{1}{b}) + (\frac{r^4}{4} + r^2)/b, & \text{otherwise}, \end{cases} \tag{2.40}$$

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. 2.10. The numerical errors from different meshes are listed in Table 2.1, 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

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

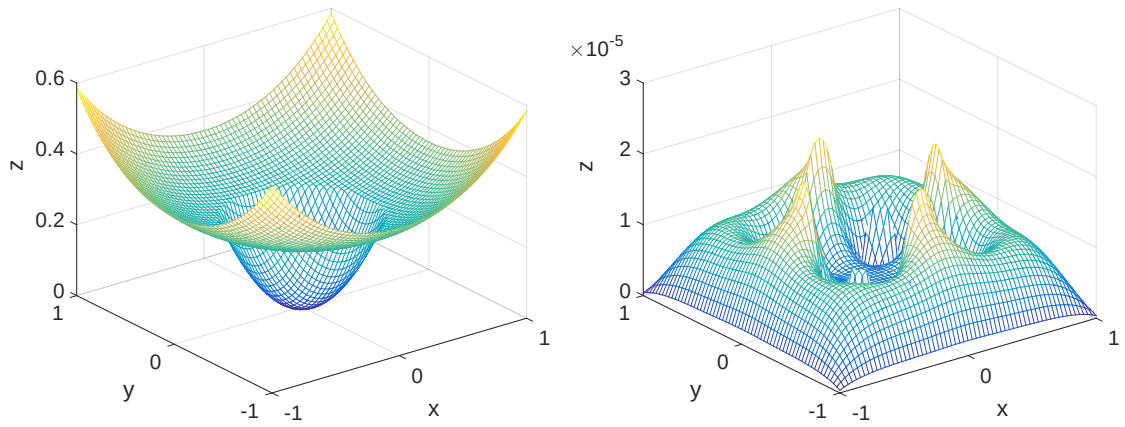

Figure 2.10: The computed solution (left), and error (right) in Example 1 on a mesh with $n = 64$.

Table 2.1: 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	

A computed solution is plotted with $n = 64$ in Fig. 2.11. 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 2.2.

Table 2.2: 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	

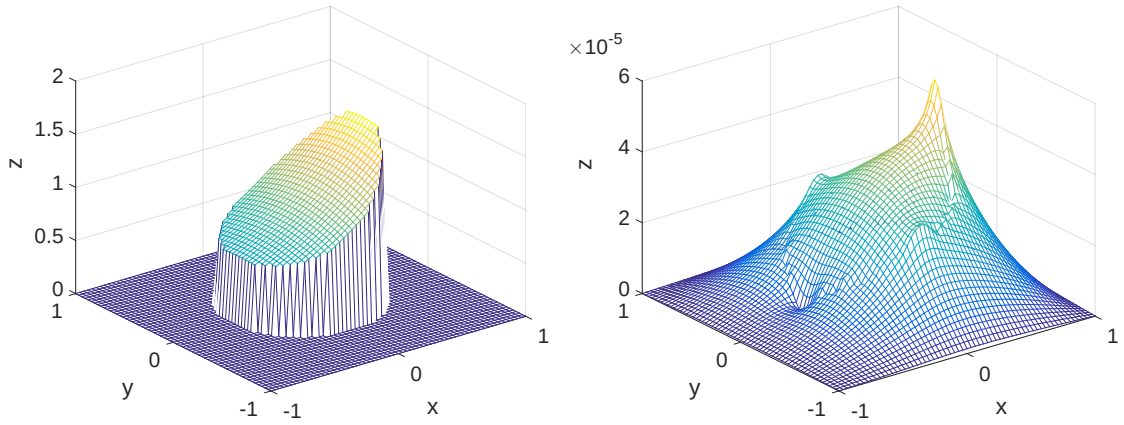

Figure 2.11: The computed solution (left), and error (right) in Example 2 on a mesh with $n = 64$. Here β is equal to 1 throughout the domain.

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

$$(\frac{x}{18/27})^2 + (\frac{y}{10/27})^2 = 1$$

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 the AMIB method achieves second order of convergence in both cases. Furthermore, the iteration number of the AMIB computation becomes slightly larger when β^- is increased from 10 to 1000. Nevertheless, the increment slope of the iteration number with respect to the mesh size n is still very small. This demonstrates that the AMIB scheme performs robustly in dealing with interface problems of high β ratio contrast.

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

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

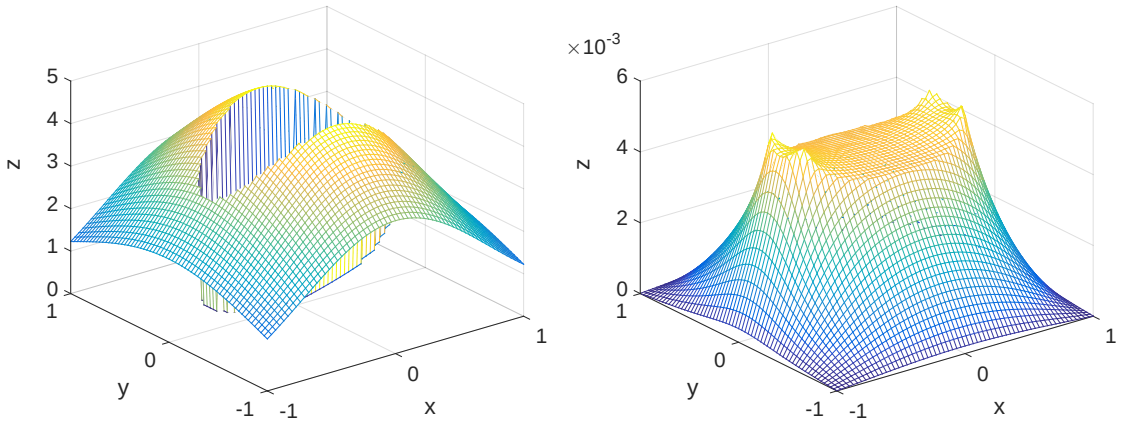

Figure 2.12: The computed solution (left), and error (right) in Example 3A on a mesh with $n = 64$. Here $\beta^+ = 1, \beta^- = 10$.

Table 2.3: 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]	5.272E-3	–	2.774E-3	–	27
[128, 128]	1.169E-3	2.17	6.030E-4	2.20	28
[256, 256]	2.765E-4	2.08	1.420E-4	2.10	33
[512, 512]	7.341E-5	1.94	3.772E-5	1.91	35
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	5.506E-3	–	2.833E-3	–	
[128, 128]	1.167E-3	2.23	5.998E-4	2.24	
[256, 256]	2.877E-4	2.02	1.481E-4	2.02	
[512, 512]	7.506E-5	1.93	3.873E-5	1.94	
$[n_x, n_y]$	AMIB-minus				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[64, 64]	2.901E-4	–	1.456E-4	–	32
[128, 128]	1.884E-4	0.62	9.275E-5	0.65	32
[256, 256]	3.055E-5	2.62	1.535E-5	2.59	36
[512, 512]	7.500E-6	2.03	3.922E-6	1.97	41

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 $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The parameter k is set to be 2. The source terms associated

Table 2.4: 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]	0.362	–	0.219	–	35
[128, 128]	0.111	1.71	6.665E-2	1.72	37
[256, 256]	2.633E-2	2.08	1.577E-2	2.08	50
[512, 512]	6.976E-3	1.92	4.170E-3	1.92	57
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	0.415	–	0.243	–	
[128, 128]	0.110	1.92	6.481E-2	1.91	
[256, 256]	2.799E-2	1.98	1.657E-2	1.97	
[512, 512]	7.203E-3	1.95	4.289E-3	1.95	
$[n_x, n_y]$	AMIB-MINUS				
	L_∞		L_2		
[64, 64]	1.768E-2	–	1.066E-2	–	51
[128, 128]	8.336E-3	1.084	4.992E-3	1.09	54
[256, 256]	1.283E-3	2.70	7.644E-4	2.70	74
[512, 512]	3.524E-4	1.86	2.098E-4	1.85	100

with the two solutions are

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

Two sets of β^+ and β^- are chosen. Table 2.5 and table 2.6 show the numerical results with $\beta^+ = 100, \beta^- = 1$(Case A) and $\beta^+ = 1, \beta^- = 100$(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 jump ratio $\rho = \beta^+/\beta^-$ change.

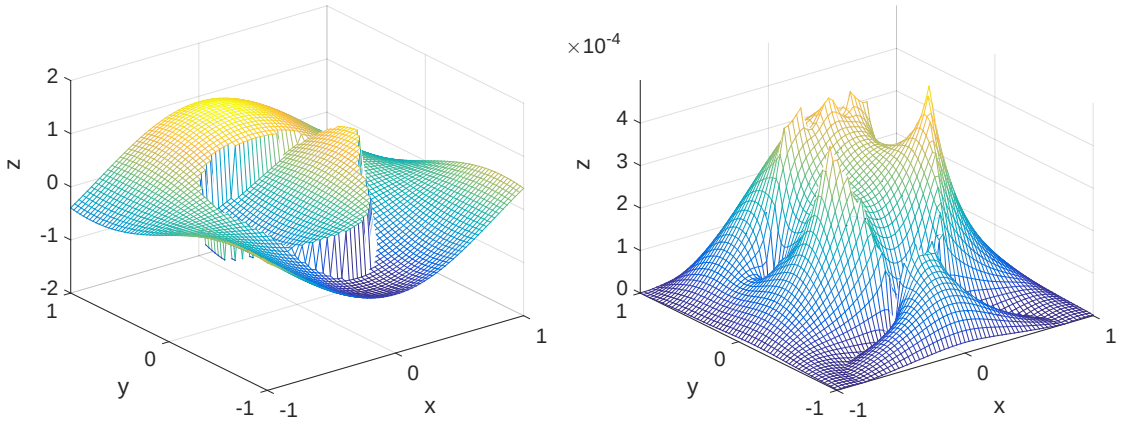

Figure 2.13: The computed solution (left), and error (right) in Example 4 (case A) with $\beta^+ = 100$ and $\beta^- = 1$ on a mesh with $n = 64$.

Table 2.5: Example 4A – Numerical error analysis. Here $\beta^+ = 100, \beta^- = 1$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[64, 64]	4.937E-4	–	1.772E-4	–	39
[128, 128]	1.081E-4	2.19	4.074E-5	2.12	45
[256, 256]	2.503E-5	2.11	9.272E-6	2.13	74
[512, 512]	5.935E-6	2.08	2.293E-6	2.02	86
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	4.756E-4	–	1.833E-4	–	
[128, 128]	9.037E-5	2.39	3.541E-5	2.37	
[256, 256]	2.492E-5	1.86	9.017E-6	1.97	
[512, 512]	5.880E-6	2.08	2.234E-6	2.01	

Table 2.6: Example 4B – Numerical error analysis. Here $\beta^+ = 1, \beta^- = 100$.

$[n_x, n_y]$	AMIB				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[64, 64]	1.421E-2	–	8.242E-3	–	35
[128, 128]	4.133E-3	2.19	2.404E-3	1.78	42
[256, 256]	8.991E-4	2.20	5.183E-4	2.21	51
[512, 512]	2.212E-4	2.02	1.275E-4	2.02	63
$[n_x, n_y]$	MIB				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	1.339E-2	–	7.512E-3	–	
[128, 128]	3.979E-3	1.75	2.276E-3	1.72	
[256, 256]	8.349E-4	2.25	4.763E-4	2.26	
[512, 512]	2.058E-4	2.03	1.178E-4	2.02	
$[n_x, n_y]$	AMIB-MINUS				
	L_∞		L_2		
	Error	Order	Error	Order	
[64, 64]	6.610E-3	–	3.491E-3	–	51
[128, 128]	7.906E-4	3.06	3.577E-4	3.29	69
[256, 256]	1.056E-4	2.90	3.732E-5	3.26	83
[512, 512]	2.727E-5	1.95	9.765E-6	1.93	101

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

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

on a square domain $[-1.00001, 1.00001] \times [-1.00001, 1.00001]$. The diffusion coefficient β is

defined as $\beta^+ = 1000$ and $\beta^- = 1$ for case A, while it is defined as $\beta^+ = 1$ and $\beta^- = 1000$ for case B. The AMIB errors are slightly larger than those of the MIB, while a second order of convergence is still achieved for the AMIB scheme for both cases.

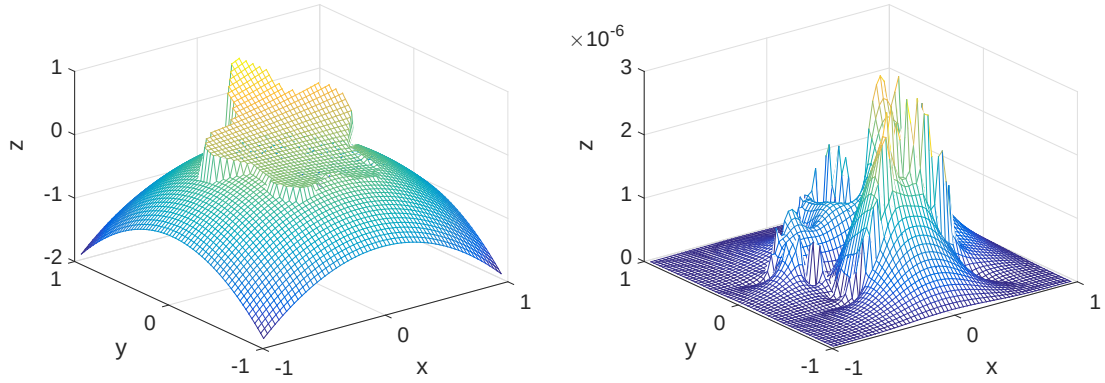

Figure 2.14: The computed solution (left), and error (right) in Example 5 with $\beta^+ = 1000$ and $\beta^- = 1$ on a mesh of $n = 64$.

Table 2.7: Example 5A – Numerical error analysis. Here $\beta^+ = 1000, \beta^- = 1$.

$[n_x, n_y]$	AMIB					
	L_∞		L_2		iter no.	condition no.
	Error	Order	Error	Order		
[32, 32]	2.670E-5	–	5.209E-6	–	33	1.95E7
[64, 64]	2.962E-6	3.17	5.902E-7	3.14	50	2.59E8
[128, 128]	6.221E-7	2.25	1.259E-7	2.22	62	1.22E10
[256, 256]	1.647E-7	1.92	3.839E-8	1.71	81	1.92E12
$[n_x, n_y]$	MIB					
	L_∞		L_2			
	Error	Order	Error	Order		
[32, 32]	1.508E-5	–	2.263E-6	–		
[64, 64]	2.5453E-6	2.57	3.625E-7	2.64		
[128, 128]	3.186E-7	3.00	7.042E-8	2.36		
[256, 256]	6.801E-8	2.23	1.463E-8	2.27		
$[n_x, n_y]$	AMIB-MINUS					
	L_∞		L_2		iter no.	condition no.
	Error	Order	Error	Order		
[32, 32]	4.350E-4	–	1.219E-4	–	27	
[64, 64]	1.041E-4	2.06	2.756E-5	2.15	33	
[128, 128]	3.331E-5	1.64	7.476E-6	1.88	62	
[256, 256]	8.686E-6	1.94	1.870E-6	2.00	71	

2.5.2 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

Table 2.8: Example 5B – Numerical error analysis. Here $\beta^+ = 1, \beta^- = 1000$.

$[n_x, n_y]$	AMIB					
	L_∞		L_2		iter no.	condition no.
	Error	Order	Error	Order		
[32, 32]	1.620E-3	–	8.911E-4	–	26	3.07E6
[64, 64]	8.723E-4	0.89	5.131E-4	0.80	37	2.85E8
[128, 128]	9.240E-4	-0.08	5.675E-4	-0.15	54	2.70E10
[256, 256]	3.160E-4	1.54	1.938E-4	1.55	74	8.88E11
$[n_x, n_y]$	MIB					
	L_∞		L_2			
	Error	Order	Error	Order		
[32, 32]	1.711E-3	–	8.831E-4	–		
[64, 64]	7.875E-4	1.12	4.469E-4	0.98		
[128, 128]	9.348E-4	-0.25	5.649E-4	-0.34		
[256, 256]	2.944E-4	1.67	1.790E-4	1.66		
$[n_x, n_y]$	AMIB-MINUS					
	L_∞		L_2		iter no.	condition no.
	Error	Order	Error	Order		
[32, 32]	0.15	–	9.381E-2	–	39	
[64, 64]	2.029E-3	0.89	1.174E-3	6.32	50	
[128, 128]	4.179E-3	-0.08	2.508E-3	-1.10	91	
[256, 256]	1.279E-3	1.54	7.647E-4	1.71	138	

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 2.9 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.

Table 2.9: Reconstructed Cartesian derivative jumps for Example 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]	1.162E-2	–	2.072E-2	–	0.215	–	0.302	–
[64, 64]	4.525E-3	1.36	7.046e-3	1.56	8.171e-2	1.40	0.151	1.00
[128, 128]	1.013E-3	2.16	1.461E-3	1.56	3.311E-2	1.30	5.980	1.34
[256, 256]	2.676E-4	1.92	3.628E-4	2.01	1.654E-2	1.00	3.173E-2	0.91

Remark 2.5.1. *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 Table* 1 – 6*, 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 lease first order accurate from Table 2.9.

2.5.3 Computational efficiency

Here we carry out a few comparisons between our augmented MIB method (AMIB) with regular MIB method (MIB) in term of the computational efficiency. With the same parameter set in the proceeding five examples, we conclude that our method significantly saves the computational time after comparing the CPU time of AMIB to that of MIB. Figure 2.15 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.

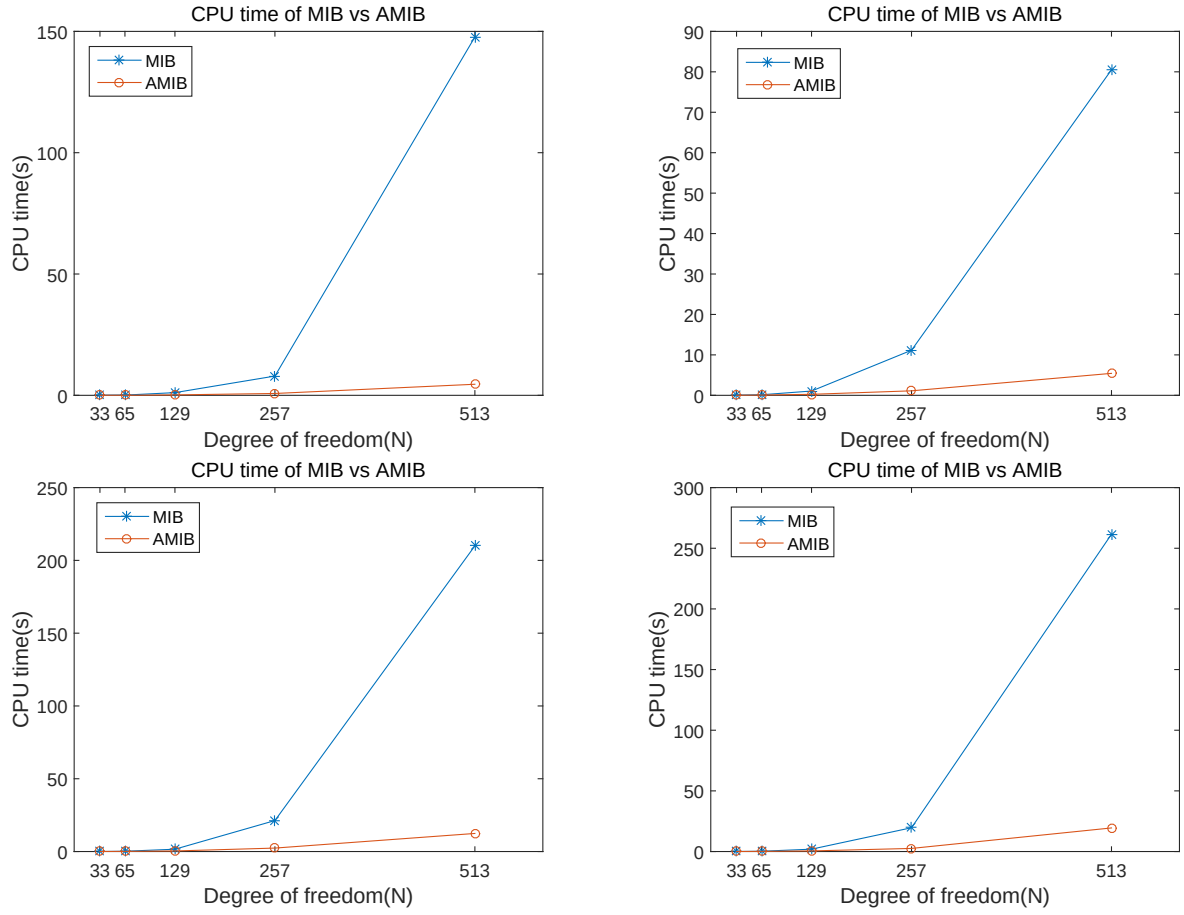

Figure 2.15: CPU time in seconds for both MIB and AMIB methods. The horizontal line is the degree of freedom in one dimension, i.e., $n + 1$.

In our previous examples, the iteration numbers needed by the biconjugate gradient method in solving the Schur complemented system (2.27) 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. 2.16. 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. 2.16 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.

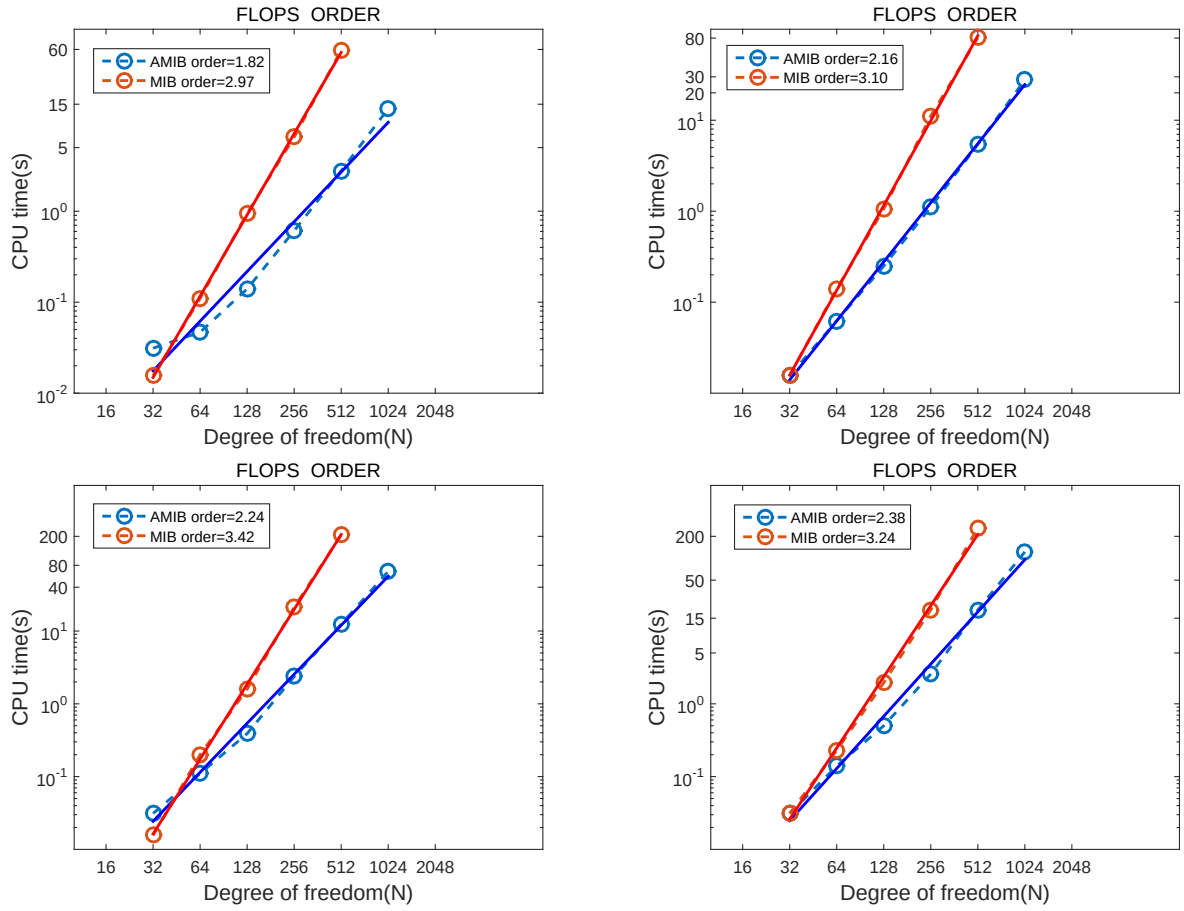

Figure 2.16: Flops order in CPU time is examined for several examples.

2.5.4 Comparison of AMIB-CFD and AMIB-CFP

Here, we would like to compare the performance two second order AMIB methods arising from corrected finite difference and corrected fictitious value based approach for interface treatment. The numerical accuracy and iteration numbers are recorded in table 2.10 for comparison purpose. The five star shaped interface as in example 5 is used. Besides, the fictitious value generation is according to iteration reducing strategy discussed in this dissertation. The same iteration stopping criteria is applied for fair comparison of the two methods.

In the first example, the analytical solution with $k = 5$ and domain setting are as in example 4. It can be observed from table 2.10 that the performance of AMIB-CFD and AMIB-CFP are basically the same since the numerical accuracy and iteration number are all very close.

Table 2.10: Numerical error analysis. Here $\beta^+ = 100, \beta^- = 1$.

$[n_x, n_y]$	AMIB-CFD				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[64, 64]	1.050E-2	–	3.078E-3	–	31
[128, 128]	4.074E-3	1.37	7.711E-4	2.00	54
[256, 256]	7.060E-4	2.53	1.934E-4	2.00	58
[512, 512]	1.739E-4	2.02	4.657E-5	2.05	64
[1024, 1024]	4.340E-5	2.00	1.166E-5	2.00	64
$[n_x, n_y]$	AMIB-CFP				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[64, 64]	1.021E-2	–	3.020E-3	–	31
[128, 128]	2.712E-3	1.91	7.639E-4	1.98	51
[256, 256]	6.987E-4	1.96	1.918E-4	1.99	60
[512, 512]	1.718E-4	2.02	4.610E-5	2.06	61
[1024, 1024]	4.293E-5	2.00	1.155E-5	2.00	68

The second example is equipped with the same domain yet the analytical solution set as

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

Numerical results in table 2.11 show the close performance of AMIB-CFD and AMIB-CFP again. The above two examples demonstrate that second order AMIB-CFD and AMIB-CFP can all be efficient solver for elliptic interface problem due to their equal efficiency and numerical accuracy. Relatively, it is easier to code for AMIB-CFP, while AMIB-CFD involves more coding details.

Table 2.11: Numerical error analysis. Here $\beta^+ = 1, \beta^- = 100$.

$[n_x, n_y]$	AMIB-CFD				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[64, 64]	1.767E-3	–	9.535E-4	–	35
[128, 128]	6.002E-4	1.56	3.367E-4	1.50	52
[256, 256]	2.335E-4	1.36	1.337E-4	1.33	65
[512, 512]	6.605E-5	1.82	3.787E-5	1.82	63
[1024, 1024]	1.655E-5	2.00	9.503E-6	1.99	86
$[n_x, n_y]$	AMIB-CFP				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[64, 64]	1.811E-3	–	9.770E-4	–	34
[128, 128]	6.207E-4	1.54	3.485E-4	1.49	50
[256, 256]	2.353E-4	1.40	1.348E-4	1.37	65
[512, 512]	6.580E-5	1.84	3.771E-5	1.84	65
[1024, 1024]	1.657E-5	1.99	9.512E-6	1.99	86

We will further comparison the fourth order versions of the two method to investigate the merits and drawback of the AMIB methods in the future.

CHAPTER 3

AMIB FOR POISSON BOUNDARY VALUE PROBLEM

In the above chapter, great details have been given to demonstrate the mechanism of AMIB method through the second order fast solver for elliptic interface problem. In addition to the above second order fast solution to the elliptic interface problem, this chapter will show an approach based on AMIB method to solve Poisson boundary value problem (BVP) efficiently.

3.1 Preliminary

Poisson's equation as a simple partial differential equation (PDE) plays a crucial role in the study of electrostatics, mechanical engineering and theoretical physics. Its numerical solution draws extensive attentions as it has many generalized forms [58] in practical utility. Efficient algorithm for the Poisson solution is desired, otherwise it may take the majority of computation time. Fast Fourier transform is one among many that are used to design efficient Poisson solver. Several FFT packages including FISHPACK [66] and FFTW [28] have been developed, yet only yielding second order accuracy. Besides FFT, other popular fast Poisson solvers include multigrid method, cyclic reduction, and FACR algorithm. When concerned with high order fast Poisson solver, compact difference scheme and Spectral methods are often combined with FFT or multigrid in order to achieve both high order accuracy and efficiency. Quite many works have been done in each category of compact differences [29, 41, 68, 68, 69] and spectral methods [3, 9, 10, 63]. High order central difference schemes have never been applied in fast Poisson solvers, even though the standard fast Poisson solver is based on the second order central difference. This is perhaps because high order central differences have long stencils, so that fictitious points outside boundaries are inevitable. A fast Poisson solver based on high order central difference schemes

enjoys several potential benefits in comparing with its counterpart based on compact finite difference or spectral methods.

3.2 Theory and algorithm

In this section, we aim at the high order fast solution of Poisson boundary value problem (BVP). The high order accuracy is to be achieved by high order central difference schemes applied on Poisson's equation in both two dimensions (2D) and three dimensions (3D). The governing equation is

$$\triangle u = f(\vec{x}), \quad \vec{x} \in \Omega \tag{3.1}$$

where $\vec{x} = (x_1, x_2, \cdots, x_d)$ is the spatial vector, Ω is the domain of interest, and $\triangle$ is the Laplacian operator defined by

$$\triangle = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}.$$

In this dissertation, d is chosen to be 2 or 3.

Since the periodic boundary condition can be simply handled in the Fourier transform method, we will only focus on three standard boundary conditions, i.e.

1. Dirichlet boundary condition:

$$u = \phi_j \text{ on } \Gamma_j,$$

2. Neumann boundary condition:

$$\frac{\partial u}{\partial n} = \phi_j \text{ on } \Gamma_j,$$

3. Robin boundary condition:

$$ku + \frac{\partial u}{\partial n} = \phi_j \text{ on } \Gamma_j,$$

where Γ_j is one of the boundaries of the computational domain as shown in Figure 3.1, and $\frac{\partial}{\partial n}$ denotes the outward normal derivative.

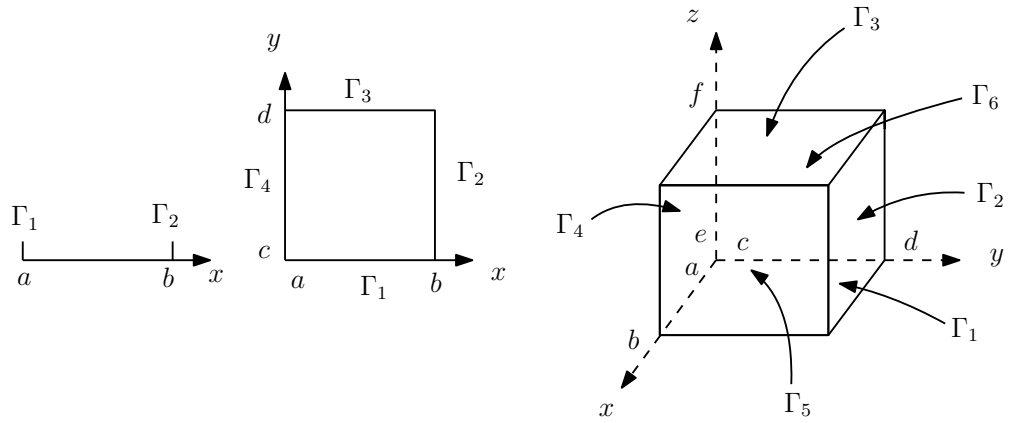

Figure 3.1: Illustration of boundary notation in 1D, 2D and 3D.

We are concerned with the high order fast solution to Poisson's equation on domain $\Omega = [a, b] \times [c, d] \times [e, f]$, which is partitioned into M, N, P equally spaced intervals in $x-$, $y-$ and $z-$directions such that the mesh sizes are $h_x = (b - a)/M, h_y = (d - c)/N$, and $h_z = (f - e)/P$ respectively. If only a 2D problem is studied, we just neglect the partition in the $z-$direction. Therefore, the grid node coordinates are defined as

$$\begin{aligned} x_i =& a + ih_x,\ y_j = c + jh_y,\ z_p = e + ph_z, \\ & i = 0, \cdots, M,\ j = 0, \cdots, N,\ p = 0, \cdots, P. \end{aligned} \tag{3.2}$$

Central finite differences are utilized for approximating the Laplacian operator in this work. As the partial derivatives are defined in a tensor product style for separable geometries, we decompose approximation to Laplacian operator in several directions separately. Without loss of generality, high order central differences to second order partial derivative with respect to x are adopted for demonstration. For simplicity, we just concern ourselves with $u = u(x)$.
The fourth order central difference to $u_{xx}(x_i)$ holds with truncation error $O(h_x^4)$:

$$u_{xx}(x_i) \approx \frac{1}{h_x^2}[-\frac{1}{12}u(x_{i-2}) + \frac{4}{3}u(x_{i-1}) - \frac{5}{2}u(x_i) + \frac{4}{3}u(x_{i+1}) - \frac{1}{12}u(x_{i+2})], \tag{3.3}$$

and sixth order central difference takes forms as below with truncation error $O(h_x^6)$:

$$u_{xx}(x_i) \approx \frac{1}{h_x^2}[\frac{1}{90}u(x_{i-3}) - \frac{3}{20}u(x_{i-2}) + \frac{3}{2}u(x_{i-1}) - \frac{49}{18}u(x_i) + \frac{3}{2}u(x_{i+1}) - \frac{3}{20}u(x_{i+2}) + \frac{1}{90}u(x_{i+3})], \tag{3.4}$$

along with the standard eighth order central difference with truncation error $O(h_x^8)$:

$$u_{xx}(x_i) \approx \frac{1}{h_x^2}[-\frac{1}{560}u(x_{i-4}) + \frac{8}{315}u(x_{i-3}) - \frac{1}{5}u(x_{i-2}) + \frac{8}{5}u(x_{i-1}) - \frac{205}{70}u(x_i) + \frac{8}{5}u(x_{i+1}) - \frac{1}{5}u(x_{i+2}) + \frac{8}{315}u(x_{i+3}) - \frac{1}{560}u(x_{i+4})]. \tag{3.5}$$

Likewise, the fourth, sixth and eighth order finite differences could be defined for u_{yy}, u_{zz} in a similar fashion.

3.2.1 Fast Poisson solver

In the literature, many high order fast Poisson solvers have been designed based on the compact differences. In this work, we will take advantage of the aforementioned central differences to construct fast Poisson solvers. As a translation invariant application of high order central finite differences will inevitably use points outside the boundary, we extend the domain by exterior grids in addition to (3.2) defined by

$$\begin{aligned} x_{-i} &= a - ih_x, x_{M+i} = b + ih_x, \quad i = 1, 2, 3, \\ y_{-j} &= c - jh_y, y_{N+j} = d + jh_y, \quad j = 1, 2, 3, \\ z_{-p} &= e - ph_z, z_{P+p} = f + ph_z, \quad p = 1, 2, 3. \end{aligned}$$

Ghost values can then be defined on these grids for application of high order central difference schemes.

In the first step of constructing our algorithm, we will consider a simple Poisson problem with Dirichlet zero boundary condition. We note that for this simple Poisson problem, a direct

application of central differences with FFT fast Poisson solver still cannot maintain high order accuracy. A remedy will be offered so that a FFT fast Poisson solver can be well constructed for the Dirichlet zero boundary. The generalization to real boundary conditions will be addressed in the next section. For simplicity, we will only consider a one dimension (1D) Poisson's equation in this section to illustrate the ideas.

Fourth order central difference scheme

Consider the fourth order finite difference (FD) scheme for a 1D Poisson's equation:

$$u_{xx}(x) = f, \tag{3.6}$$

subject to Dirichlet zero boundary condition. In order to apply FFT directly, we further assume anti-symmetry property up to fourth order for ghost values outside the domain

$$u(x_0) = 0, \quad u(x_{-1}) = -u(x_1), \tag{3.7}$$

$$u(x_M) = 0, \quad u(x_{M+1}) = -u(x_{M-1}). \tag{3.8}$$

Here, we denote u_i as the approximation to the exact value of $u(x_i)$ and f_i as $f(x_i)$:

$$u_i \approx u(x_i), \quad i = -1, 0, \cdots, M+1,$$
$$f_i = f(x_i), \quad i = 1, 2, \cdots, M-1.$$

To achieve higher order of accuracy, a higher order anti-symmetry property is assumed. Examples with orders being six and eight are given in Appendix. We note that the anti-symmetry properties like (3.7) and (3.8) may not hold for a general Dirichlet zero boundary. A systematic procedure to bypass this constraint will be constructed later.

The fourth order FD discretization (3.3) for equation (3.6) gives an equation system:

$$\frac{1}{h_x^2}(-\frac{1}{12}u_{i-2} + \frac{4}{3}u_{i-1} - \frac{5}{2}u_i + \frac{4}{3}u_{i+1} - \frac{1}{12}u_{i+2}) = f_i, \text{ for } i = 1, \cdots, M-1. \tag{3.9}$$

Due to the assumption of anti-symmetry of solution across the boundary, system (3.9) could be reduced to a matrix-vector form

$$AU = F, \tag{3.10}$$

where $A \in \mathbb{R}^{M-1,M-1}$, $U \in \mathbb{R}^{M-1}$ and $F \in \mathbb{R}^{M-1}$ are given by

$$A = \frac{1}{h_x^2}\begin{pmatrix} -\frac{5}{2}+\frac{1}{12} & \frac{4}{3} & -\frac{1}{12} & 0 & 0 & 0 & \cdots & 0 \\ \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} & -\frac{1}{12} & 0 & 0 & \cdots & 0 \\ -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} & -\frac{1}{12} & 0 & \cdots & 0 \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ 0 & & & -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} & -\frac{1}{12} \\ \vdots & & & & -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} \\ 0 & 0 & \cdots & & & -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2}+\frac{1}{12} \end{pmatrix}, \tag{3.11}$$

together with

$$U = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{M-1} \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{M-1} \end{pmatrix}.$$

Note that the entries at the corners of matrix A are due to the anti-symmetry assumption.

Now we focus on fast solver for the solution of above equation system. Based on discrete Sine transform, the solution could be expanded as

$$u_j = \sum_{l=1}^{M-1} \hat{u}_l \sin(\frac{l j\pi}{M}), -1 \le j \le M+1. \tag{3.12}$$

where

$$\hat{u}_l = \frac{2}{M} \sum_{j=1}^{M-1} u_j \sin(\frac{l j\pi}{M}), l = 1, 2, \cdots, M-1. \tag{3.13}$$

Besides, the source term f_j has discrete Sine transform

$$f_j = \sum_{l=1}^{M-1} \hat{f}_l \sin(\frac{l j\pi}{M}), 1 \le j \le M-1, \tag{3.14}$$

where

$$\hat{f}_l = \frac{2}{M} \sum_{j=1}^{M-1} f_j \sin(\frac{l j\pi}{M}), l = 1, 2, \cdots, M-1. \tag{3.15}$$

Plugging (3.12) and (3.14) into (3.9), we obtain

$$\begin{aligned}
&\sum_{l=1}^{M-1} \hat{u}_l \frac{1}{h_x^2}[-\frac{1}{12}\sin(\frac{l(j-2)\pi}{M}) + \frac{4}{3}\sin(\frac{l(j-1)\pi}{M}) - \frac{5}{2}\sin(\frac{l j\pi}{M}) \\
&\qquad + \frac{4}{3}\sin(\frac{l(j+1)\pi}{M}) - \frac{1}{12}\sin(\frac{l(j+2)\pi}{M})] \\
&= \sum_{l=1}^{M-1} \hat{u}_l \frac{1}{h_x^2}[-\frac{1}{3}(\cos(\frac{l\pi}{M}) - 1)(\cos(\frac{l\pi}{M}) - 7)]\sin(\frac{l j\pi}{M}) \\
&= \sum_{l=1}^{M-1} \hat{f}_l \sin(\frac{l j\pi}{M}), \quad \text{for } \; j = 1, 2, \cdots, M-1.
\end{aligned}$$

The above equation system is well-posed, and hence by equating the coefficients of the same terms, we get the relation

$$\hat{u}_l = \frac{\hat{f}_l}{\lambda_l}, \quad \text{for } \; l = 1, 2, \cdots, M-1,$$

where $\lambda_l = -\frac{1}{3h_x^2}[\cos(\frac{l\pi}{M}) - 1][\cos(\frac{l\pi}{M}) - 7]$.

A procedure for the FFT solution can be described as below:

1. Compute Sine transform for $f(x_j)$ via inverse fast Sine transform(IFST).

   $\hat{f}_l = \frac{2}{M} \sum_{j=1}^{M-1} f(x_j) \sin(\frac{ljp\pi}{M})$, for $l = 1, \cdots, M-1$.

2. $\hat{u}_l = \frac{\hat{f}_l}{\lambda_l}$, for $l = 1, 2, \cdots, M-1$, where $\lambda_l = -\frac{1}{3h_x^2}[\cos(\frac{l\pi}{M}) - 1][\cos(\frac{l\pi}{M}) - 7]$.

3. Compute u_j via fast Sine transform (FST): $u_j = \sum_{l=1}^{M-1} \hat{u}_l \sin(\frac{ljp\pi}{M})$, for $j = 1, \cdots, M-1$.

Numerical validation

We consider two 1D Poisson problems with Dirichlet zero boundary condition, with one satisfying anti-symmetry and the other one not. Two numerical implementations of fourth order central difference will be considered. The first one employs the FFT procedure presented above. The second one further assumes that the analytical values on ghost nodes outside the boundaries are known. Then one can move such ghost values to the right-side hand of algebraic system, and solves the system by the LU decomposition.

Example 1 in 1D. Consider Poisson's equation

$$u_{xx} = 16\pi^2 \sin(4\pi x) \text{ for } x \in [0, 1]$$

with solution $u(x) = -\sin(4\pi x)$ satisfying anti-symmetry. The numerical results are shown in Table 3.1. It is obvious that both FFT and LU methods attain the same precision and fourth order of convergence is achieved in both cases. In algebraic computations, the FFT method is much faster than the LU decomposition.

Example 2 in 1D. We next consider Poisson's equation

$$u_{xx} = 16\pi^2 \cos(4\pi x) \text{ for } x \in [0, 1]$$

Table 3.1: Numerical errors of the FFT and LU methods for Example 1 in 1D. Here N denotes the number of partitions on the given interval.

N	FFT		LU	
	Error	Order	Error	Order
32	2.607E-4	–	2.607E-4	–
64	1.646E-5	3.99	1.646E-5	3.99
128	1.031E-6	4.00	1.031E-6	4.00
256	6.450E-8	4.00	6.450E-8	4.00
512	4.032E-9	4.00	4.032E-9	4.00

with solution $u(x) = -\cos(4\pi x) + 1$ satisfying the homogeneous Dirichlet boundary condition $u = 0$ at $x = 0$ and $x = 1$. Nevertheless, the anti-symmetry property is not held. Table 3.2 lists the errors of the FFT and LU methods for fourth order central difference scheme. It is observed that the FFT method only provides second order accuracy while the LU gives fourth order accuracy. This demonstrates the importance of anti-symmetry of solution when using the above FFT Poisson solver.

Table 3.2: Numerical errors of the FFT and LU methods for Example 2 in 1D.

N	FFT		LU	
	Error	Order	Error	Order
32	1.321E-2	–	5.229E-4	–
64	3.235E-3	2.03	3.294E-5	3.99
128	8.046E-4	2.00	2.063E-6	4.00
256	2.009E-4	2.00	1.290E-7	4.00
512	5.020E-5	2.00	8.061E-9	4.00

Remark 3.2.1. *So far, a fourth order fast Poisson solver via FST has been formulated for the Poisson problem with Dirichlet zero boundary condition. A crucial assumption here is that one layer of ghost value outside of domain satisfies the anti-symmetry property across the boundary. Higher order fast Poisson solvers can be constructed in this manner. In Appendix, by assuming two and three layers of ghost values satisfying anti-symmetry respectively, sixth and eighth order central difference schemes can be formulated by the same FFT procedure.*

Remark 3.2.2. *For the two dimensional (2D) or three dimensional (3D) Poisson's equation, we continue to assume the homogeneous Dirichlet boundary and anti-symmetry property in each dimension as described in 1D. By the tensor product approach, the discretization and FFT procedure for high dimensional problems can be similarly completed.*

Remark 3.2.3. *The aforementioned fast Poisson solvers only apply to the case of homogeneous Dirichlet condition with anti-symmetric solution across the boundary. However, in practice the anti-symmetry property may not hold or is not easily to be verified for homogeneous Dirichlet boundary conditions. Moreover, general Poisson problems face more complex boundary conditions. Hence, it is necessary to explore fast Poisson solvers for general solutions. We will construct a systematic procedure to bypass the anti-symmetric issue.*

3.2.2 Immersed boundary problem and corrected differences

In the previous section, fast Poisson solvers based on high order central differences have been constructed with the assumption of anti-symmetry of solution across the boundary. However, such assumption is generally not valid for Poisson problems. This is essentially the obstacle that prevents the application of FFT Poisson solver to high order central difference schemes. To resolve this difficulty, an immersed boundary problem will be introduced in this dissertation. To be more clear, the original domain is embedded in a larger domain such that fictitious solution equals to 0 on the extended domain. The aforementioned fast Poisson solvers can then be facilitated, since the zero solution automatically meets the requirement of anti-symmetry on the boundary of extended domain. However, the entire solution consisting of original solution and fictitious solution is discontinuous on the original boundary or on the interface of the immersed boundary problem. The direct FFT computation of discontinuous solutions will suffer Gibbs oscillations. Instead, corrected differences will be employed in the proposed augmented matched interface and boundary (AMIB) method, so that central differences are preserved with some correction terms being used to restore the desired accuracy.

Immersed boundary

With a uniform grid spacing in each dimension, we embed the original domain Ω in a larger rectangle or cube B such that the distance between interior boundary $\partial\Omega$ and the exterior boundary ∂B of B is s multiples of h, where s is a positive integer. The value of s varies according to the desired accuracy of central difference approximation for efficiency concern. Technically,

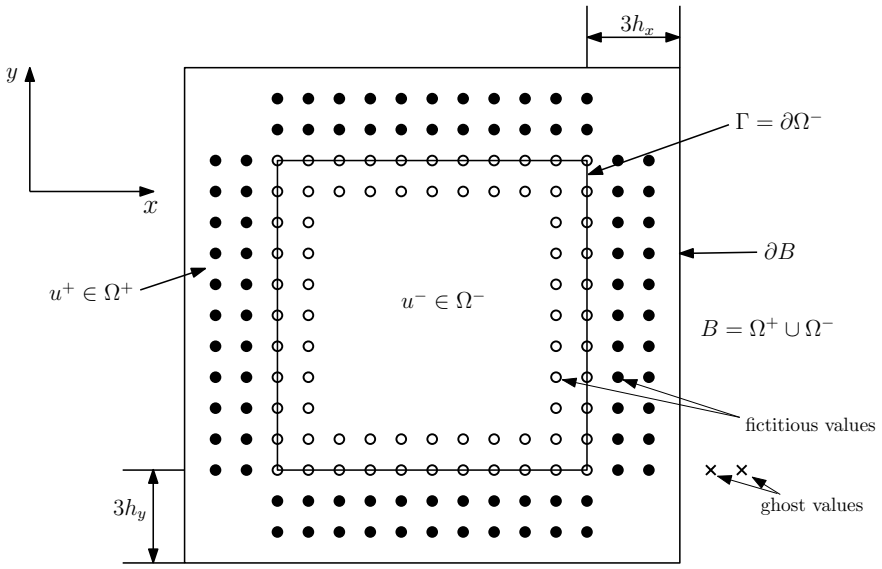

Figure 3.2: A demonstration of immersed boundary problem for fourth order central difference scheme with two interior and exterior layers of *fictitious values* around the interface $\Gamma = \partial\Omega^-$. Here filled and open circles stand for fictitious values to be used in approximation of jump quantities from corrected differences. Meanwhile, *ghost values* (depicted as crossings) are zero padding values beyond the expanded domain satisfying anti-symmetry.

just a minimum integer number s is needed to carry out the approximation of desired accuracy, otherwise extra computational efforts will be wasted. Fig. 3.2 illustrates the case in 2D for fourth order central difference scheme with $s = 3$. As a consequence, the grid node coordinates are redefined as below in the domain B

$$\begin{aligned} x_i =& a + (i - s)h_x,\ y_j = c + (j - s)h_y,\ z_k = e + (k - s)h_z, \\ & i = 0, \cdots, M + 3s,\ j = 0, \cdots, N + 3s,\ k = 0, \cdots, P + 3s, \end{aligned} \tag{3.16}$$

where h_x, h_y, h_z are identically defined as (3.2).

From now on, we denote the original domain Ω as Ω^- and the extended subdomain as Ω^+, yielding the whole domain $B = \Omega^+ \cup \Omega^-$. Correspondingly, the solution in the original domain is defined as u^- in Ω^-, and solution in the extended domain Ω^+ as u^+. Source term is defined as

$f^- \in \Omega^-$ and $f^+ \in \Omega^+$. Then the immersed boundary problem can be modeled as

$$\triangle u(\vec{x}) = f, \quad \vec{x} \in B \tag{3.17}$$

with source term being

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

Note that as the artificial solution $u(\vec{x}) = 0$ for $\vec{x} \in \Omega^+$, $f(\vec{x}) = 0$ for $\vec{x} \in \Omega^+$. Further beyond the exterior boundary ∂B, u is considered as 0 such that the anti-symmetry property is now satisfied on boundary ∂B. In other words, ghost values beyond ∂B are set to be 0. This facilitates the utilization of high order central differences and FFT Poisson solvers. However, this remodeled problem introduces an interface $\Gamma = \partial\Omega^-$, across which discontinuity of solution and source term emerges.

It is well known that central difference approximations will suffer accuracy reduction near the interface Γ when the approximation involves nodes from both sides of Γ. Grid nodes in such situation are called irregular points, otherwise regular points. To be more clear, we take the case of fourth order central difference for instance. In general, assume the interface intersects the grid line $y = y_j$ at some point α between x_i and x_{i+1}. As the fourth order central difference for nodes x_{i-1}, x_i, x_{i+1} and x_{i+2} will use the node information on the other side of the interface, these four nodes are called irregular points. For sixth and eighth order central difference with even wider stencil, more irregular points are defined along the grid line accordingly. In the literature, many methods have been proposed to correct finite difference approximation for interface problems [6, 42, 45, 72, 82, 85]. In this study, we will follow the AMIB method [24] that uses corrected finite differences [72] to tackle such irregular points problem. The corrected differences incorporate several interface jump quantities to compensate the discontinuity, while central differences are still used for regular points.

Corrected finite differences for Laplacian

Taking advantage that central difference approximation to Laplacian is carried out dimension by dimension, we again just focus on the derivation of corrected differences in 1D, by taking $u = u(x)$ and $h = h_x$. Assume the smoothness of the piecewise function $u \in C^{l+1}$ in each side of the interface. Function values at two nodes could be related by corrected Taylor expansion with some jump conditions included, from which corrected finite difference of different orders could be derived. In the first place, the following two important results (3.18) and (3.19) can be cited from *Lemma 1* and *Remark 2* of [72]. Here one considers the corrected Taylor expansion for two adjacent points on each side of the interface:

Assume the interface $x = \alpha$ lies in between grid points x_j and x_{j+1}, then it follows that

$$u(x_{j+1}) = \sum_{k=0}^{l} \frac{h^k}{k!} u^{(k)}(x_j) + \sum_{k=0}^{l} \frac{(h^+)^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{3.18}$$

and

$$u(x_j) = \sum_{k=0}^{l} \frac{(-h)^k}{k!} u^{(k)}(x_{j+1}) - \sum_{k=0}^{l} \frac{(h^-)^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{3.19}$$

where $h^- = x_j - \alpha$, $h^+ = x_{j+1} - \alpha$, and $[u^{(m)}]_\alpha = \lim\limits_{x \to \alpha^+} u^{(m)}(x) - \lim\limits_{x \to \alpha^-} u^{(m)}(x)$.

In the present study, the above results have to be extended for high order central difference schemes which involve longer stencils. In particular, besides x_j and x_{j+1}, jump corrected Taylor expansions are needed for nodes further away from interface $x = \alpha$.

$$u(x_{j+j_2}) = \sum_{k=0}^{l} \frac{[(j_2 - j_1)h]^k}{k!} u^{(k)}(x_{j-j_1}) + \sum_{k=0}^{l} \frac{[(j_2 - 1)h + h^+]^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{3.20}$$

and

$$u(x_{j-j_1}) = \sum_{k=0}^{l} \frac{[(j_1 - j_2)h]^k}{k!} u^{(k)}(x_{j+j_2}) - \sum_{k=0}^{l} \frac{[-j_1 h + h^-]^k}{k!} [u^{(k)}]|_{x=\alpha} + O(h^{l+1}) \tag{3.21}$$

where integers $j_2 \geq 1$ and $j_1 \geq 0$ are index increments while j denotes the index location. Here we still assume that $x_j \leq \alpha \leq x_{j+1}$, $h^- = x_j - \alpha$, and $h^+ = x_{j+1} - \alpha$.

Equations (3.20) and (3.21) are generalized form of (3.18) and (3.19). The proof of these two equations is just a minor modification of the proof for (3.18) and (3.19). Thus it is omitted here. For details, refer to [72]. When $j_2 = 1$, and $j_1 = 0$, equations (3.20) and (3.21) are consistent with (3.18) and (3.19). With aid of formula (3.20) and (3.21), fourth order corrected differences can be obtained as below:

Theorem 1. Corrected fourth differences. Let $x_j \leq \alpha < x_{j+1}, h^- = x_j - \alpha$, 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 $O(h^4)$:

$$u_{xx}(x_{j-1}) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_{j-3}) + \frac{4}{3}u(x_{j-2}) - \frac{5}{2}u(x_{j-1}) + \frac{4}{3}u(x_j) - \frac{1}{12}u(x_{j+1})] + \frac{1}{12h^2}\sum_{m=0}^{5}\frac{(h^+)^m}{m!}[u^{(m)}], \tag{3.22}$$

$$u_{xx}(x_j) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_{j-2}) + \frac{4}{3}u(x_{j-1}) - \frac{5}{2}u(x_j) + \frac{4}{3}u(x_{j+1}) - \frac{1}{12}u(x_{j+2})] - \frac{4}{3h^2}\sum_{m=0}^{5}\frac{(h^+)^m}{m!}[u^{(m)}] + \frac{1}{12h^2}\sum_{m=0}^{5}\frac{(h+h^+)^m}{m!}[u^{(m)}], \tag{3.23}$$

$$u_{xx}(x_{j+1}) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_{j-1}) + \frac{4}{3}u(x_j) - \frac{5}{2}u(x_{j+1}) + \frac{4}{3}u(x_{j+2}) - \frac{1}{12}u(x_{j+3})] + \frac{4}{3h^2}\sum_{m=0}^{5}\frac{(h^-)^m}{m!}[u^{(m)}] - \frac{1}{12h^2}\sum_{m=0}^{5}\frac{(h^- - h)^m}{m!}[u^{(m)}], \tag{3.24}$$

$$u_{xx}(x_{j+2}) \approx \frac{1}{h^2}[-\frac{1}{12}u(x_j) + \frac{4}{3}u(x_{j+1}) - \frac{5}{2}u(x_{j+2}) + \frac{4}{3}u(x_{j+3}) - \frac{1}{12}u(x_{j+4})] - \frac{1}{12h^2}\sum_{m=0}^{5}\frac{(h^-)^m}{m!}[u^{(m)}]. \tag{3.25}$$

Remark 3.2.4. *We note that one may just take m ranging from 0 to 4 to achieve the third order accuracy locally at irregular points. Since the fourth order central difference is applied at all regular points except for at a few irregular points, a global fourth order of convergence can still be maintained. Thus, for a better efficiency, we will take* $m = 0, \ldots, 4$ *in the present study.*

Remark 3.2.5. *The fourth order corrected differences have been illustrated above in 1D manner.*

In higher dimensional study, such operators are applicable in each dimension once intersections of grid line with interface are encountered. Besides the fourth order case, sixth or eighth order corrected difference can be derived with the details described in the Appendix.

Remark 3.2.6. *The local distances $h^- = x_j - \alpha$ and $h^+ = x_{j+1} - \alpha$ defined in the corrected differences should be determined before approximation. In our expanded domain, the interface Γ aligns with the grid line. Thus either h^- or h^+ is equal to zero. For convenience, we consider all grid nodes on the interface as inside the interface. Therefore, there are only two situations in terms of the local distance. For the interface close to the left (bottom) boundary, $h^- = -h$ and $h^+ = 0$. We name this situation as type I. On the other hand, for the interface close to the right (upper) boundary, $h^- = 0$ and $h^+ = h$. We name this situation as type II.*

Reconstructing the Cartesian derivative jumps

So far we have established corrected differences with some derivative jumps $[u^{(m)}]$ undetermined. In the proposed AMIB method, such jump quantities will be calculated through enforcing the given boundary conditions. A systematic MIB boundary closure [77, 78, 81, 83] has been developed in the literature for implementing high order central difference schemes with any combination of Dirichlet, Neumann, and Robin conditions. In the present study, this MIB method will be employed to enforce boundary conditions and correspondingly generate a sufficient number of fictitious values outside $\partial\Omega^-$ or interface Γ. Unlike the original MIB [77, 78, 81, 83], one will not directly modify the discrete Laplacian approximation by using fictitious values. Instead, the fictitious values will be utilized for calculating the derivative jumps $[u^{(m)}]$, as in the AMIB method [24].

A minimal number of fictitious nodes is determined as follows for the high order central difference schemes. As mentioned above, it is sufficient to use one less order locally in the corrected jump quantities to guarantee the high orders. To ensure a third, fifth, and seventh order convergence locally for the irregular points for fourth, sixth, and eighth corrected differences respectively, polynomials of degree 4, 6, and 8 are needed for approximation. As a consequence,

2, 3, and 4 fictitious values outside Γ are combined with 3, 4, and 5 real function values on the other side of the interface to give desired polynomials respectively for each case. Fig. 3.2 shows two layers of fictitious values needed on each side of the interface Γ for the fourth order corrected differences. We note that the interface Γ is regarded as in Ω^- in the present notation.

Fictitious values formulation

To reconstruct the Cartesian derivative jumps, we first review how fictitious values are calculated in the the MIB method [81]. A systematic procedure has been developed in [81] for handling Dirichlet, Neumann, and Robin boundary conditions. This procedure iteratively enforces the boundary condition to determine fictitious values as smooth extension of solution around the interface Γ. The MIB formulation of fictitious values is explained in great details for various boundary conditions in [81].

In the present study, it is sufficient to illustrate the MIB formulation of fictitious values in the case of Robin boundary condition in 1D,

$$ku + \frac{\partial u}{\partial n} = ku - \frac{\partial u}{\partial x} = \phi \quad \text{on } \Gamma_1. \tag{3.26}$$

We generate fictitious values iteratively by repeatedly matching the boundary conditions across the boundary. Referring to Figure 3, to generate M fictitious values f_i for $i = 1, 2, \cdots, M$ outside the domain, $L + 1$ function values u_j for $j = 0, 1, \cdots, L$ inside the domain are to be used. Firstly,

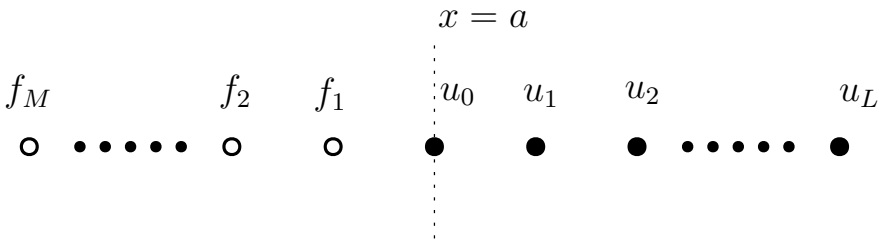

Figure 3.3: An illustration of fictitious values near the left boundary.

one fictitious value f_1 is constructed according to approximation to boundary (3.26):

$$ku_0 + C_{2,1}^{(1)} f_1 + \sum_{i=2}^{L+2} C_{2,i}^{(1)} u_{i-2} = \phi, \tag{3.27}$$

where $C_{2,i}^{(1)}$ are finite difference (FD) weights in approximating first derivative at $x = a$ by using a stencil $\{f_1, u_0, \cdots, u_L\}$. The notation 2 in first subscript of $C_{2,i}^{(1)}$ is because u_0 is the second point in the FD stencil. As a result, unknown f_1 can be represented as the combination of u_i for $i = 0, 1, \cdots, L$ and ϕ. Taking f_1 as known, we may further seek to determine fictitious value f_2 using one iterative step

$$ku_0 + C_{3,1}^{(1)} f_2 + C_{3,2}^{(1)} f_1 + \sum_{i=3}^{L+3} C_{3,i}^{(1)} u_{i-3} = \phi,$$

where $C_{3,i}^{(1)}$ are FD weights to approximate first derivative at $x = a$ using a stencil $\{f_2, f_1, u_0, u_1, \cdots, u_L\}$. In this manner, we can further obtain representation of fictitious values $f_3, f_4, \cdots, f_M$ in terms of u_i for $i = 0, 1, \cdots, L$. Even though it is flexible to choose the value for L, we expect to set L so that high-order accuracy is achieved in boundary implementation. As suggested in [81], the condition $L + 1 = 2M$ is sufficient to guarantee the order requirement when M fictitious values are needed in high order central differences near the boundary. For example, in the fourth order central difference, two layers of fictitious values are needed. We may adopt $L = 3$ for formulation of these two fictitious values.

The MIB implementation of Neumann boundary condition can be similarly formulated. For the Dirichlet condition, we will follow the idea in [81] to generate a numerical boundary condition of the form

$$\frac{\partial^2 u}{\partial n^2} = \phi, \tag{3.28}$$

where n denotes the x- or y-direction. Such boundary condition was manually derived based on the given Poisson equation and Dirichlet boundary conditions [26]. For example, suppose Dirichlet condition is given as $u(x, y) = \phi_4(y)$ on Γ_4. Taking second order derivative on $\phi_4(y)$ with

respect to y leads to $\frac{d^2\phi_4(y)}{dy^2}$, which is essentially $\frac{\partial^2 u(x,y)}{\partial y^2}$ along boundary Γ_4. Assuming that the governing equation $\triangle u = f$ is valid on the boundary Γ_4, we can derive a new boundary condition

$$\frac{\partial^2 u}{\partial x^2}\Big|_{\Gamma_4} = \left[f - \frac{\partial^2 u}{\partial y^2}\right]_{\Gamma_4} = \left[f - \frac{d^2\phi_4(y)}{dy^2}\right]_{\Gamma_4} \triangleq \phi(x). \tag{3.29}$$

Finally, the fictitious values calculated from Dirichlet, Neumann, and Robin boundary conditions can be rewritten in to a general form in 2D. Denote $\hat{u}_{i,j}$ as a fictitious value at (x_i, y_j). We have

$$\hat{u}_{i,j} = \sum_{(x_I, y_J)\in\mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0\phi, \tag{3.30}$$

which is a linear combination of functions values on a chosen set $\mathbb{S}_{i,j}$, and known boundary data. The node set $\mathbb{S}_{i,j}$ can be determined by the choice of M and L before calculating the MIB weights $W_{I,J}$. Such a MIB formulation can be easily extended to 3D.

Approximation to derivative jumps

The jump quantity at $x = \alpha$ is defined as

$$[\frac{\partial^m u}{\partial x^m}]|_{x=\alpha} = \lim_{x\to\alpha^+} \frac{\partial^m u}{\partial x^m} - \lim_{x\to\alpha^-} \frac{\partial^m u}{\partial x^m} \tag{3.31}$$

To approximate jump quantities in the fourth order corrected differences, we consider using polynomials of degree 4 from each side of the interface Γ. Hence, derivative jumps can be approximated is following formulation:

$$\begin{aligned}[\frac{\partial^k u}{\partial x^k}]|_{x=\alpha} \approx\ & (w^k_{i-1,j}\hat{u}_{i-1,j} + w^k_{i,j}\hat{u}_{i,j} + \sum_{l=1}^{3} w^k_{i+l,j}u_{i+l,j}) \\ & - (\sum_{l=1}^{3} w^k_{i-3+l,j}u_{i-3+l,j} + w^k_{i+1,j}\hat{u}_{i+1,j} + w^k_{i+2,j}\hat{u}_{i+2,j}),\end{aligned} \tag{3.32}$$

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 $O(h^3)$. In current immersed boundary problem, as the external function is $u^+ = 0$, we may use exact zero value derivative for one side derivative limit. As a result, one of the two parenthesis term may vanish in practical use. For example, in the case shown in Fig.3.4, the term in the second parenthesis is equal to zero.

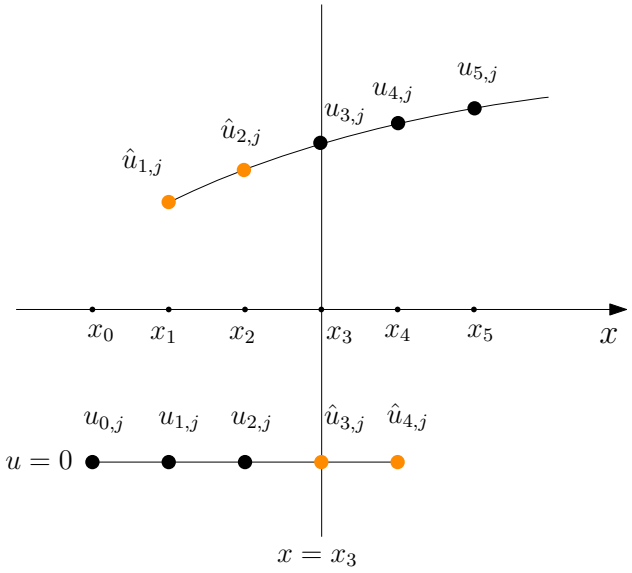

Figure 3.4: An illustration of jumps approximation via two Lagrange polynomials from two sides of the interface $x = x_3$ for the fourth order approximation. The black dots stand for the real values while the yellow dots denote the fictitious values.

Plugging fictitious values representation (3.30) into (3.32) gives equivalent forms:

$$\sum_{(x_I,y_J)\in\mathbb{S}} C_{I,J}u_{I,J} + [\frac{\partial^m u}{\partial x^m}] = C_0\phi, \tag{3.33}$$

where $C_{I,J}$ is the corresponding weights of function value $u_{I,J}$ in approximation to jump quantity $[\frac{\partial^m u}{\partial x^m}]$, and $\mathbb{S}$ is the set of needed grid points in (3.33). Equivalent form for jump quantities $[\frac{\partial^m u}{\partial y^m}]$ could be derived in a similar fashion in the $y-$ direction.

The quantities $u_{I,J}$ are unknowns to be solved. In the AMIB method, we take $[\frac{\partial^m u}{\partial x^m}]$ and $[\frac{\partial^m u}{\partial y^m}], m = 0, 1, \cdots, 4$ as auxiliary variables. The numerical approximation to $[\frac{\partial^m u}{\partial x^m}]$ and $[\frac{\partial^m u}{\partial y^m}]$

exemplified in (3.33) can be rewritten in one matrix-vector form:

$$CU + IQ = \Phi, \tag{3.34}$$

where C stands for the weights coefficients matrix, U represents the function variables, I is the identity matrix, Q is the introduced auxiliary variables on all boundary nodes, and Φ is from the known boundary quantities ϕ.

3.2.3 Augmented system

With correction terms defined as auxiliary variables along with the function values, one augmented system can be constructed. Now we take approximation to 2D Poisson's equation for demonstration. Let $U_{i,j}$ indicate the discrete solution of $u(x_i, y_j)$ at (x_i, y_j). Based on the corrected difference analysis, the problem (3.1) is discretized as

$$L_h U_{i,j} + C_{i,j} = f_{i,j}, \;\; 1 \le i \le M + 2s - 1, \;\; 1 \le j \le N + 2s - 1, \tag{3.35}$$

where $C_{i,j}$ is the correction term, and $L_h U_{i,j}$ is the standard high order central difference scheme to Laplacian with a degree of freedom $N1 = (M + 2s - 1) \times (N + 2s - 1)$. The s has the same meaning in (3.16).

Below we take the fourth order study for instance. The parameter s is equal to 2. Then $N1 = (M + 3) \times (N + 3)$. Note that correction term only exists at irregular points but vanishes at regular points. In other words, $C_{i,j}$ exists for approximation at the each two layers of grid points surrounding the interface. Otherwise, $C_{i,j}$ vanishes for the other interior grid points inside the interface as shown in Figure 3.2. Via Eq. (3.35), a relation between numerical solution and auxiliary variables in x- or y-partial derivatives is obtained as

$$AU + BQ = F, \tag{3.36}$$

where A is a symmetric, diagonally dominant matrix of dimension $N1$ by $N1$, B is a matrix of

coefficients from correction terms and F is a vector with entries being f_{ij}. We note that the inversion of matrix A can be facilitated with the above FFT algorithm. Variable U stands for the numerical solution to the immersed boundary problem and Q is a vector of introduced 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 the intersection of interface and grid lines. The total number of auxiliary variables, denoted as $N2$, is 5 times that of all intersection points, while the latter is proportional to M or N. That means $N2$ is one dimension smaller than $N1$. Matrix B is a sparse matrix of dimension $N1 \times N2$, and it is composed of the coefficients from the correction terms in the corrected differences.

Moreover, inspired by formula (3.32), approximation to jump quantities could be formulated as

$$CU + IQ = \Phi, \tag{3.37}$$

where notations in (3.37) have the same meaning in (3.32). Matrix C is a sparse matrix of dimension $N2 \times N1$, and I is a $N2$ by $N2$ identity matrix. An augmented MIB matrix is therefore obtained by combining (3.36) and (3.37),

$$KW = R, \tag{3.38}$$

where

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

With augmented system (3.38) formulated, Schur complement can then be used for efficient solution U.

3.3 Extensions

3.3.1 An extension on high order algorithms via FFT

The fourth order algorithms via FFT have been demonstrated in great details. High order algorithms, for example sixth and eighth order, can be derived with minor modification on assumptions and parameters. Below lists the procedures for sixth and eighth order schemes in one dimensional manner. Reader interested in such high order algorithm could easily go through the mathematical formulation in two or three dimensions.

Sixth order FFT solution for one-dimensional Poisson equation.

The anti-symmetry for sixth order study can be assumed as

$$u_0 = 0, u_{-1} = -u_1, u_{-2} = -u_2,$$
$$u_M = 0, u_{M+1} = -u_{M-1}, u_{M+2} = -u_{M-2}.$$

The procedure for one dimensional FFT solution of sixth order can be developed as below:

1. Compute Sine transform for $f(x_j)$ via IFST $\hat{f}_l = \frac{2}{M} \sum_{j=1}^{M-1} f(x_j) \sin(\frac{ljp\pi}{M})$, for $l = 1, \cdots, M-1$.
2. $\hat{u}_l = \frac{\hat{f}_l}{\lambda_l}$, for $l = 1, \cdots, M-1$,
   where $\lambda_l = \frac{1}{h_x^2}[\frac{1}{45}\cos(\frac{3l\pi}{M}) - \frac{3}{10}\cos(\frac{2l\pi}{M}) + 3\cos(\frac{l\pi}{M}) - \frac{49}{18}]$.
3. Compute u_j via FST: $u_j = \sum_{l=1}^{M-1} \sin(\frac{ljp\pi}{M})$, for $j = 1, \cdots, M-1$.

Eighth order FFT solution for one-dimensional Poisson equation.

For the eighth order study, we further assume the following anti-symmetry

$$u_0 = 0, u_{-1} = u_1, u_{-2} = -u_2, u_{-3} = -u_3,$$
$$u_M = 0, u_{M+1} = -u_{M-1}, u_{M+2} = -u_{M-2}, u_{M+3} = -u_{M-3}.$$

Therefore the procedure for the FFT solution of eighth order is formulated as below:

1. Compute Sine transform for $f(x_j)$ via IFST $\hat{f}_l = \frac{2}{M}\sum_{j=1}^{M-1} f(x_j)\sin(\frac{lj\pi}{M})$, for $l = 1, \cdots, M-1$.

2. $\hat{u}_l = \frac{\hat{f}_l}{\lambda_l}$, for $l = 1, \cdots, M-1$,
   $\lambda_l = \frac{1}{h_x^2}[-\frac{1}{280}\cos(\frac{4l\pi}{M}) + \frac{16}{315}\cos(\frac{3l\pi}{M}) - \frac{2}{5}\cos(\frac{2l\pi}{M}) + \frac{16}{5}\cos(\frac{l\pi}{M}) - \frac{205}{72}]$.

3. Compute u_j via FST: $u_j = \sum_{l=1}^{M-1}\sin(\frac{lj\pi}{M})$, for $j = 1, \cdots, M-1$.

3.3.2 An extension on high order corrected differences.

Fourth order corrected differences have been explained in the body part of the work. An further extension on high order corrected differences are illustrated below.

Theorem 2. Corrected sixth differences. Let $x_j \le \alpha < x_{j+1}, h^- = x_j - \alpha$, and $h^+ = x_{j+1} - \alpha$. Suppose $u \in C^8[x_j - 3h, \alpha) \bigcap C^8(\alpha, x_{j+1} + 3h]$, with derivative extending continuously up to the interface α. Then the following approximations hold to $O(h^6)$:

$$\begin{aligned} u_{xx}(x_{j-2}) \approx & \frac{1}{h^2}[\frac{1}{90}u(x_{j-5}) - \frac{3}{20}u(x_{j-4}) + \frac{3}{2}u(x_{j-3}) - \frac{49}{18}u(x_{j-2}) + \frac{3}{2}u(x_{j-1}) \\ & -\frac{3}{20}u(x_j) + \frac{1}{90}u(x_{j+1})] - \frac{1}{90h^2}\sum_{m=0}^{7}\frac{(h^+)^m}{m!}[u^{(m)}], \end{aligned}$$

$$\begin{aligned} u_{xx}(x_{j-1}) \approx & \frac{1}{h^2}[\frac{1}{90}u(x_{j-4}) - \frac{3}{20}u(x_{j-3}) + \frac{3}{2}u(x_{j-2}) - \frac{49}{18}u(x_{j-1}) + \frac{3}{2}u(x_j) \\ & -\frac{3}{20}u(x_{j+1}) + \frac{1}{90}u(x_{j+2})] \\ & -\frac{1}{90h^2}\sum_{m=0}^{7}\frac{(h+h^+)^m}{m!}[u^{(m)}] + \frac{3}{20h^2}\sum_{m=0}^{7}\frac{(h^+)^m}{m!}[u^{(m)}], \end{aligned}$$

$$u_{xx}(x_j) \approx \frac{1}{h^2}[\frac{1}{90}u(x_{j-3}) - \frac{3}{20}u(x_{j-2}) + \frac{3}{2}u(x_{j-1}) - \frac{49}{18}u(x_j) + \frac{3}{2}u(x_{j+1}) - \frac{3}{20}u(x_{j+2}) + \frac{1}{90}u(x_{j+3})] - \frac{1}{90h^2}\sum_{m=0}^{7}\frac{(2h+h^+)^m}{m!}[u^{(m)}] + \frac{3}{20h^2}\sum_{m=0}^{7}\frac{(h+h^+)^m}{m!}[u^{(m)}] - \frac{3}{2h^2}\sum_{m=0}^{7}\frac{(h^+)^m}{m!}[u^{(m)}],$$

$$u_{xx}(x_{j+1}) \approx \frac{1}{h^2}[\frac{1}{90}u(x_{j-2}) - \frac{3}{20}u(x_{j-1}) + \frac{3}{2}u(x_j) - \frac{49}{18}u(x_{j+1}) + \frac{3}{2}u(x_{j+2}) - \frac{3}{20}u(x_{j+3}) + \frac{1}{90}u(x_{j+4})] + \frac{1}{90h^2}\sum_{m=0}^{7}\frac{(h^- - 2h)^m}{m!}[u^{(m)}] - \frac{3}{20h^2}\sum_{m=0}^{7}\frac{(h^- - h)^m}{m!}[u^{(m)}] + \frac{3}{2h^2}\sum_{m=0}^{7}\frac{(h^-)^m}{m!}[u^{(m)}],$$

$$u_{xx}(x_{j+2}) \approx \frac{1}{h^2}[\frac{1}{90}u(x_{j-1}) - \frac{3}{20}u(x_j) + \frac{3}{2}u(x_{j+1}) - \frac{49}{18}u(x_{j+2}) + \frac{3}{2}u(x_{j+3}) - \frac{3}{20}u(x_{j+4}) + \frac{1}{90}u(x_{j+5})] + \frac{1}{90h^2}\sum_{m=0}^{7}\frac{(h^- - h)^m}{m!}[u^{(m)}] - \frac{3}{20h^2}\sum_{m=0}^{7}\frac{(h^-)^m}{m!}[u^{(m)}],$$

$$u_{xx}(x_{j+3}) \approx \frac{1}{h^2}[\frac{1}{90}u(x_j) - \frac{3}{20}u(x_{j+1}) + \frac{3}{2}u(x_{j+2}) - \frac{49}{18}u(x_{j+3}) + \frac{3}{2}u(x_{j+4}) - \frac{3}{20}u(x_{j+5}) + \frac{1}{90}u(x_{j+6})] + \frac{1}{90h^2}\sum_{m=0}^{7}\frac{(h^-)^m}{m!}[u^{(m)}].$$

Here note that we may just make m range from 0 to 6 to ensure sixth order accuracy for elliptic problem globally despite local fifth order accuracy.

Theorem 3. Corrected eighth differences. Let $x_j \le \alpha < x_{j+1}, h^- = x_j - \alpha$, and $h^+ = x_{j+1} - \alpha$. Suppose $u \in C^{10}[x_j - 4h, \alpha) \bigcap C^{10}(\alpha, x_{j+1} + 4h]$, with derivative extending continuously up to the

interface α. Then the following approximations hold to $O(h^8)$:

$$\begin{aligned}u_{xx}(x_{j-3}) \approx & \frac{1}{h^2}[-\frac{1}{560}u(x_{j-7}) + \frac{8}{315}u(x_{j-6}) - \frac{1}{5}u(x_{j-5}) + \frac{8}{5}u(x_{j-4}) - \frac{205}{72}u(x_{j-3}) \\ & + \frac{8}{5}u(x_{j-2}) - \frac{1}{5}u(x_{j-1}) + \frac{8}{315}u(x_j) - \frac{1}{560}u(x_{j+1})] + \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(h^+)^m}{m!}[u^{(m)}],\end{aligned}$$

$$\begin{aligned}u_{xx}(x_{j-2}) \approx & \frac{1}{h^2}[-\frac{1}{560}u(x_{j-6}) + \frac{8}{315}u(x_{j-5}) - \frac{1}{5}u(x_{j-4}) + \frac{8}{5}u(x_{j-3}) - \frac{205}{72}u(x_{j-2}) \\ & + \frac{8}{5}u(x_{j-1}) - \frac{1}{5}u(x_j) + \frac{8}{315}u(x_{j+1}) - \frac{1}{560}u(x_{j+2})] \\ & - \frac{8}{315h^2}\sum_{m=0}^{9}\frac{(h^+)^m}{m!}[u^{(m)}] + \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(h+h^+)^m}{m!}[u^{(m)}],\end{aligned}$$

$$\begin{aligned}u_{xx}(x_{j-1}) \approx & \frac{1}{h^2}[-\frac{1}{560}u(x_{j-5}) + \frac{8}{315}u(x_{j-4}) - \frac{1}{5}u(x_{j-3})\frac{8}{5}u(x_{j-2}) - \frac{205}{72}u(x_{j-1}) \\ & + \frac{8}{5}u(x_j) - \frac{1}{5}u(x_{j+1}) + \frac{8}{315}u(x_{j+2}) - \frac{1}{560}u(x_{j+3})] + \frac{1}{5h^2}\sum_{m=0}^{9}\frac{(h^+)^m}{m!}[u^{(m)}] \\ & - \frac{8}{315h^2}\sum_{m=0}^{9}\frac{(h+h^+)^m}{m!}[u^{(m)}] + \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(2h+h^+)^m}{m!}[u^{(m)}],\end{aligned}$$

$$\begin{aligned}u_{xx}(x_j) \approx & \frac{1}{h^2}[-\frac{1}{560}u(x_{j-4}) + \frac{8}{315}u(x_{j-3}) - \frac{1}{5}u(x_{j-2}) + \frac{8}{5}u(x_{j-1}) - \frac{205}{72}u(x_j) \\ & + \frac{8}{5}u(x_{j+1}) - \frac{1}{5}u(x_{j+2}) + \frac{8}{315}u(x_{j+3}) - \frac{1}{560}u(x_{j+4})] \\ & - \frac{8}{5}\frac{1}{h^2}\sum_{m=0}^{9}\frac{(h^+)^m}{m!}[u^{(m)}] + \frac{1}{5h^2}\sum_{m=0}^{9}\frac{(h+h^+)^m}{m!}[u^{(m)}] \\ & - \frac{8}{315h^2}\sum_{m=0}^{9}\frac{(2h+h^+)^m}{m!}[u^{(m)}] + \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(3h+h^+)^m}{m!}[u^{(m)}],\end{aligned}$$

$$u_{xx}(x_{j+1}) \approx \frac{1}{h^2}[-\frac{1}{560}u(x_{j-3}) + \frac{8}{315}u(x_{j-2}) - \frac{1}{5}u(x_{j-1}) + \frac{8}{5}u(x_j) - \frac{205}{72}u(x_{j+1})$$
$$+\frac{8}{5}u(x_{j+2}) - \frac{1}{5}u(x_{j+3}) + \frac{8}{315}u(x_{j+4}) - \frac{1}{560}u(x_{j+5})]$$
$$+\frac{8}{5}\frac{1}{h^2}\sum_{m=0}^{9}\frac{(h^-)^m}{m!}[u^{(m)}] - \frac{1}{5h^2}\sum_{m=0}^{9}\frac{(h^- - h)^m}{m!}[u^{(m)}]$$
$$+\frac{8}{315h^2}\sum_{m=0}^{9}\frac{(h^- - 2h)^m}{m!}[u^{(m)}] - \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(h^- - 3h)^m}{m!}[u^{(m)}],$$

$$u_{xx}(x_{j+2}) \approx \frac{1}{h^2}[-\frac{1}{560}u(x_{j-2}) + \frac{8}{315}u(x_{j-1}) - \frac{1}{5}u(x_j) + \frac{8}{5}u(x_{j+1}) - \frac{205}{72}u(x_{j+2})$$
$$+\frac{8}{5}u(x_{j+3}) - \frac{1}{5}u(x_{j+4}) + \frac{8}{315}u(x_{j+5}) - \frac{1}{560}u(x_{j+6})] - \frac{1}{5h^2}\sum_{m=0}^{9}\frac{(h^-)^m}{m!}[u^{(m)}]$$
$$+\frac{8}{315h^2}\sum_{m=0}^{9}\frac{(h^- - h)^m}{m!}[u^{(m)}] - \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(h^- - 2h)^m}{m!}[u^{(m)}],$$

$$u_{xx}(x_{j+3}) \approx \frac{1}{h^2}[-\frac{1}{560}u(x_{j-1}) + \frac{8}{315}u(x_j) - \frac{1}{5}u(x_{j+1}) + \frac{8}{5}u(x_{j+2}) - \frac{205}{72}u(x_{j+3})$$
$$+\frac{8}{5}u(x_{j+4}) - \frac{1}{5}u(x_{j+5}) + \frac{8}{315}u(x_{j+6}) - \frac{1}{560}u(x_{j+7})]$$
$$+\frac{8}{315h^2}\sum_{m=0}^{9}\frac{(h^-)^m}{m!}[u^{(m)}] - \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(h^- - h)^m}{m!}[u^{(m)}],$$

$$u_{xx}(x_{j+4}) \approx \frac{1}{h^2}[-\frac{1}{560}u(x_j) + \frac{8}{315}u(x_{j+1}) - \frac{1}{5}u(x_{j+2}) + \frac{8}{5}u(x_{j+3}) - \frac{205}{72}u(x_{j+4})$$
$$+\frac{8}{5}u(x_{j+5}) - \frac{1}{5}u(x_{j+6}) + \frac{8}{315}u(x_{j+7}) - \frac{1}{560}u(x_{j+8})] - \frac{1}{560h^2}\sum_{m=0}^{9}\frac{(h^-)^m}{m!}[u^{(m)}],$$

Here note that we may just make m range from 0 to 8 to ensure eighth order accuracy for elliptic problem globally despite local seventh order accuracy.

3.4 Numerical experiments

In this section, we examine the performance of the proposed augmented matched interface and boundary (AMIB) method in solving two or three dimensional Poisson's equation with an arbitrary combination of Dirichlet, Neumann, and Robin boundary conditions. The AMIB approach is primarily designed for high order central difference schemes, but the same approach is applicable to second order central difference as well. In the present study, we will test numerical performance of AMIB2, AMIB4, AMIB6, and AMIB8 methods, with the number after AMIB standing for the designed order of accuracy. For a comparison, the standard MIB boundary closure method [77, 78, 81, 83] will be employed too. Without Schur complement and FFT components, a biconjugate gradient iteration is simply employed in the MIB to solve the sparse system, and the termination criteria are identical to those in the AMIB algorithm. Like the AMIB, one can also specify the order of accuracy for the MIB method.

For simplicity, a square domain with uniform mesh is used such that there are same amount of grids in x, y or z direction in the given domain, respectively. Figure 3.1 gives an illustration of the boundary notations. In the following presentation, we will focus mainly on 2D setting. The numerical accuracy and convergence will be tested by comparing the numerical and exact solutions under the maximum norm and L_2 norm defined as

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

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

where $u_h(x_i, y_j), u(x_i, y_j)$ are numerical and analytical solution, respectively. The set S_{Ω^-} stands for the set of grid points on the given domain before the proposed domain expansion, and N is the number of grid points in each direction of set S_{Ω^-}. The purpose of error analysis on such set is because the solution on the given domain is of concern, but not on Ω^+.

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 of mesh h_i for $i = 1, 2$ using either of the above two defined norms on N by N mesh. The solution calculation is facilitated by a FFT subroutine from Numerical Recipes [62] with 2^n summation. Due to the limitation of partition number equal to 2^n in the subroutine, non-bisection refinement of interval size is used in the error analysis.

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

3.4.1 Accuracy studies

In this section, several comparisons on numerical accuracy between MIB and AMIB methods are studied. In each table below, N_x, N_y and N_z for AMIB and MIB stand for grid numbers on the domain Ω^- in each direction.

Example 1. We first examine the second and fourth order accuracy of the proposed AMIB method for Dirichlet boundary conditions. Tests on 2D Poisson's equation defined in a domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ are studied for this purpose. Here Poisson's equation is given as

$$u_{xx} + u_{yy} = e^x(2 + y^2 + 2\sin(y) + 4x\sin(y)),$$

with analytical solution $u = e^x(x^2 \sin(y) + y^2)$. The Dirichlet boundary condition along the four boundaries are determined according to the exact solution.

The second and fourth order convergence could be observed under each of the two norms in Table 3.3. As the computing time is related to the iteration number for solving the auxiliary variables in Schur complement, we report the iteration numbers in the second last column in Table 3.3. The last column is the CPU time in seconds for each case. For the present study, the iteration number has no change or just increases a little when mesh size is doubled. Consequently, the AMIB method is very efficient. By comparing the second and fourth accurate results, the

AMIB4 is superior to AMIB2 in terms of accuracy and computing time.

Table 3.3: Example 1 –Second and fourth order numerical error analysis.

$[N_x, N_y]$	AMIB2					
	L_∞		L_2		iter no.	CPU time (s)
	Error	Order	Error	Order		
[29, 29]	7.529E-4	–	2.965E-4	–	11	6.95E-3
[61, 61]	1.547e-4	2.03	6.333E-5	2.09	12	2.64E-2
[125, 125]	3.575E-5	1.99	1.496E-5	1.99	12	9.64E-2
[253, 253]	8.631E-6	1.99	3.651E-6	1.99	12	0.325
[509, 509]	2.122E-6	1.99	9.030E-7	1.99	12	1.463
$[N_x, N_y]$	AMIB4					
	L_∞		L_2		iter no.	CPU time (s)
	Error	Order	Error	Order		
[27, 27]	1.336E-6	–	5.251E-7	–	20	1.142E-2
[59, 59]	5.485E-8	3.98	2.301E-8	3.90	20	3.954E-2
[123, 123]	2.892E-9	3.96	1.264E-9	3.90	20	0.151
[251, 251]	1.673E-10	3.97	7.461E-11	3.94	20	0.518
[507, 507]	1.088E-11	3.88	4.669E-12	3.93	21	2.406

Example 2. This example is dedicated to the comparison of AMIB and MIB for solving Poisson's equation

$$u_{xx} + u_{yy} = 2e^x \cos(x + y) - e^x \sin(x + y)$$

on domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, subject to Dirichlet boundary conditions determined by the exact solution $u = e^x \sin(x + y)$. Table 3.4 indicates that both methods produce a fourth order of convergence, while the AMIB4 is much more efficient than the MIB4, especially when a dense mesh is considered.

Table 3.4: Example 2 –Fourth order numerical error analysis of AMIB vs MIB.

$[N_x, N_y]$	AMIB4					
	L_∞		L_2		CPU time(s)	iter no.
	Error	Order	Error	Order		
[11, 11]	4.408E-5	–	1.549E-5	–	4.823E-3	20
[27, 27]	1.096E-6	3.87	4.712E-7	3.65	1.290E-2	20
[59, 59]	4.662E-8	3.94	2.209E-8	3.81	4.056E-2	20
[123, 123]	2.442E-9	3.97	1.212E-9	3.90	0.139	21
[251, 251]	1.400E-10	3.98	7.111E-11	3.95	0.529	21
[507, 507]	7.771E-12	4.10	4.004E-12	4.08	2.403	21
$[N_x, N_y]$	MIB4					
	L_∞		L_2		CPU time(s)	
	Error	Order	Error	Order		
[13, 13]	4.070E-5	–	1.841E-5	–	2.00E-3	
[29, 29]	1.076E-6	4.28	5.267E-7	4.19	2.466E-2	
[61, 61]	4.644E-8	4.12	2.345E-8	4.08	0.207	
[125, 125]	2.444E-9	4.06	1.252E-9	4.04	1.836	
[253, 253]	1.423E-10	4.00	7.329E-11	4.00	15.386	
[509, 509]	3.363E-11	2.06	1.417E-11	2.34	148.41	

Example 3. This example focuses on validating the high-order convergence of our new methods. Restricted on a domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, a 2D Poisson's equation is defined as

$$u_{xx} + u_{yy} = -2k^2 \sin(kx)\cos(ky).$$

subject to mixed boundary conditions:

$$\begin{aligned}
u &= \sin(kx)\cos(\frac{k\pi}{3}) && \text{on } \Gamma_1,\\
u + \frac{\partial u}{\partial n} &= (\sin(\frac{k\pi}{3}) + k\cos(\frac{k\pi}{3}))\cos(ky) && \text{on } \Gamma_2,\\
u + \frac{\partial u}{\partial n} &= (\cos(\frac{k\pi}{3}) - k\sin(\frac{k\pi}{3}))\sin(kx) && \text{on } \Gamma_3,\\
u_x &= k\cos(\frac{k\pi}{3})\cos(ky) && \text{on } \Gamma_4.
\end{aligned}$$

It has analytical solution $u = \sin(kx)\cos(ky)$, in which the wave number k could be chosen as a large number. blue Basically, only for highly oscillating solutions, high-order convergence can be detected numerically. In this example, the parameter k is uniformly set to be 15 for testing the AMIB4, AMIB6, and AMIB8 with mixed boundary conditions.

The high-order convergence of the AMIB method is validated in Table 3.5. It is clear that the AMIB4, AMIB6, and AMIB8 achieves, respectively, the fourth, sixth, and eighth order of convergence. Moreover, the AMIB8 is obviously the most accurate scheme.

The iteration numbers are also reported in Table 3.5. Two comments are in order. First, the iteration number increases with respect to the mesh size for this Poisson problem. But such increment is slow and weakly depends on the mesh size - when the mesh size is doubled for four times, the iteration number is doubled once. Second, the iteration number seems to not depend on the order of AMIB method. This demonstrates the robustness of the AMIB method.

Example 4. This example is devoted to test the AMIB method for solving the Helmholtz equation

$$u_{xx} + u_{yy} + k^2 u = e^{-x^2-\frac{y^2}{2}}(4x^2 + y^2 + k^2 - 3)$$

Table 3.5: Example 3 –High order numerical error analysis.

$[N_x, N_y]$	AMIB4 and $k = 15$				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[27, 27]	6.125E-2	-	1.368E-2	-	30
[59, 59]	3.000E-3	3.76	5.575E-4	3.99	30
[125, 125]	1.476E-4	4.05	2.848E-5	4.00	34
[251, 251]	8.521E-6	3.97	1.614E-6	4.00	43
[507, 507]	5.072E-7	4.00	9.612E-8	4.00	57
$[N_x, N_y]$	AMIB6 and $k = 15$				
	L_∞		L_2		
	Error	Order	Error	Order	
[25, 25]	0.135	–	2.001E-2	–	31
[57, 57]	9.283E-4	5.87	1.496E-4	5.78	36
[123, 123]	7.120E-6	6.39	1.078E-6	6.47	41
[249, 249]	8.167E-8	6.15	1.051E-8	6.38	50
[505, 505]	1.070E-9	6.11	1.317E-10	6.18	69
$[N_x, N_y]$	AMIB8 and $k = 15$				
	L_∞		L_2		
	Error	Order	Error	Order	
[23, 23]	0.140	-	2.354E-2	-	34
[55, 55]	3.103E-4	6.81	4.482E-5	6.97	42
[121, 121]	4.131E-7	8.47	6.780E-8	8.31	48
[247, 247]	9.587E-10	8.25	1.242E-10	8.57	57
[503, 503]	3.173E-12	8.01	3.760E-13	8.13	70

subject to Robin boundary conditions:

$$
\begin{aligned}
u + \frac{\partial u}{\partial n} &= (1 - \frac{\pi}{3}) e^{-x^2 - \frac{\pi^2}{18}} \quad \text{on } \Gamma_1, \\
u + \frac{\partial u}{\partial n} &= (1 + \frac{2\pi}{3}) e^{-\frac{\pi^2}{9} - \frac{y^2}{2}} \quad \text{on } \Gamma_2, \\
u + \frac{\partial u}{\partial n} &= (1 + \frac{\pi}{3}) e^{-x^2 - \frac{\pi^2}{18}} \quad \text{on } \Gamma_3, \\
u + \frac{\partial u}{\partial n} &= (1 - \frac{2\pi}{3}) e^{-\frac{\pi^2}{9} - \frac{y^2}{2}} \quad \text{on } \Gamma_4,
\end{aligned}
$$

which are determined by analytical solution $u = e^{-x^2 - \frac{y^2}{2}}$ on domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. Parameter k is chosen to be 20 in the test. Table 3.6 shows that the AMIB4 method is equally successful in solving the Helmholtz equation with Robin boundary conditions.

Table 3.6: Example 4 –Fourth order numerical error analysis for the Helmholtz equation with $k = 20$.

$[N_x, N_y]$	AMIB4				
	L_∞		L_2		iter no.
	Error	Order	Error	Order	
[27, 27]	6.255E-7	–	2.218E-7	–	53
[59, 59]	1.1732E-8	4.96	5.777E-9	4.55	38
[123, 123]	9.682E-10	3.35	3.307E-10	3.85	64
[251, 251]	5.751E-11	3.94	2.014E-11	3.90	75
[507, 507]	3.502E-12	3.97	1.246E-12	3.95	94

Example 5. Both boundary condition implementation and solution of algebraic system are conducted in dimension by dimension manner in the AMIB method. Thus, the generalization of 2D AMIB method to 3D is straightforward. Here the AMIB4 method is verified by the following 3D Poisson problem:

$$u_{xx} + u_{yy} + u_{zz} = \sin(kx)e^{y+z}(2 - k^2)$$

with Dirichlet boundary conditions determined by solution $u = \sin(kx)e^{y+z}$ on the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. Parameter k is set to be 5 in the 3D test. The fourth order convergence of the AMIB4 for 3D Poisson problem is numerically verified through Table 3.7. Moreover, the iteration number increases very little when the mesh size is doubled in each dimension.

Table 3.7: Example 5 –Fourth order numerical error analysis of the AMIB4 on 3D Poisson's equation.

$[N_x, N_y, N_z]$	AMIB4					
	L_∞		L_2		iter no.	CPU times(s)
	Error	Order	Error	Order		
[11, 11, 11]	0.870	–	0.127	–	37	4.722E-2
[27, 27, 27]	3.337E-2	3.81	3.347E-3	3.80	49	0.392
[59, 59, 59]	1.426E-3	4.13	1.216E-4	4.13	52	3.191
[123, 123, 123]	7.436E-5	4.10	5.760E-6	4.10	55	28.792
[251, 251, 251]	4.270E-6	4.06	3.132E-7	4.06	55	296.97

3.4.2 Computational efficiency

In this section, the computational efficiency of the AMIB method will be further investigated. A comparison between the results from FISHPACK [66], AMIB2 and AMIB4 will be considered in 2D. In the FISHPACK, a direct FFT implementation is conducted for the second order central difference. Theoretically, the computational cost of FISHPACK is of $O(N^2 \log N)$, where N is the degree of freedom in each dimension on a squared domain. In the proposed AMIB method, an iterative solution is needed for the Schur complement system, whose size is one-dimensionally smaller. Because the iteration number weakly depends on N, the computational cost of the AMIB method could be on the order of $O(N^2 \log N)$ too. This will be numerically verified in this subsection.

We note two things in our flop order test. First, the change of $\log N$ in comparing with N is almost

negligible in the total computational cost. So we have to drop this term in complexity in flop order test. In particular, we will fit the CPU time to the form of $O(N^r)$ by the least squares method. The numerically detected flop order r will characterize the efficiency of the AMIB method. Second, we re-studied Example 3 from the last section by changing boundary conditions. This is because FISHPACK cannot handle Robin condition. Instead, Dirichlet boundary conditions determined by exact solutions are employed.

In the experiment, the parameter k is set to be 5. We adopt 10^{-10} as the error tolerance for the AMIB2, as it already guarantees the expected convergence. For the AMIB4, the tolerance is still 10^{-15}. From table 3.8, we can see that both AMIB2 and FISHPACK produce second order accuracy. The AMIB4 has the expected fourth order accuracy. To analyze the flop order, the CPU time is plotted against the degree of freedom in each dimension, i.e., N, in Figure 3.5. For a given N, AMIB algorithm is more expensive than FISHPACK, while the CPU time difference between AMIB2 and AMIB4 is small. Moreover, least squares fitting is conducted to compute flop order r in $O(N^r)$ setting. From Figure 3.5, it can be observed that the flop order r for each method is around 2. Thus, the computational cost of the AMIB method is roughly on the order of $O(N^2 \log N)$ too, whereas the AMIB method has larger proportionality constants on the operation count $O(N^2 \log N)$ compared to FISHPACK.

On the other hand, in terms of cost-efficiency, high order methods have the advantage of providing better accuracy than lower order methods on the same mesh size. Table 3.8 shows that the AMIB4 has accuracy significantly improved than the second order methods. In fact, the errors of the AMIB4 on a coarse mesh 65×65 are smaller than those of second order AMIB2 and FISHPACK on 513×513. To demonstrate the cost-efficiency, we plot the accuracy against CPU time in Figure 3.6 for both error norms. Obviously, the AMIB2 scheme as a byproduct of this study, is dominated by the FISHPACK - they are of the same accuracy level, while the AMIB2 is slower. Nevertheless, it can be observed in this figure that to achieve a precision around 10^{-5}, the proposed AMIB4 method is more efficient than the FISHPACK. Moreover, the efficiency gain of the AMIB4 becomes more significant when a better accuracy is targeted. For log-log lines shown

in Figure 3.6, we calculated the slopes by least squares fitting. The AMIB4 yields a slope around -2, while that of FISHPACK is about -1. Based on these results, it is projected that to achieve a precision around 10^{-10}, the FISHPACK would take about 10^5 seconds of CPU time (assume memory is large enough), which is about $10,000$ times more expensive than the proposed AMIB4 scheme.

Table 3.8: Computational efficiency comparison.

$[N_x, N_y]$	FISHPACK					
	L_∞		L_2		CPU time (s)	
	Error	Order	Error	Order		
[65, 65]	2.384E-3	–	9.717E-4	–	1.098E-3	
[129, 129]	6.073E-4	1.97	2.464E-4	1.98	4.837E-3	
[257, 257]	1.530E-4	1.99	6.207E-5	1.99	1.665E-2	
[513, 513]	3.840E-5	1.99	1.558E-5	1.99	6.084E-2	
[1025, 1025]	9.617E-6	2.00	3.902E-6	2.00	0.243	
$[N_x, N_y]$	AMIB2					
	L_∞		L_2		iter no.	CPU time (s)
	Error	Order	Error	Order		
[61, 61]	2.790E-3	–	1.087E-3	–	10	1.92E-2
[125, 125]	6.565E-4	1.99	2.604E-4	1.97	10	7.432E-2
[253, 253]	1.590E-4	2.00	6.380E-5	1.98	12	0.311
[509, 509]	3.915E-5	2.00	1.579E-5	1.99	12	1.471
[1021, 1021]	9.711E-6	2.00	3.929E-6	2.00	12	6.267
$[N_x, N_y]$	AMIB4					
	L_∞		L_2		iter no.	CPU time (s)
	Error	Order	Error	Order		
[59, 59]	1.273E-5	–	4.764E-6	–	19	4.100E-2
[123, 123]	6.609E-7	3.98	2.570E-7	3.93	19	0.127
[251, 251]	3.768E-8	3.99	1.496E-8	3.97	20	0.516
[507, 507]	2.250E-9	4.00	9.03E-10	3.98	20	2.284
[1019, 1019]	1.374E-10	4.00	5.544E-11	4.00	21	10.613

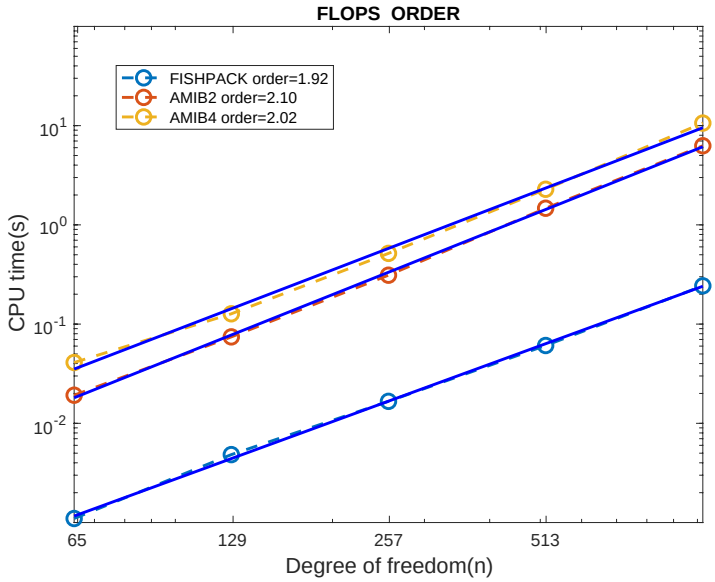

Figure 3.5: A comparison of computational cost between three methods, and flop orders are tested to be around 2 for each method.

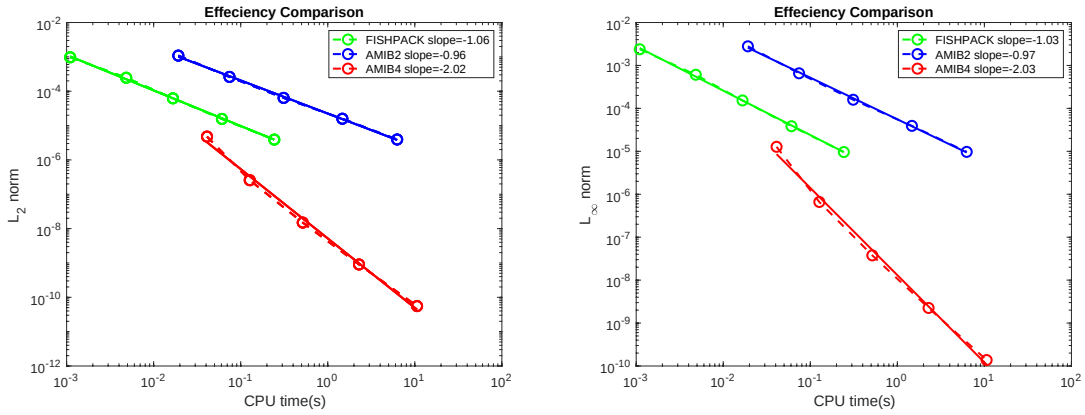

Figure 3.6: A comparison between the three methods on accuracy VS. CPU time under two norms.

CHAPTER 4

AMIB FOR POISSON BOUNDARY VALUE PROBLEM ON STAGGERED GRIDS

4.1 Preliminary

In this chapter, we are concerned with the development of fast and high order Poisson solvers for rectangular domains and uniform meshes with staggered boundaries. It is known that staggered grids are widely used in scientific computing to solve partial differential equations (PDEs) with multiple dependent variables, such as incompressible Navier-Stokes equations [31, 76], Maxwell's equations [82], and wave equations [12, 40]. By defining different variables at different grid locations such as centers or edges, a staggered grid could avoid the odd-even decoupling error of a collocated grid arrangement. Moreover, staggered grids enjoy some advantages in handling multiple variables, including better computational efficiency, smaller memory requirement and easier implementation. In such scenario, the boundaries of a rectangular or cubic domain is staggered with the grid, i.e., the boundary locates midway between two adjacent grid nodes. We will call such a boundary as a staggered boundary, on which Dirichlet, Neumann, or Robin boundary condition needs to be satisfied. The problem to be solved can be regarded as one component needed in solving multiple variable PDEs on staggered grids. For example, the Poisson problem modeling pressure on staggered grids is derived in the projection method on unsteady incompressible Navier-Stokes equation, where solution from fast Poisson solver is desired to accelerate the computation process [31]. Moreover, high order methods have been applied to tackle many Poisson-related problems on staggered grids [12, 40, 76]. It is thus interested to develop fast and high order Poisson solvers for staggered boundaries.

4.2 Theory and algorithm

4.2.1 Immersed boundary problem

For staggered boundaries in 2D, the following grid node coordinates are defined

$$R^0 = \{(x_i, y_j) | x_i = a + (i - 0.5)h_x,\ y_j = c + (j - 0.5)h_y,$$
$$i = 0, \cdots, M + 1,\ j = 0, \cdots, N + 1\} \tag{4.1}$$

where $h_x = \frac{b-a}{M}, h_y = \frac{d-c}{N}$ such that the boundaries of the given domain lie in midway between two grid points as shown in Fig 4.1. However, as mentioned in last section, it is necessary to embed the original domain Ω into a larger rectangle such that the aforementioned FFT solvers can be applied when the zero solutions are defined on the extended part. The extended length of $2.5h_x$ or $2.5h_y$ for $x-$ or $y-$ direction is sufficient in supporting fourth order solvers, because the requirement of being anti-symmetric across the new boundary can be satisfied. A comparison between the current grid setting and the one for non-staggered boundaries is shown in Fig. 4.2. After extension, the grid coordinates are redefined as:

$$R = \{(x_i, y_j) | x_i = a + (i - 2.5)h_x,\ y_j = c + (j - 2.5)h_y,$$
$$i = 0, \cdots, M + 5,\ j = 0, \cdots, N + 5\}, \tag{4.2}$$

where h_x, h_y are the same as above. The figure of grids in (4.2) would be equipped with two more layers of grid points outside each side of the domain in comparison with Fig. 4.1.

4.2.2 Reconstructing the Cartesian derivative jumps

Polynomial approximation to derivative jumps as previously proposed can still be employed for derivative jumps $[u^{(m)}]$ in the corrected differences. The right one in Fig. 4.2 shows two layers of fictitious values needed near the interface Γ for the fourth order corrected differences.

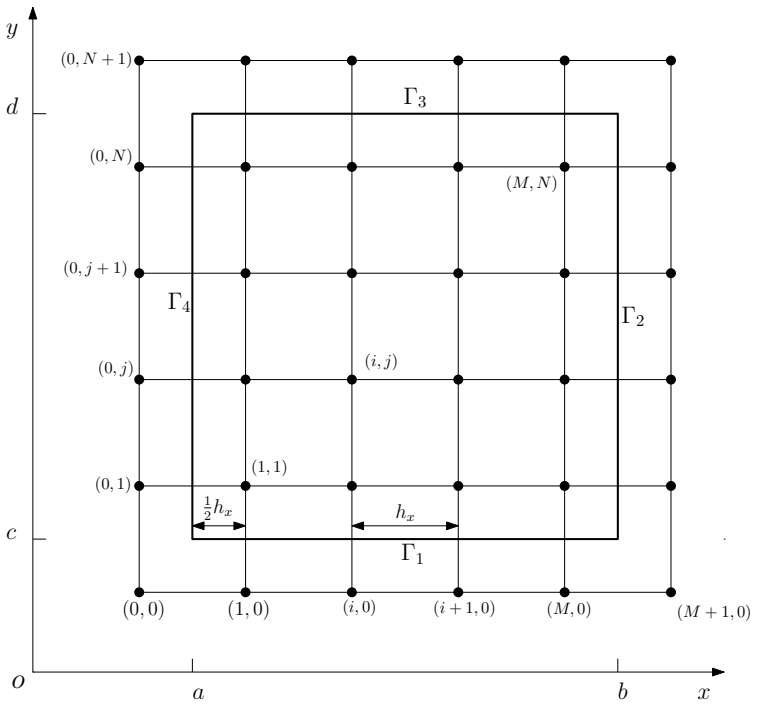

Figure 4.1: Poisson solutions on black dots in a uniform grid with staggered boundaries. Each boundary is located midway between two nodes.

Fictitious values formulation

For fictitious value generation across staggered grid, it is still sufficient to illustrate the process in the case of Robin boundary condition in 1D,

$$ku + \frac{\partial u}{\partial n} = ku - \frac{\partial u}{\partial x} = \phi \quad \text{on } \Gamma_1. \tag{4.3}$$

This procedure can be easily modified for the cases of Dirichlet and Neumann boundary conditions, as Robin boundary condition is essentially comprised of these two conditions. We generate fictitious values iteratively by repeatedly matching the boundary conditions across the boundary. Referring to Figure 4.3, to generate M fictitious values $\hat{u}_i$ for $i = 1, 2, \cdots, M$ outside the domain, $L + 1$ function values u_j for $j = 0, 1, \cdots, L$ inside the domain are to be used. We can

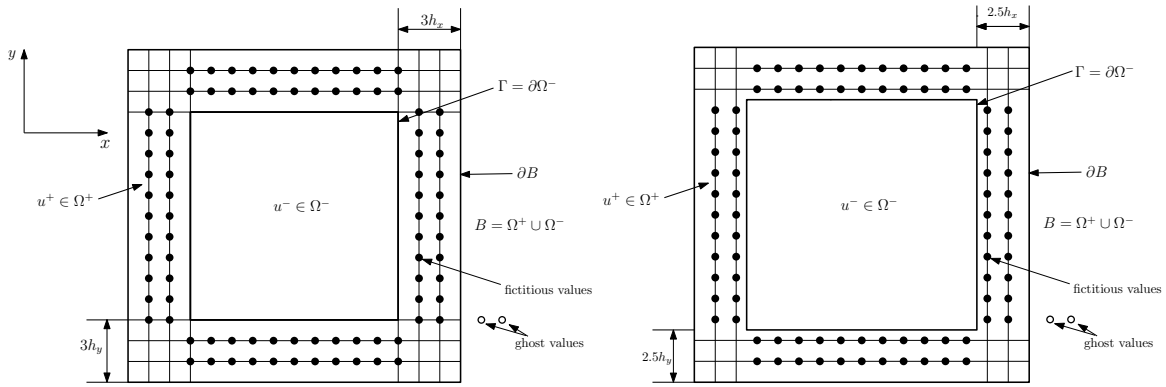

Figure 4.2: A comparison on two kinds of grids construction for immersed boundary problem with fourth order central difference scheme. The left is for usual grid, and the right one is for staggered grid. Filled circles stand for fictitious values used in approximation of jump quantities from corrected differences. *Ghost values* (depicted as open circles) are zero padding values beyond the expanded domain satisfying anti-symmetry.

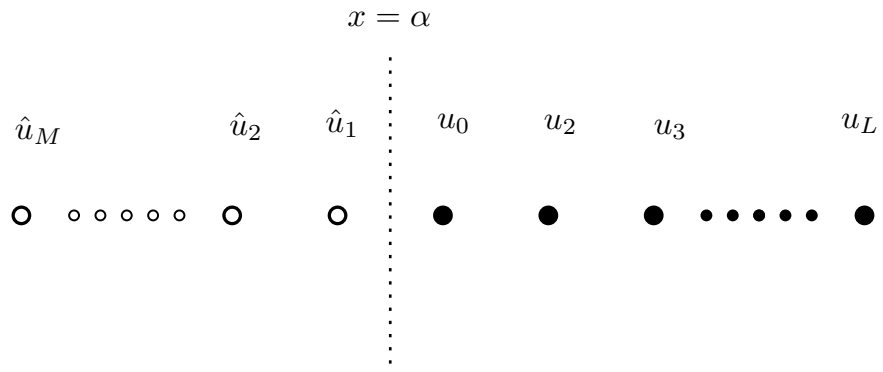

Figure 4.3: An illustration of fictitious values near the staggered left boundary.

generate the first layer of fictitious value $\hat{u}_1$ according to approximation to boundary (4.3):

$$k(C_{2,1}^{(0)}\hat{u}_1 + \sum_{i=2}^{L+2} C_{2,i}^{(0)} u_{i-2}) + (C_{2,1}^{(1)}\hat{u}_1 + \sum_{i=2}^{L+2} C_{2,i}^{(1)} u_{i-2}) = \phi, \tag{4.4}$$

where $C_{2,i}^{(0)}$ and $C_{2,i}^{(1)}$ are finite difference (FD) weights in approximating zeroth and first derivative at $x = a$ by using a stencil $\{\hat{u}_1, u_0, \cdots, u_L\}$. The approximation in (4.4) is different from (3.27) for the usual Robin boundary conditions in the sense that the Dirichlet term is approximated as well due to that fact the domain boundary does not align with grid lines. The notation 2 in first subscript of $C_{2,i}^{(1)}$ and $C_{2,i}^{(1)}$ is because u_0 is the second point in the FD stencils. As a result, unknown $\hat{u}_1$ can be represented as the combination of u_i for $i = 0, 1, \cdots, L$ and ϕ. Taking $\hat{u}_1$ as

known, we can determine another fictitious value $\hat{u}_2$ by one more iteration

$$k(C_{3,1}^{(0)}\hat{u}_2 + C_{3,2}^{(0)}\hat{u}_1 + \sum_{i=3}^{L+3} C_{3,i}^{(0)}u_{i-3}) + (C_{3,1}^{(1)}\hat{u}_2 + C_{3,2}^{(1)}\hat{u}_1 + \sum_{i=3}^{L+3} C_{3,i}^{(1)}u_{i-3}) = \phi, \tag{4.5}$$

where $C_{3,i}^{(0)}, C_{3,i}^{(1)}$ are FD weights to approximate zeroth and first order derivatives at $x = a$ using a stencil $\{\hat{u}_2, \hat{u}_1, u_0, u_1, \cdots, u_L\}$. The first subscript 3 in $C_{3,i}^{(0)}$ and $C_{3,i}^{(1)}$ is because u_0 is the third point in the FD stencils. In such iterative approach, we can obtain representation of fictitious values $\hat{u}_3, \hat{u}_4, \cdots, \hat{u}_M$ in terms of u_i for $i = 0, 1, \cdots, L$. Still, L is related to M with $L + 1 = 2M$. As shown below for the fourth order study, two layers of fictitious values are needed for approximation of jump quantities, which means M and L is set to be 2 and 3 respectively.
We here conduct some error analysis about the accuracy of the fictitious values by enforcing Robin boundary conditions, which will be used for further analysis in the next subsection. In the situation of $M = 2$ and $L = 3$, we assume the real function values to be f_1 and f_2 corresponding to the fictitious values $\hat{u}_1$ and $\hat{u}_2$ at the same locations. Due to Lagrange interpolation, we have following equation:

$$k(C_{2,1}^{(0)}f_1 + \sum_{i=2}^{5} C_{2,i}^{(0)}u_{i-2} + O(h^5)) + (C_{2,1}^{(1)}f_1 + \sum_{i=2}^{5} C_{2,i}^{(1)}u_{i-2} + O(h^4)) = \phi, \tag{4.6}$$

Subtracting Eq.(4.5) from Eq.(4.6) gives

$$k[C_{2,1}^{(0)}(\hat{u}_1 - f_1) + O(h^5)] + [C_{2,1}^{(1)}(\hat{u}_1 - f_1) + O(h^4)] = 0.$$

Equivalently,

$$(kC_{2,1}^{(0)} + C_{2,1}^{(1)})(f_1 - \hat{u}_1) = O(h^5) + O(h^4) = O(h^4).$$

Therefore,

$$\epsilon_1 = f_1 - \hat{u}_1 = \frac{O(h^4)}{kC_{2,1}^{(1)} + C_{2,1}^{(2)}} = \frac{O(h^4)}{O(1) + O(\frac{1}{h})} = O(h^5).$$

The last second equality holds because $C_{2,i}^{(0)}$ and $C_{2,i}^{(1)}$ scale as $O(1)$ and $O(\frac{1}{h})$ respectively. From this, we may notice that the error between real function value f_1 and the fictitious value $\hat{u}_1$ is of $O(h^5)$.

In the second step with one more fictitious value, it holds that

$$\begin{aligned} k(C_{3,1}^{(0)} f_2 + C_{3,2}^{(0)} f_1 + \sum_{i=3}^{5} C_{3,i}^{(0)} u_{i-3} + O(h^6)) \\ +(C_{3,1}^{(1)} f_2 + C_{3,2}^{(1)} f_1 + \sum_{i=3}^{5} C_{3,i}^{(1)} u_{i-3} + O(h^5)) = \phi, \end{aligned} \tag{4.7}$$

Subtracting Eq.(4.5) from Eq.(4.7) yields to

$$k[C_{3,1}^{(0)}(\hat{u}_2 - f_2) + C_{3,2}^{(0)}(\hat{u}_1 - f_1) + O(h^6)] + [C_{3,1}^{(1)}(\hat{u}_2 - f_2) + C_{3,2}^{(1)}(\hat{u}_1 - f_1) + O(h^5)] = 0.$$

Since $\hat{u}_1 - f_1 = O(h^5)$, $C_{3,i}^{(0)}$ scales as $O(1)$, and $C_{3,i}^{(1)}$ scales $O(\frac{1}{h})$, we arrive at

$$k[C_{3,1}^{(0)}(\hat{u}_2 - f_2) + O(h^5)] + [C_{3,1}^{(1)}(\hat{u}_2 - f_2) + O(h^4)] = 0.$$

Equivalently,

$$(kC_{3,1}^{(0)} + C_{3,1}^{(1)})(f_2 - \hat{u}_2) = O(h^5) + O(h^4) = O(h^4).$$

Hence,

$$\epsilon_2 = f_2 - \hat{u}_2 = \frac{O(h^4)}{kC_{3,1}^{(0)} + C_{3,1}^{(1)}} = \frac{O(h^4)}{O(1) + O(\frac{1}{h})} = O(h^5).$$

In short, the errors of fictitious values $\hat{u}_1$ and $\hat{u}_2$ are of $O(h^5)$ when $L = 3$.

The MIB implementation of Dirichlet or Neumann boundary condition at staggered boundaries can be similarly formulated. Each implementation may just follow the formula inside the first or second parenthesis in (4.4) and (4.5).

The present MIB scheme has some advantages over that on the usual grid discussed in the last section. For instance, it is still valid even if the solution loses some high order regularity on the boundary. We will demonstrate this in numerical experiments.

In summary, the fictitious values $\hat{u}_{i,j}$ at (x_i, y_j) calculated from Dirichlet, Neumann, and Robin boundary conditions can be written into a general form:

$$\hat{u}_{i,j} = \sum_{(x_I, y_J) \in \mathbb{S}_{i,j}} W_{I,J} u_{I,J} + W_0 \phi, \tag{4.8}$$

which is a linear combination of functions values on a chosen set $\mathbb{S}_{i,j}$, and known boundary data. The derivative jumps are approximated in the same manner as before using fictitious values. Below a short approximation error analysis is given:

$$\begin{aligned}
[\frac{\partial^m u}{\partial x^m}] &= \frac{\partial^m u^-}{\partial x^m} - 0 \\
&= w^m_{i-1,j} f_{i-1,j} + w^m_{i,j} f_{i,j} + \sum_{k=1}^{3} w^m_{i+k,j} u_{i+k,j} + O(h^{5-m}) \\
&= w^m_{i-1,j} (\hat{u}_{i-1,j} + O(h^5)) + w^m_{i,j} (\hat{u}_{i,j} + O(h^5)) + \sum_{k=1}^{3} w^m_{i+k,j} u_{i+k,j} + O(h^{5-m}) \\
&= w^m_{i-1,j} \hat{u}_{i-1,j} + O(h^{5-m}) + w^m_{i,j} \hat{u}_{i,j} + \sum_{k=1}^{3} w^m_{i+k,j} u_{i+k,j} + O(h^{5-m}) \\
&= w^m_{i-1,j} \hat{u}_{i-1,j} + w^m_{i,j} \hat{u}_{i,j} + \sum_{k=1}^{3} w^m_{i+k,j} u_{i+k,j} + O(h^{5-m})
\end{aligned} \tag{4.9}$$

where $u^- \in \Omega^-$, and $w^m_{p,q}$ stands for the weights in finite difference approximation at $x = \alpha$ and scales as $O(h^{-m})$. This leads to the approximation error of $O(h^{5-m})$ in the last second equality. The approximation in the first equality is due to the Lagrange interpolation error. In $w^m_{p,q}$, subscript

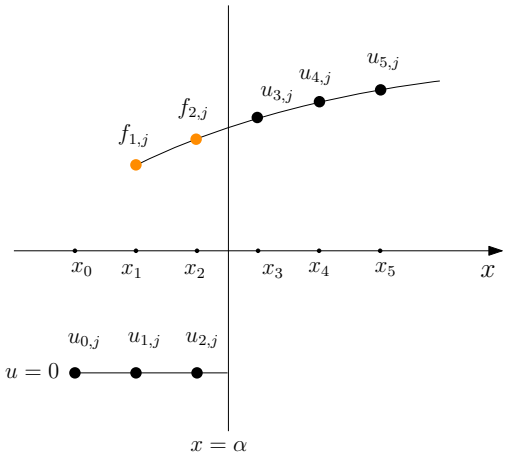

Figure 4.4: An illustration of jumps approximation via a Lagrange polynomial from inside of the interface $x = \alpha$ and zeros solution from the outside for the fourth order approximation. The black dots stand for the real values while the yellow dots denote the fictitious values.

(p, q) denotes the location (x_p, y_q), and $m = 0, 1, \cdots, 4$ indicates the mth derivative. Consequently, the local truncation error in the fourth order corrected differences will keep third order of $O(h^3)$.

4.2.3 Augmented system

With the fictitious values used in approximation to the derivative jumps of the corrected differences, augmented system as in (3.38) can be formulated for solving Poisson BVP on staggered grid as before,

$$KW = R.$$

Schur complement is then employed to solve for high order fast solution of the Poisson boundary value problem on staggered grid.

4.3 Numerical experiments

In this section, we examine the performance of the proposed AMIB method in solving two or three dimensional Poisson's equation with an arbitrary combination of Dirichlet, Neumann, and Robin boundary conditions on staggered boundaries. The numerical performance of AMIB4, AMIB6, and AMIB8 methods will be tested, with the number after AMIB standing for the designed order of accuracy. Moreover, we will compare the present AMIB with the one in [26] in

some examples. To denote the difference, we use S to stand for the study with staggered boundary conditions. For example, AMIB4-S indicates the fourth order AMIB method on staggered boundary conditions, while AMIB4-NS means the non-staggered case.

The domain setup and the norms and convergence rate for numerical errors are defined as in the last section. All the experiments were carried out on MacBook Pro with 8.00 RAM and Intel 2.3 GHz Intel Core i5.

4.3.1 The AMIB method for staggered boundaries

In this subsection, we examine the AMIB method for Poisson problems in 2D and 3D with Dirichlet conditions.

Example 1. This example focuses on validating the ultra high orders of our new schemes. Restricted on a domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, a 2D Poisson's equation is defined as

$$u_{xx} + u_{yy} = -2k^2 \sin(kx)\cos(ky) \tag{4.10}$$

with Dirichlet boundary condition determined by analytical solution $u = \sin(kx)\cos(ky)$, in which the wave number k could be chosen as a large number. Basically, only for highly oscillating solutions, ultra high order of convergence can be detected numerically. In this example, AMIB4-S, AMIB6-S, AMIB8-S are examined under above Dirichlet boundary conditions with the parameter k set to be 5,10 and 20, respectively.

The high order convergence of the AMIB-S methods is validated in Table 4.1. We note that because different wave numbers are used, the error comparison among different orders does not make much sense. In particular, the errors on the coarsest mesh for the AMIB8-S are much larger than those of AMIB6-S, simply because the numerical problem becomes much challenging for $k = 20$. It can be seen that the fourth and sixth orders are achieved. The eighth order accuracy of AMIB8-S is not reached first, but is attained eventually when the mesh resolution is fine enough. As the computing time is related to the iteration number for solving the auxiliary variables in Schur complement, we report the iteration number in the second last column in Table 4.1. The

last column is the CPU time in seconds for each case. For the present study, the iteration number has no change or just increases a little when mesh size is doubled. Consequently, we may say that the iteration number weakly depends the mesh size in this case, and the AMIB-S method is very efficient.

Table 4.1: Example 1 –High order numerical error analysis.

$[N_x, N_y]$	AMIB4-S and $k = 5$					
	L_∞		L_2		iter no.	CPU time
	Error	Order	Error	Order		
[27, 27]	2.917E-4	–	1.131E-4	–	11	4.912E-3
[59, 59]	1.133E-5	4.16	4.655E-6	4.08	11	1.168E-2
[123, 123]	5.956E-7	4.01	2.430E-7	4.02	10	3.261E-2
[251, 251]	3.563E-8	3.95	1.431E-8	3.97	10	0.125
[507, 507]	2.187E-9	3.97	8.795E-10	3.97	10	0.468
$[N_x, N_y]$	AMIB6-S and $k = 10$					
	L_∞		L_2		iter no.	CPU time
	Error	Order	Error	Order		
[25, 25]	9.183E-4	–	3.241E-4	–	30	1.135E-2
[57, 57]	9.968E-6	5.48	2.885E-6	5.73	38	3.671E-2
[121, 121]	1.160E-7	5.92	3.479E-8	5.87	44	1.121
[249, 249]	1.762E-9	5.80	4.823E-10	5.93	48	0.447
[505, 505]	2.920E-11	5.80	7.158E-12	5.95	52	2.029
$[N_x, N_y]$	AMIB8-S and $k = 20$					
	L_∞		L_2			
	Error	Order	Error	Order		
[23, 23]	4.932E-2	–	1.343E-2	–	35	1.451E-2
[55, 55]	1.439E-4	6.70	2.625E-5	7.15	46	5.823E-2
[119, 119]	4.458E-7	7.49	6.451E-8	7.79	53	0.177
[247, 247]	1.830E-9	7.53	2.013E-10	7.90	60	0.603
[503, 503]	6.625E-12	7.90	7.006E-13	7.96	63	2.482

Example 2. Both boundary condition implementation and solution of algebraic system are conducted in dimension by dimension manner in the AMIB method. Thus, the generalization of 2D AMIB method to 3D is straightforward. Here the performance of AMIB4-S is tested by the following 3D Poisson problem:

$$u_{xx} + u_{yy} + u_{zz} = \sin(kx)e^{y+z}(2 - k^2) \tag{4.11}$$

with Dirichlet boundary conditions determined by solution $u = \sin(kx)e^{y+z}$ on the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. Parameter k is set to be 5 in this 3D test. The fourth order convergence of the AMIB4-S for 3D Poisson problem is numerically verified through Table 4.2. Moreover, the iteration number increases very little when the mesh size is doubled in each dimension.

Table 4.2: Example 2 –Fourth order numerical error analysis of the AMIB4-S on 3D Poisson's equation.

$[N_x, N_y, N_z]$	AMIB4-S					
	L_∞		L_2		iter no.	CPU times(s)
	Error	Order	Error	Order		
[11, 11, 11]	2.291E-2	–	5.127E-3	–	31	4.098E-2
[27, 27, 27]	6.986E-4	3.89	1.780E-4	3.74	39	0.301
[59, 59, 59]	3.095E-5	3.99	7.852E-6	3.99	42	2.689
[123, 123, 123]	1.639E-6	4.00	4.161E-7	4.00	44	24.28
[251, 251, 251]	9.445E-8	4.00	2.400E-8	4.00	45	246.57

4.3.2 Comparison of AMIB-S and AMIB-NS

Because only a single variable Poisson's equation and a uniform mesh are considered in this dissertation, the users actually have the freedom to partition the domain into either staggered or non-staggered boundaries. This means for many problems studied in this work, both AMIB-S and AMIB-NS methods can be applied. It is thus interested to compare the performance of two methods for solving the same problem. In particular, two special examples are designed to demonstrate the advantages of AMIB-S over AMIB-NS.

Example 3. This example shows some advantage of AMIB4-S over AMIB4-NS on Poisson problem with analytical solution:

$$u = \sin(kx)(y + \frac{\pi}{3})^{\frac{5}{2}} \tag{4.12}$$

on domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with $k = 5$. On four sides, Dirichlet boundary conditions are given by the analytical solution. Note that the analytical solution has good smoothness except at $y = -\frac{\pi}{3}$, on which the partial derivatives with respect to y are defined only up to second order.

Recall that the implementations of AMIB-S and AMIB-NS are different for Dirichlet conditions. For AMIB4-S, the boundary enforcement is basically through interpolation. Nevertheless, for AMIB4-NS, numerical boundary conditions like (3.29) need to be constructed, which essentially involve approximations of Laplacian along $y = -\frac{\pi}{3}$. For the fourth order central difference approximations, this requires at least sixth order regularity, which is obviously not true for the present example. Hence, order reduction is inevitable for the AMIB4-NS scheme. On the other hand, since the boundary $y = -\frac{\pi}{3}$ is staggered with respect to the grid, the approximation of

Laplacian is simply avoided at this low-regularity boundary in the AMIB4-S discretization. Consequently, the order of AMIB4-S is not affected by the low regularity solution.
In the table 4.3, we can observe that AMIB4-S is superior to AMIB4-NS in preserving fourth order accuracy on the numerical results.

Table 4.3: Example 3 –Fourth order numerical error analysis of the AMIB4-S and AMIB4-NS.

$[N_x, N_y]$	AMIB4-S					
	L_∞		L_2		iter no.	CPU time
	Error	Order	Error	Order		
[59, 59]	3.739E-5	–	1.270E-5	–	42	3.556E-2
[123, 123]	1.977E-6	4.00	6.737E-7	4.00	41	9.587E-2
[251, 251]	1.139E-7	4.00	3.946E-8	3.97	43	0.371
[507, 507]	6.847E-9	4.00	2.636E-9	3.85	44	1.690
$[N_x, N_y]$	AMIB4-NS					
	L_∞		L_2			
	Error	Order	Error	Order		
[59, 59]	4.107E-5	–	1.341E-5	–	20	1.780E-2
[123, 123]	3.426E-6	3.34	8.878E-7	3.65	20	6.472E-2
[251, 251]	6.210E-7	2.38	1.062E-7	2.96	20	0.189
[507, 507]	1.116E-7	2.43	1.746E-8	2.56	21	0.845

Example 4. In this example, we consider a problem in which the PDE is not satisfied at the Dirichlet boundaries. Such a problem is frequently encountered in practice, due to some randomness or perturbation in experimental measurements. On domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, the Poisson problem is given as

$$u_{xx} + u_{yy} = -2k^2 \sin(kx)\cos(ky) \tag{4.13}$$

subject to a perturbed Dirichlet boundary condition defined by the function $v(x, y) = \sin(k_0 x)\cos(k_0 y)$. A small perturbation exists in parameters with $k = k_0 + \epsilon$. Here we set $k_0 = 10$ and $\epsilon = 0.01$.
The AMIB4-NS scheme is not applicable to this problem, because the PDE solution will not satisfy the Dirichlet condition. Hence, the numerical boundary condition (3.29) based on the PDE and Dirichlet condition is simply erroneous. This leaves the AMIB4-S scheme a natural choice for the present problem, which can still maintain the fourth order convergence. As there is no analytical solution for this problem, we generate reference solutions on sufficiently dense mesh to

carry out numerical analysis, which means we have to compare the solutions on coarse meshes to those on fine meshes. Suppose that there are N grids on the coarse mesh of the given domain and M grids on the fine mesh in each direction. Here M has to be $(1 - 2k_2)$ times of N with k_2 being negative integers such the each grid on coarse mesh can agree with the grid on fine mesh at the same location. The index i on the coarse mesh should correspond to j on fine mesh with the relation $j = (1 - 2k_2)i + k_2$. In this way, we can use the solutions on sufficiently fine mesh as reference solutions for numerical analysis. In our numerical experiments, we choose fine meshes with k_2 as $-39, -19, -9, -4$ and -2 for grid number N_x equal to $10, 20, 40, 80$, and 160 on the coarse meshes. In this example, the FFT subroutine used is from FFTPACK.

Table 4.4: Example 4–Fourth order numerical error analysis of the AMIB4-S.

$[N_x, N_y]$	AMIB4-S					
	L_∞		L_2		iter no.	CPU time
	Error	Order	Error	Order		
[10, 10]	1.486E-1	–	7.269E-2	–	16	3.812E-3
[20, 20]	1.129E-2	3.72	5.447E-3	3.74	26	4.205E-3
[40, 40]	8.478E-3	3.74	3.775E-4	3.99	31	1.404E-2
[80, 80]	5.703E-4	3.89	2.447E-5	4.00	36	4.107E-2
[160, 160]	3.617E-5	3.98	1.546E-6	4.00	39	0.238

The numerical results of the AMIB4-S scheme are reported in Table 4.4. It can be seen that the AMIB4-S scheme is accurate and efficient for problems without analytical solutions.

Example 5. This example compares the results of AMIB4-S and AMIB4-NS for solving Poisson's equation

$$u_{xx} + u_{yy} = e^{-x^2-\frac{y^2}{2}}(4x^2 + y^2 - 3) \tag{4.14}$$

subject to Robin boundary conditions:

$$\begin{aligned}
u + \frac{\partial u}{\partial n} &= (1 - \frac{\pi}{3})e^{-x^2-\frac{\pi^2}{18}} \quad \text{on } \Gamma_1, \\
u + \frac{\partial u}{\partial n} &= (1 + \frac{2\pi}{3})e^{-\frac{\pi^2}{9}-\frac{y^2}{2}} \quad \text{on } \Gamma_2, \\
u + \frac{\partial u}{\partial n} &= (1 + \frac{\pi}{3})e^{-x^2-\frac{\pi^2}{18}} \quad \text{on } \Gamma_3, \\
u + \frac{\partial u}{\partial n} &= (1 - \frac{2\pi}{3})e^{-\frac{\pi^2}{9}-\frac{y^2}{2}} \quad \text{on } \Gamma_4,
\end{aligned}$$

which are determined by analytical solution $u = e^{-x^2-\frac{y^2}{2}}$ on domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. Table 4.5 shows that the both approaches give good accuracy in solving the Poisson equation with Robin boundary conditions, but the iteration number of AMIB4-S is higher than that of AMIB4-NS.

Table 4.5: Example 5 –Fourth order numerical error analysis.

$[N_x, N_y]$	AMIB4-S					
	L_∞		L_2		iter no.	CPU time
	Error	Order	Error	Order		
[11, 11]	4.182E-4	–	3.017E-4	–	15	5.820E-3
[27, 27]	3.893E-6	5.21	2.161E-6	5.50	20	6.565E-3
[59, 59]	2.839E-7	3.35	1.177E-7	3.72	26	2.135E-2
[123, 123]	1.744E-8	3.80	8.023E-9	3.66	34	8.622E-2
[251, 251]	1.071E-9	3.92	5.162E-10	3.85	45	0.389
$[N_x, N_y]$	AMIB4-NS					
	L_∞		L_2		iter no.	CPU time
	Error	Order	Error	Order		
[11, 11]	6.064E-4	–	3.669E-4	–	16	2.919E-3
[27, 27]	8.515E-6	4.46	4.075E-6	4.71	21	6.771E-3
[59, 59]	2.791E-7	4.26	1.322E-7	4.27	22	1.952E-2
[123, 123]	1.291E-8	4.13	6.417E-9	4.07	26	6.725E-2
[251, 251]	6.967E-10	4.07	3.594E-10	4.02	33	0.302

Example 6. Consider Poisson's equation

$$u_{xx} + u_{yy} = -2e^x \sin(x + y) - e^x \cos(x + y) \tag{4.15}$$

on domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, subject to mixed boundary conditions determined by the exact solution $u = e^x \cos(x + y)$.

$$\begin{aligned}
u &= e^x \cos(x + \frac{\pi}{3}) && \text{on } \Gamma_1, \\
u + \frac{\partial u}{\partial n} &= e^{\frac{\pi}{3}}(2\cos(\frac{\pi}{3} + y) - \sin(\frac{\pi}{3} + y)) && \text{on } \Gamma_2, \\
u + \frac{\partial u}{\partial n} &= e^x(\cos(x + \frac{\pi}{3}) - \sin(x + \frac{\pi}{3})) && \text{on } \Gamma_3, \\
u_x &= e^{-\frac{\pi}{3}}(\cos(-\frac{\pi}{3} + y) - \sin(-\frac{\pi}{3} + y)) && \text{on } \Gamma_4.
\end{aligned}$$

Table 4.6 indicates that the AMIB4-S produces a fourth order of convergence. However, the iterative number increases dramatically as $N_x = N_y$ increases. So does the CPU time. Consequently, the AMIB4-S scheme becomes inefficient comparing with the AMIB4-NS scheme. For the AMIB4-NS scheme, the overall complexity is still on the order of $O(N \log N)$ for this 2D

problem with $N = N_x \times N_y$. In general, it is found that the AMIB4-S scheme fails to reach the desired efficiency for problems with mixed boundary conditions as in this example. Therefore, whenever the regular AMIB4-NS scheme is applicable, we recommend to use the regular AMIB method for both accuracy and efficiency.

Table 4.6: Example 6 –Fourth order numerical error analysis of AMIB4-S for mixed boundary conditions.

$[N_x, N_y]$	AMIB4-S					
	L_∞		L_2		CPU time(s)	iter no.
	Error	Order	Error	Order		
[11, 11]	1.245E-4	–	5.487E-5	–	3.980E-3	31
[27, 27]	5.075E-6	3.56	1.595E-6	3.94	1.209E-2	48
[59, 59]	2.542E-7	3.83	7.063E-8	3.99	4.603E-2	69
[123, 123]	1.414E-8	3.93	3.727E-9	4.00	0.233	108
[251, 251]	8.333E-10	3.97	2.147E-10	4.00	1.323	157
[507, 507]	5.402E-11	3.89	1.716E-11	3.59	8.193	224
$[N_x, N_y]$	AMIB4-NS					
	L_∞		L_2		CPU time(s)	iter no.
	Error	Order	Error	Order		
[11, 11]	4.883E-4	–	1.469E-4	–	3.946E-3	28
[27, 27]	1.480E-5	3.66	5.404E-6	3.46	8.879E-3	31
[59, 59]	6.514E-7	3.89	2.586E-7	3.79	2.810E-2	38
[123, 123]	3.457E-8	3.95	1.419E-8	3.90	0.120	50
[251, 251]	1.987E-9	3.98	8.276E-10	3.96	0.578	67
[507, 507]	9.724E-11	4.28	3.177E-11	4.62	3.522	96

4.3.3 Combined AMIB method

Example 7. In this example, we combine AMIB4-S and AMIB4-NS schemes to attack one type of problems for which high order fast Poisson solvers have never been reported. Practical applications sometimes lead to the consideration of a staggered boundary on one end but a non-staggered boundary on the other end. Since in the AMIB method, the boundary treatments at left and right ends are independent, the AMIB4-S and AMIB4-NS schemes can be simply employed at either end. The combined AMIB method is flexible and robust in handling any combination of grid setting.

We test our method on the following problem. On domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, the Poisson problem has

$$u_{xx} + u_{yy} = \cos(ky)e^y(-k^2 + 1) \tag{4.16}$$

with analytical solution $u = \cos(kx)e^y$. Suppose it is subject to boundary conditions and grid

setting as below:

$$
\begin{aligned}
u_y &= \cos(kx)e^{-\frac{\pi}{3}} \ \text{ on } \ \Gamma_1, \\
u &= \cos(\frac{k\pi}{3})e^{y} \ \text{ on } \ \Gamma_2, \\
u &= \cos(kx)e^{\frac{\pi}{3}} \ \text{ on } \ \Gamma_3, \\
u_x &= \sin(\frac{k\pi}{3})e^{y} \ \text{ on } \ \Gamma_4.
\end{aligned}
$$

In addition, we have staggered boundary condition imposed on the boundaries Γ_1 and Γ_4. The results in Table 4.7 indicate that the combined AMIB method achieves a fourth order convergence, while being efficient.

Table 4.7: Example 7 –Fourth order numerical error analysis of the AMIB4.

$[N_x, N_y]$	AMIB4					
	L_∞		L_2		iter no.	CPU time
	Error	Order	Error	Order		
[27, 27]	5.680E-3	–	2.427E-3	–	30	9.736E-3
[59, 59]	2.356E-4	4.07	9.742E-5	4.64	39	2.959E-2
[123, 123]	1.194E-5	4.06	4.863E-6	4.08	51	0.123
[251, 251]	6.719E-7	4.03	2.715E-7	4.04	69	0.572
[507, 507]	3.984E-8	4.01	1.604E-8	4.02	101	3.964

4.3.4 Comparison of two AMIB methods

Besides corrected difference that can restore the spatial accuracy across the interface, corrected fictitious value approach can also be used to introduce auxiliary variables to form augmented system in the AMIB method. It is thoroughly discussed in the section 2.4 for solving second order accurate solution of elliptic interface problem. The AMIB based on corrected fictitious value can also be used to solve Poisson BVP with expected high efficiency. In this subsection, we give a comparison of the two different AMIB methods for solving such Poisson BVP on rectangular domain.

Comparisons are carried out below for AMIB-CFD, AMIB-CFP and MIB on staggered mesh and traditional mesh. Besides, AMIB-CFP with the Schur complement explicitly formed is also implemented with the biconjugate gradient solver [62] replaced by the one in Slatec software. We name this one as AMIB-CFP-SLATEC. The iteration termination tolerance is set to be e^{-15} in all

biconjugate gradient solver. The selected tolerance is reasonable for guaranteeing the fourth order convergence. Let's solve 2D Poisson equation

$$u_{xx} + u_{yy} = -2k^2 \sin(kx)\cos(ky)$$

with exact solution $u = \sin(kx)\cos(ky)$, and $k = 5$.

The difference on accuracy or time of AMIB-CFD and AMIB-CFP is not significant for Dirichlet boundary condition on four boundaries. Below just lists one example with difference conditions on the four boundaries.

$$\begin{aligned}
u &= -\sin(kx)\sin(\frac{k\pi}{3}) && \text{on } \Gamma_1 \\
u + \frac{\partial u}{\partial n} &= (\sin(\frac{k\pi}{3}) + k\cos(\frac{k\pi}{3}))\cos(ky) && \text{on } \Gamma_2 \\
u + \frac{\partial u}{\partial n} &= (\cos(\frac{k\pi}{3}) - k\sin(\frac{k\pi}{3}))\sin(kx) && \text{on } \Gamma_3 \\
u_x &= k\cos(\frac{k\pi}{3})\cos(ky) && \text{on } \Gamma_4.
\end{aligned}$$

Below four tables are listed. The preconditioner used for biconjugate gradient squared method in SLATEC is Incomplete LU preconditioner. Two conclusion are drawn below:

Firstly, from table 4.8 and 4.9, we can see AMIB-CFP-SLATEC does not improve the computing time. By efficiency, AMIB-CFD> AMIB-CFP > MIB > AMIB-CFP-SLATEC . From my understanding, forming explicit coefficient matrix $D - CA^{-1}B$, and solving matrix system for AMIB-CFP-SLATEC both take extra too much time and storage such that total time expense is higher than the first three methods with iterative approach in Schur complement or for the whole linear system. One aspect is that $A^{-1}B$ will uses FFT as many times as multiple of interface point or fictitious point number, while in iterative solver FFT is used as many times as the iteration number. Normally, iteration number is fairly small compared with no. of interface point or fictitious point number. On the other hand, the biconjugate solver in SLATEC software is does not improve computing speed of biconjugate gradient iterative solver in [62] much.

Secondly, from table 4.10 and 4.11, the K_2 condition number of AMIB-CFD is much bigger than that of AMIB-CFP, which is opposite to our expectation.

Is the condition number of explicit coefficient matrix $D - CA^{-1}B$ the same as the condition number in iteration process for product $(D - CA^{-1}B)x$ during atimes and asolve? As I observed, the iteration numbers are different between $(D - CA^{-1}B)x$ and $(D - CA^{-1}B)x = Dx - CA^{-1}Bx$ for iterations, where $(D - CA^{-1}B)x$ means $D - CA^{-1}B$ is one formed explicit matrix block and $(D - CA^{-1}B)x = Dx - CA^{-1}Bx$ is the process described in the chapter for second AMIB method. This might matter in the unexpected table 4.10, 4.11. So the condition numbers reported in tables 4.10 and 4.11 are for formed explicit matrix. The AMIB-CFD has a larger condition number, but converges faster than the AMIB-CFP. This further confirms that the actual computing process of AMIB through iterative approach is not determined by the condition number of the explicitly formed Schur complement matrix. It is not clear what is causing AMIB-CFP to converge slower than AMIB-CFD.

Table 4.8: Example for staggered mesh

$[N_x, N_y]$	AMIB-CFD,$TOL = e^{-15}$					
	L_∞		L_2		iter no.	time(s)
	Error	Order	Error	Order		
[59, 59]	1.204e-5	3.89	4.7444-6	3.92	38	9.36e-2
[123, 123]	6.434E-7	3.89	2.486E-7	3.92	43	0.374
[251, 251]	3.721E-8	3.91	1.454E-8	3.95	48	1.59
[507, 507]	2.236E-9	3.91	8.872e-10	3.98	51	7.35
$[N_x, N_y]$	AMIB-CFP,$TOL = e^{-15}$					
	L_∞		L_2			
	Error	Order	Error	Order		
[59, 59]	5.570e-5	–	9.547E-6	–	153	0.312
[123, 123]	3.100E-6	3.88	5.319E-7	3.88	236	1.825
[251, 251]	1.821E-7	3.95	3.134E-8	3.94	362	10.80
[507, 507]	1.102E-8	3.98	1.901E-9	3.97	539	74.83
$[N_x, N_y]$	MIB,$TOL = e^{-15}$					
	L_∞		L_2			
	Error	Order	Error	Order		
[59, 59]	5.570E-5	–	9.550E-6	–	476	0.360
[123, 123]	3.101E-6	3.88	5.319E-7	3.88	967	2.667
[251, 251]	1.821E-7	3.95	3.134E-8	3.95	1975	21.75
[507, 507]	1.102E-8	3.98	1.901E-9	3.97	3942	181.95
$[N_x, N_y]$	Corrected FP by SLATEC,$TOL = e^{-5}$					
	L_∞		L_2			
	Error	Order	Error	Order		
[59, 59]	5.570E-5	–	9.550E-6	–	–	2.98
[123, 123]	3.101E-6	3.88	5.319E-7	3.88	–	26.66

Table 4.9: Example for traditional mesh

$[N_x, N_y]$	AMIB-CFD,$TOL = e^{-15}$					
	L_∞		L_2		iter no.	time(s)
	Error	Order	Error	Order		
[59, 59]	4.036e-5	–	1.045e-5	–	33	7.80e-2
[123, 123]	2.210E-6	3.91	5.703E-7	3.91	40	0.327
[251, 251]	1.289E-7	3.96	3.331E-8	3.96	54	1.84
[507, 507]	7.778E-9	3.98	2.012e-9	3.98	73	10.39
$[N_x, N_y]$	AMIB-CFP,$TOL = e^{-15}$					
	L_∞		L_2			
	Error	Order	Error	Order		
[59, 59]	6.952e-5	–	2.298E-5	–	212	0.421
[123, 123]	4.185E-6	3.78	1.360E-6	3.80	371	2.761
[251, 251]	2.534E-7	3.91	8.184E-8	3.92	697	20.76
[507, 507]	1.467E-8	4.04	4.598E-9	4.08	1293	181.28
$[N_x, N_y]$	MIB,$TOL = e^{-15}$					
	L_∞		L_2			
	Error	Order	Error	Order		
[59, 59]	6.952E-5	–	2.299E-5	–	802	0.514
[123, 123]	4.185E-6	3.78	1.360E-6	3.80	1683	4.48
[251, 251]	2.534E-7	3.91	8.185E-8	3.92	3741	41.54
[507, 507]	1.555E-8	3.96	5.009E-9	3.96	7616	344.64
$[N_x, N_y]$	Corrected FP by SLATEC,$TOL = e^{-5}$					
	L_∞		L_2			
	Error	Order	Error	Order		
[59, 59]	6.952E-5	–	2.299E-5	–	–	3.01
[123, 123]	4.185E-6	3.78	1.360E-6	3.80	–	27.09

Table 4.10: Condition number on STAGGERED mesh

$[N_x, N_y]$	AMIB-CFD	AMIB-CFP
	K_2	K_2
[17, 17]	3396947.77223732	762.167796634209
[33, 33]	4105648767.1682	2517.31682965568
[65, 65]	2233279876335.98	6451.51612800563

Table 4.11: Condition number on TRADITIONAL mesh

$[N_x, N_y]$	AMIB-CFD	AMIB-CFP
	K_2	K_2
[27, 27]	1984676.39782731	6456.20825807497
[59, 59]	3538943676.14228	36762.6935579654
[123, 123]	2104705219581.62	176246.160469646

4.3.5 AMIB for problem with low regularities

In solving the Poisson BVP, AMIB assumes high order regularity on the given domain. We are further interested in its performance in solving problem with low regularity. Consider Laplace's equation in upper half-plane, i.e., $y > 0$, with

$$u(x,0) = \begin{cases} 0 \text{ if } x > 0, \\ 1 \text{ if } x < 0, \end{cases}$$

A solution is

$$u(x, y) = \frac{1}{\pi} \tan^{-1} \frac{y}{x} = \frac{\theta}{\pi},$$

where θ is the angle in the radians measured from the x-axis. The normal derivative of the solution along the boundary is the derivative with respect to y. We have

$$\frac{\partial u(x, y)}{\partial y} = \frac{1}{\pi} \frac{1/x}{1 + (y/x)^2} = \frac{x}{\pi(x^2 + y^2)}.$$

At the boundary, we have the normal derivative

$$u_y(x, 0) = \frac{1}{\pi x},$$

which is unbounded at the point of discontinuity in the boundary date function. The normal derivative is unbounded at the point where the tangential derivative of the boundary date is unbounded. This example is borrowed from [65]. It can be observed that the solution is discontinuous at $x = 0$ along x-axis, causing low regularity near the point $(0, 0)$. It make the assumption of applying central difference method invalid. However, we test second order central difference method to solve the concerned problem. The computation domain is set as $[-1, 1] \times [0, 2]$ with uniform partitioning number in each direction. To avoid setting the point $(0, 0)$ on a grid point, we choose the partition number N_x or N_Y as an odd number for traditional grid setting as in table 4.12. A simple Dirichlet boundary condition is given by the analytical solution. For the second order discretization, the boundary condition is analytically plugged into the linear system. From table 4.12, we can see that first order accuracy can be achieved under L_2

Table 4.12: Numerical error analysis of MIB2 on traditional grids.

$[N_x, N_y]$	MIB2			
	L_∞		L_2	
	Error	Order	Error	Order
[19, 19]	0.0341	–	2.7E-3	–
[39, 39]	0.0341	0	1.4E-3	0.95
[79, 79]	0.0341	0	6.872E-4	1.03
[159, 159]	0.0341	0	3.436E-4	1.00

norm, but it is not convergent under L_∞ norm. The maximum numerical error occurs at grid point near $(0, 0)$, and persists even though finer mesh is used. As a comparison, staggered grid is also for numerical solution, which is shown in table.4.13. The N_x or N_y in this table indicates the grid number inside the give domain. Similar accuracy reduction is observed in table 4.13.

Table 4.13: Numerical error analysis of MIB2 on staggered grids.

$[N_x, N_y]$	MIB2			
	L_∞		L_2	
	Error	Order	Error	Order
[18, 18]	1.98E-2	–	2.24E-3	–
[38, 38]	1.98E-2	0	1.06E-3	1.08
[78, 78]	1.98E-2	0	5.19E-4	1.03
[158, 158]	1.98E-2	0	2.56E-4	1.02

The order reduction from above two approach concerned with second order accuracy predicts that high order scheme will not provide desired high order accuracy when solving above problem.

Table 4.14: Numerical error analysis of MIB4 on traditional grids.

$[N_x, N_y]$	MIB4			
	L_∞		L_2	
	Error	Order	Error	Order
[17, 17]	3.65E-2	–	5.40E-3	–
[37, 37]	3.39E-2	0.11	2.64E-3	1.03
[77, 77]	3.27E-2	0 .05	1.30E-3	1.02
[157, 157]	3.21E-2	0.03	6.48E-4	1.00

Table 4.15: Numerical error analysis of MIB4 on staggered grids.

$[N_x, N_y]$	MIB4			
	L_∞		L_2	
	Error	Order	Error	Order
[16, 16]	1.56E-2	–	1.53E-3	–
[36, 36]	1.56E-2	0	6.82E-4	1.17
[76, 76]	1.56E-2	0	3.23E-4	1.08
[156, 156]	1.56E-2	0	1.57E-4	1.04

The numerical results in table 4.14 and 4.15 show the accuracy reduction of MIB4 for solving the above low regularity problem. Only first order accuracy under L_2 norm is observed. Such low accuracy even with high order scheme makes the further discussion on application of AMIB unnecessary. In order for AMIB practical for solving such low regularity problem, robust and high accurate of MIB method need to be investigated. This can be a future algorithm study for us.

CHAPTER 5

FOURTH ORDER AMIB FOR ELLIPTIC INTERFACE PROBLEM

This chapter is dedicated to a fourth order accurate AMIB method for solving elliptic interface problem, based on the development of second order AMIB in chapter two and the fast algorithm for solving Poisson BVP on rectangular domain in chapter three.

5.1 Preliminary

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. 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 [32, 54, 72] 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 $O(N^2)$. 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. Several multigrid interface algorithms have been successfully developed for elliptic interface problems, including IIM [2, 50], piecewise-polynomial interface method [15], ghost point method [18], and virtual node method [8, 35]. Beside, some augmented system works [24, 45, 72]

by introducing auxiliary variables also contribute to efficient elliptic interface algorithm. 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. The objective of this section is to discuss 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 $O(N \log N)$.

5.2 Theory and algorithm

For elliptic interface problems with piecewise constant coefficients, the original PDE (1.1) can be rewritten as below after moving the coefficient β to the right hand side,

$$\triangle u = -\frac{f(x,y)}{\beta}, \quad (x,y) \in (\Omega^- \cup \Omega^+) \setminus \Gamma, \tag{5.1}$$

with the same Dirichlet boundary conditions (1.2) and interface jump conditions (1.3) and (1.4) imposed.

We are concerned with solving (5.1) 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

$$x_i = a + ih_x, \; y_j = c + jh_y, \; i = 0, \cdots, m, \; j = 0, \cdots, n. \tag{5.2}$$

This work aims to develop a fourth order AMIB method for Poisson interface problem (5.1), 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 [85] will be formulated in the augmented approach [24] to treat smoothly curved interfaces. For the exterior part, the fourth order AMIB scheme introduced in [26] 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 [26], the first step of our AMIB algorithm is to introduce an immersed boundary.

5.2.1 Immersed boundary

As shown in last sections, 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 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

$$\begin{aligned} R = \{(x_i, y_j) | x_i = a + (i-2)h_x,\ y_j = c + (j-2)h_y, \\ i = 0, \cdots, m+4,\ j = 0, \cdots, n+4\}, \end{aligned} \tag{5.3}$$

where h_x, h_y are defined as above. One may refer to Fig.5.1 for the setting of grid nodes (5.3). From now on, we denote the extended subdomain as Ω^e, yielding a new computational domain composed of three subdomains:

$$B = \Omega^+ \cup \Omega^- \cup \Omega^e. \tag{5.4}$$

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

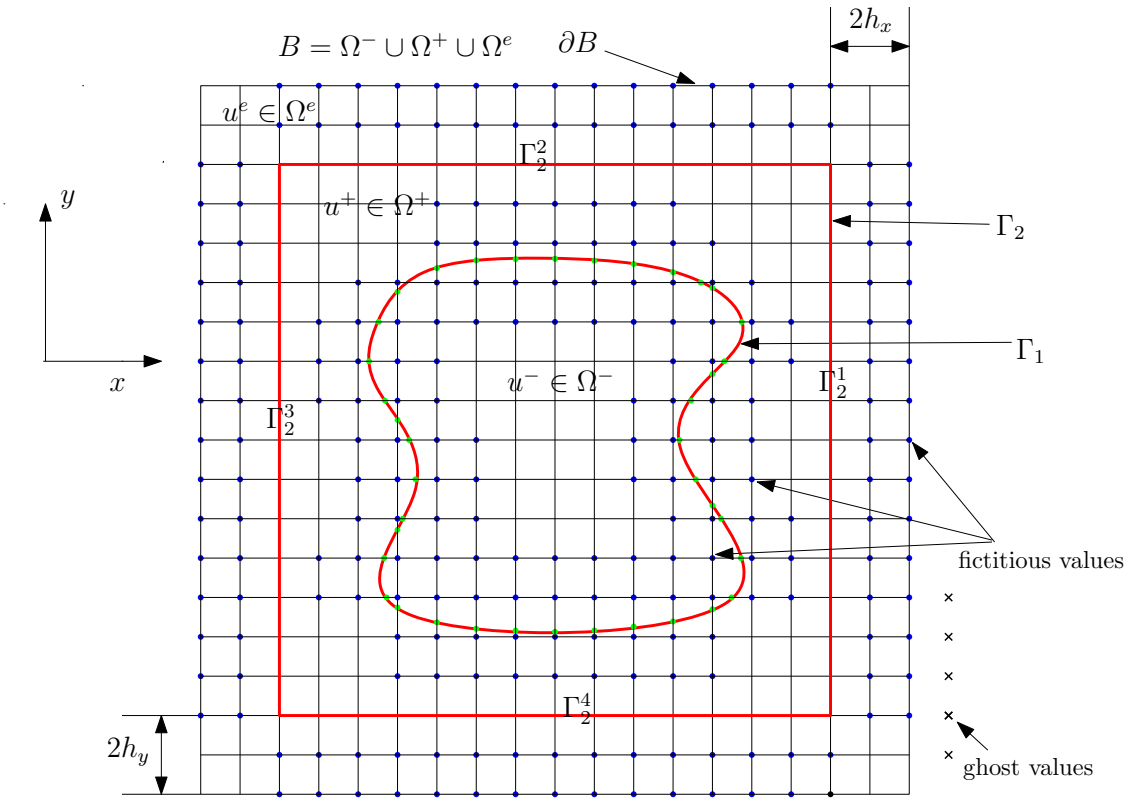

Figure 5.1: The original problem is recasted into an immersed boundary problem. The red curves and lines indicate the original and introduced interfaces, respectively. Blue dots stand for the needed fictitious points for correcting the fourth order central difference.

defined on the introduced interface $\Gamma_2 = \Omega^+ \cap \Omega^e$:

$$[\![u]\!] := u^+ - u^e = g(x, y). \tag{5.5}$$

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

$$\triangle u = -\frac{f(x, y)}{\beta}, \quad (x, y) \in B \setminus \Gamma, \tag{5.6}$$

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 (1.3) - (1.4) and the immersed interface condition (5.5). Besides, a Dirichlet boundary condition is trivially assumed for the zero-padding solution

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

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. Corrected difference 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.5.1 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.5.1, 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.5.1. A rather tight grid setting is shown in Fig.5.1 to save the space.

5.2.2 Laplacian approximation

Augmented approach will be utilized 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., fourth order corrected central difference via Cartesian derivative jumps and reconstructing Cartesian derivative jumps.

Corrected central difference

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.

Cartesian derivative jumps

To apply corrected differences approximation to Laplacian at the irregular points near the interface $\Gamma = \Gamma_1 \cup \Gamma_2$, derivative jumps should be approximated. AMIB method adopts polynomials with aid of fictitious values for such approximation as shown in formula (3.32). Figure 5.2 shows the needed two side approximation for interface Γ_1 and Γ_2 for our fourth order study.

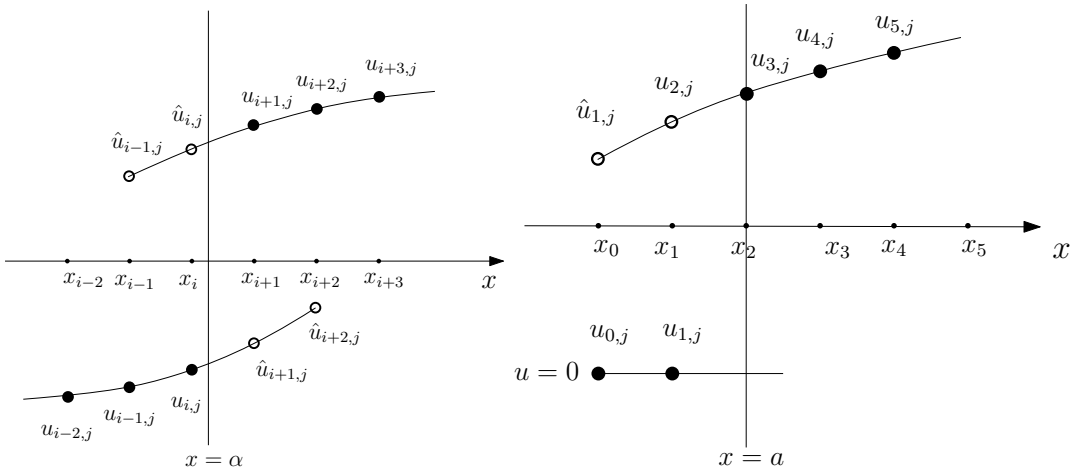

Figure 5.2: The numerical approximation to the derivative jumps for the two types of interfaces. The left one is for Γ_1, while the right one is for the boundary Γ_2. In both charts, filled circles stand for real values while empty circles denote fictitious values.

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. (2.9). Two scenarios are shown in Fig.5.3. 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.5.3, Eq. (2.9) 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 (2.9). 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.(2.9). Again, in case of only one grid point in between α_1 and α_2, a grid refinement is needed.

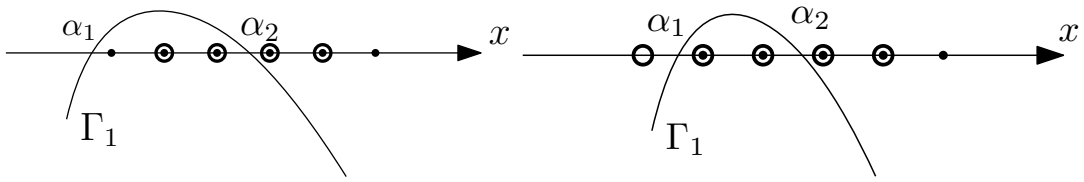

Figure 5.3: Derivative jump approximation in two scenarios. Empty circles denote the fictitious values at the grid points, while black dots indicate real function values.

5.2.3 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. In the MIB method, such fictitious value is generated by rigorously imposing jump conditions [85] or boundary conditions [81]. In the present study, there are two types of interfaces Γ_1 and Γ_2. Their MIB treatments require different strategies.

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 (1.3) and (1.4) together with one more jump condition, which is analytically derived by differentiating Eq. (1.3) along the tangential direction τ of the interface as below,

$$[\![u_\tau]\!] = u_\tau^+ - u_\tau^- = \rho(x, y). \tag{5.7}$$

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. 5.4 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

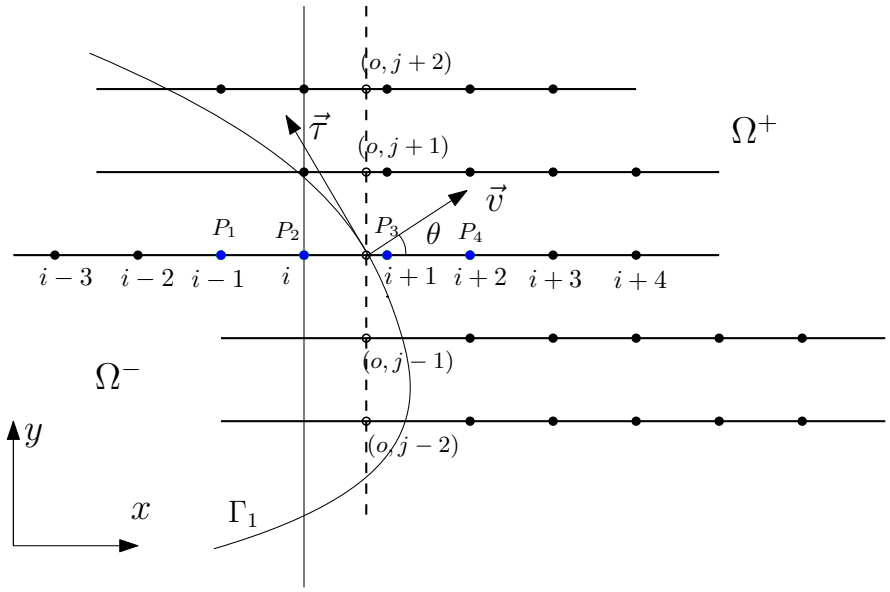

Figure 5.4: Consider the case where the interface intersects $y = y_j$ at (x_o, y_j). At four irregular points $P_1(i-1, j), P_2(i, j)$, $P_3(i+1, j)$, and $P_4(i+2, j)$, fictitious values (in blue color) can be constructed. Here θ is the angle between positive $x-$ direction and the normal vector $\vec{v}$.

Cartesian form [85],

$$[\![u]\!] = u^+ - u^- = \phi(x^*, y^*), \tag{5.8}$$

$$[\![u_\tau]\!] = (-u_x^+ \sin\theta + u_y^+ \cos\theta) - (-u_x^- \sin\theta + u_y^- \cos\theta) = \rho(x^*, y^*), \tag{5.9}$$

$$[\![\beta u_v]\!] = \beta^+(u_x^+ \cos\theta + u_y^+ \sin\theta) - \beta^-(u_x^- \cos\theta + u_y^- \sin\theta) = \psi(x^*, y^*), \tag{5.10}$$

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 (5.11) or (5.12) can be obtained with three of the four quantities remaining. If u_y^- is eliminated, the following equations can be derived

$$[\![u]\!] = u^+ - u^-, \;\; [\![\beta u_v]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ u_x^+ - C_x^- u_x^- + C_y^+ u_y^+, \tag{5.11}$$

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

$$[\![u]\!] = u^+ - u^-, \;\; [\![\beta u_v]\!] + \beta^- \cot\theta [\![u_\tau]\!] = D_y^+ u_y^+ - D_y^- u_y^- + D_x^+ u_x^+, \tag{5.12}$$

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 [79, 85], 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. (5.11) and (5.12) to determine all fictitious values. Moreover, Eqns. (5.11) and (5.12) enable us to treat 2D jump conditions only along Cartesian directions. For example, if u_y^+ is known or can be numerically determined in (5.11), then Eq. (5.11) involves only one-dimensional (1D) jump conditions in x direction. Similarly, after approximating u_x^+ in the perpendicular direction, Eq. (5.12) 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.5.4 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 (5.11) are discretized as below

$$[\![u]\!] = W_0^+ U^+ - W_0^- U^- \tag{5.13}$$

$$[\![\beta u_v]\!] - \beta^- \tan\theta [\![u_\tau]\!] = C_x^+ W_1^+ U^+ - C_x^- W_1^- U^- + C_y^+ P^+ U_\Gamma, \tag{5.14}$$

with the following vector representations

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

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

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

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

The discretization in (5.13) and (5.14) 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 (5.11) 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 5.4. Additionally, each of these five auxiliary values are interpolated or extrapolated by five function values in Ω^+. For example, in Fig. 5.4, $u_{\Gamma,j+1}$ 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 $u_{\Gamma,j+2}, u_{\Gamma,j}, u_{\Gamma,j-1}$ and $u_{\Gamma,j-2}$ will be extrapolated by five function values from their right side in the Ω^+ domain.

With the function values in U_Γ appropriately approximated, (5.13) and (5.14) 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

$$\begin{aligned}\hat{u}_{i,j} &= \sum_{(x_I,y_J)\in\mathbb{S}_{i,j}} W_{I,J}u_{I,J} + W_0[\![u]\!] + W_1[\![u_\tau]\!] + W_2[\![\beta u_\nu]\!] \\ &= \sum_{(x_I,y_J)\in\mathbb{S}_{i,j}} W_{I,J}u_{I,J} + W_0\phi(x^*,y^*) + W_1\rho(x^*,y^*) + W_2\psi(x^*,y^*),\end{aligned} \tag{5.15}$$

where $\mathbb{S}_{i,j}$ represents a set of surrounding nodes involved in (5.13) and (5.14).

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 (5.15).

In this manner, four layers of fictitious values can be generated by repeatedly enforcing the jump conditions (5.11) and (5.12). One may refer to [85] for more details.

Fictitious values near interface Γ_2.

Generation of fictitious values required near the interface Γ_2 is easily done by following the treatment discussed in section for Poisson BVP on usual grid.

5.2.4 Formulation of augmented system

The utilization of fourth order corrected differences together with the Cartesian derivative jumps introduced as auxiliary variables can form augmented system like in (3.38)

$$KW = R. \tag{5.16}$$

With the augmented system formulated, Schur complement approach can again be used for fast fourth order solution U of elliptic interface problem.

5.3 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 [85], as well as the second order AMIB (AMIB2) method developed in [24].

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,

$$\begin{aligned} &\frac{\partial u_h}{\partial x}(x_i, y_j) \\ &\quad \triangleq -\frac{1}{12}u_h(x_{i-2}, y_j) + \frac{4}{3}u_h(x_{i-1}, y_j) - \frac{5}{2}u_h(x_i, y_j) + \frac{4}{3}u_h(x_{i+1}, y_j) - \frac{1}{12}u_h(x_{i+2}, y_j). \end{aligned} \tag{5.17}$$

When the fourth order central difference stencil crosses the interface, we need to substitute the corresponding fictitious value into (5.17) as in the usual MIB scheme [85]. 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 [62] 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.

5.3.1 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

$$u(x,y) = \begin{cases} e^{-x^2-0.5y^2}, & r \le 0.5, \\ \sin(kx)\sin(ky), & \text{otherwise}, \end{cases} \tag{5.18}$$

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. 5.5 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 5.1. 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.

Table 5.1: 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

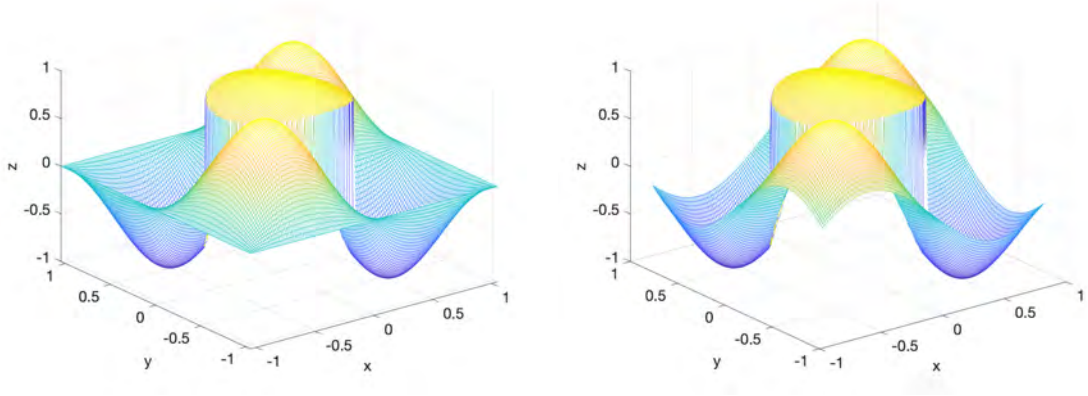

Figure 5.5: The computed solution of Example 1 (left) and Example 2 (right) on a mesh with $n = 128$.

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. 5.5 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 [24].
By considering $\beta^+ = 100$ and $\beta^- = 1$, the numerical errors of the AMIB2, AMIB4, and MIB4 are reported in Table 5.2. 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 [26], 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.

Table 5.2: 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 5.3: 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

We next increase the diffusion coefficient to be $\beta^+ = 1000$ and $\beta^- = 1$, while keeping the other parameters unchanged. In Table 5.3, the numerical results from the AMIB4, AMIB2, and MIB4

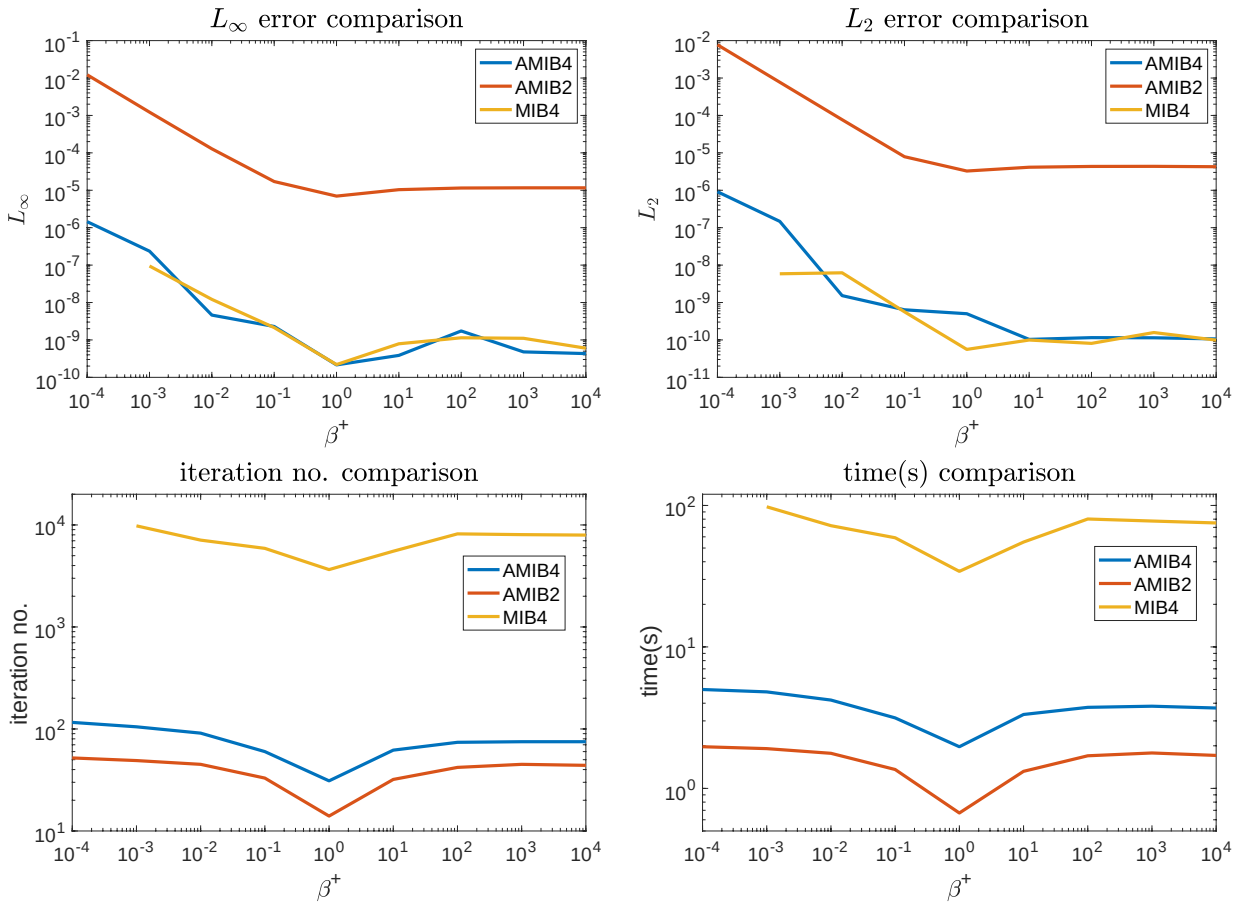

Figure 5.6: Impact of highly contrasted coefficients to the MIB and AMIB methods for the circle interface problem. With a fixed $\beta^- = 1$, β^+ ranges from 10^{-4} to 10^4. Here, the mesh size for the AMIB4, AMIB2, and MIB4, respectively, is taken as $n = 508$, $n = 512$ and $n = 508$.

are displayed. By comparing with Table 5.2, we found that the changes in both accuracies and efficiency are very minor.

Following the studies in [7], 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. 5.6, 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 [7] 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

$$u(x, y) = \begin{cases} \sin(kx)\cos(ky), & \text{inside } \Gamma, \\ \cos(kx)e^y, & \text{otherwise}, \end{cases} \tag{5.19}$$

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 5.4. 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. 5.7. 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 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.

Table 5.4: 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

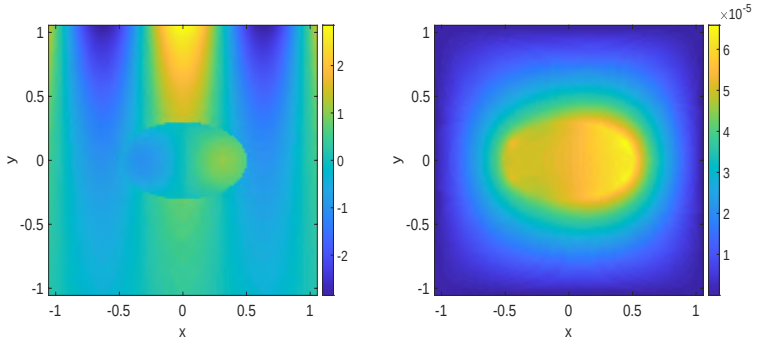

Figure 5.7: The numerical solution (left) and error (right) of Example 3a on a mesh with $n = 124$.

We also want to investigate the performance of the AMIB4 for treating other types of boundary conditions. Like the original AMIB4 scheme [26], the present AMIB4 method is able to handle any type of general boundary conditions. Moreover, 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

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

where $\vec{v}$ denotes the outer normal direction towards the rectangular domain. The numerical results are presented in Table 5.5. 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 [26]. 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.

Table 5.5: 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

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.

The numerical results are presented in Table 5.6. 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. 5.8. Large errors are clustered around the interface Γ_1, particularly when the normal direction is not align 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.

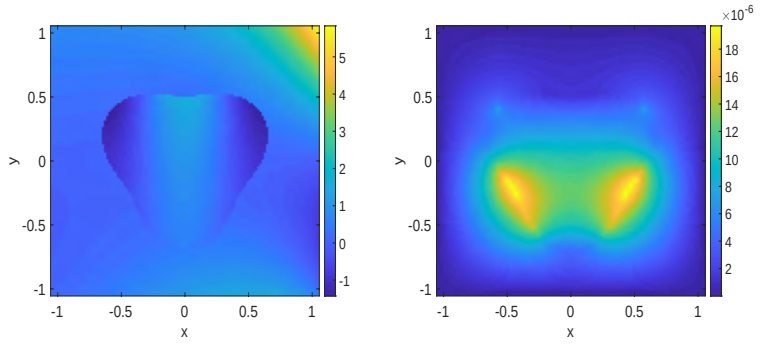

Figure 5.8: The numerical solution (left) and error (right) of Example 4 in on a mesh with $n = 124$.

Table 5.6: 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

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

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

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 5.7 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. 5.9 shows that errors are large in four out of five leafs, but it is small for the top leaf. We believe this is due to the solution, see Fig. 5.9 left. The solution changes rapidly near the interface for the other four leafs, but such change is relatively slower for the top leaf.

Table 5.7: 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

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

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

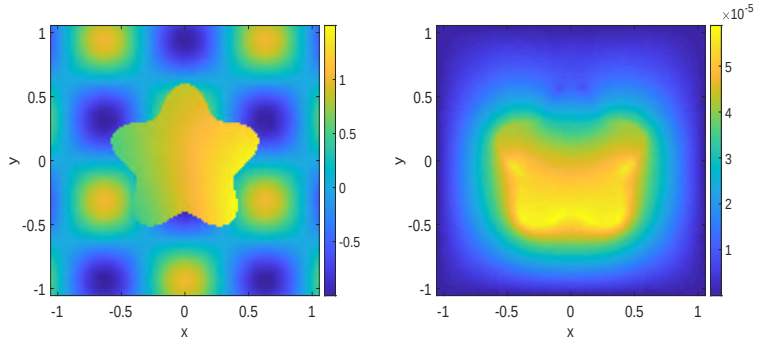

Figure 5.9: The numerical solution (left) and error (right) of Example 5 on a mesh with $n = 124$.

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 interface:

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

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 5.8. 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. 5.10, 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.

Table 5.8: 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

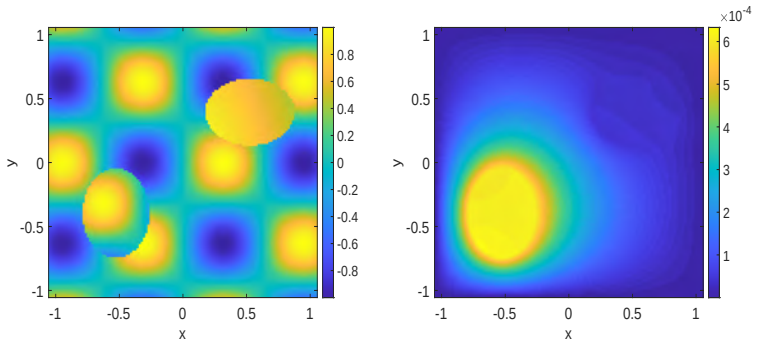

Figure 5.10: The numerical solution (left) and error (right) of Example 6 on a mesh with $n = 124$.

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

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

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 5.9. 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 [21]. In Table 5.9, 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 (5.17) 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.

Table 5.9: 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

5.3.2 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. 5.11 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. 5.11. 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.

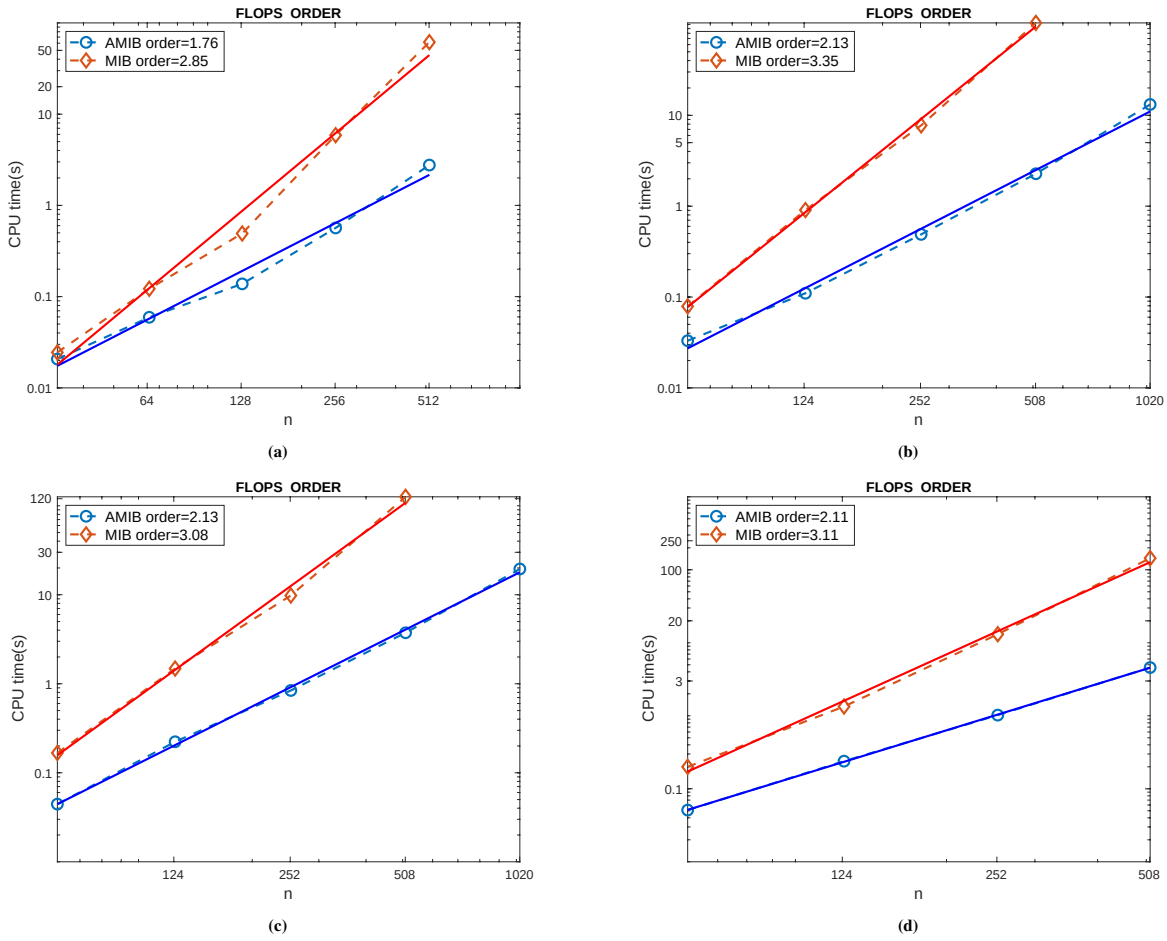

Figure 5.11: Flops order in CPU time is examined for several examples. Here n represents the degree of freedom in each direction.

CHAPTER 6

AMIB FOR PARABOLIC INTERFACE PROBLEM

The preceding chapters discussed AMIB method for solving a series of elliptic problems with FFT accelerating the computing speed. However, it is not easy to generalize FFT algorithm for solving non-constant coefficient parabolic interface problem. To design a robust and efficient algorithm for solving such problem, attempts have been made in different directions. An AMIB method based on multigird method is proposed, and its efficiency is extensively tested. It shows significant progress compared to MIB in achieving high efficiency for solving parabolic interface problem.

6.1 Theory and algorithm

This section is focused on the following two dimensional (2D) parabolic interface problem

$$\frac{\partial u}{\partial t} = \nabla \cdot (\beta \nabla u) + f, \quad \text{in } \Omega \subset \mathbb{R}^2 \tag{6.1}$$

with Dirichlet boundary conditions of u imposed along the boundary $\partial\Omega$ of a given rectangular domain. The whole domain is separated by an embedded interface into two subdomains such that $\Gamma = \Omega^+ \cap \Omega^-$, and $\Omega = \Omega^+ \cup \Omega^-$. The diffusion coefficient β is defined to be a piecewise constant in different subdomains. Across the interface Γ, the function values of u from the two subdomains are associated by the following jump conditions

$$[\![u]\!] := u^+ - u^- = \phi(x, y, t), \tag{6.2}$$

$$[\![\beta u_n]\!] := \beta^+ \nabla u^+ \cdot \vec{n} - \beta^- \nabla u^- \cdot \vec{n} = \psi(x, y, t), \tag{6.3}$$

where superscripts + and − on u and β signify the corresponding subdomains, $\vec{n}$ indicates the normal direction pointing from Ω^- to Ω^+, and the jump values ϕ and ψ are allowed to be temporal and spatial dependent to account for the most general scenario.

So far, we are concerned with a good approximation to u_{n+1} resulting from FFT treatment for uniform constant coefficients β. This is again reduced to a Poisson problem in the one of the subdomains. The regular Runge-Kutta methods may give an accurate approximation, but it is conditionally stable such that the approximation by Runge-Kutta methods causes several stability problem. To avoid that, very small time step is required, which is not efficient.

In order to design stable and accurate schemes. We consider second order trapezoid splitting algorithms through two dimensional Poisson equation:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f = F, \tag{6.4}$$

First, the right side F of Poisson equation is decomposed into several component functions to make the application of ODE problems easier on the individual F_i compared to the whole F. A simple one is via $F = F_1 + F_2$, where $F_1 = u_{xx} + 0.5 * f$ and $F_2 = u_{yy} + 0.5 * f$. Then based on this decomposition, one trapozoidal splitting scheme is introduced as

$$v_0 = u_n \tag{6.5}$$

$$v_1 = v_0 + \frac{1}{2} dt \cdot F_1(t_n, v_0) \tag{6.6}$$

$$v_2 = v_1 + \frac{1}{2} dt \cdot F_2(t_n, v_1) \tag{6.7}$$

$$v_3 = v_2 + \frac{1}{2} dt \cdot F_2(t_{n+1}, v_3) \tag{6.8}$$

$$v_4 = v_3 + \frac{1}{2} dt \cdot F_1(t_{n+1}, v_4) \tag{6.9}$$

$$v_{n+1} = v_4 \tag{6.10}$$

With the above steps, the iteration is implemented to achieve the numerical solution at certain stopping time.

Generally, the trapezoidal splitting method can be viewed as a forward Euler step followed by a backward Euler step. The above scheme gives two explicit steps and two implicit steps. For the explicit steps (6.6) (6.7), it is computationally cheap, while the two implicit steps (6.8) (6.9) are contributing the most of computing time. The middle four steps are all related to dimension by dimension calculation. For the general Poisson problem, the scheme is second order accurate in time, and unconditionally stable. However, for the interface problem, the two implicit steps (6.8) (6.9) involve treatment on interface, which may be similar to the proposed MIB-based ADI scheme. Such MIB-based ADI scheme will be explored elsewhere. We note that the FFT algorithm cannot be implemented in dimension splitting approaches.

As discusses above, splitting approach may not be feasible to solve the interface problem. We may consider a different approach as follows. We focus on the piecewise constant coefficient parabolic interface problem.

$$\frac{\partial u}{\partial t} = \beta \Delta u + f, \quad \text{in } \Omega^+ \cup \Omega^- \setminus \Gamma. \tag{6.11}$$

or

$$\frac{1}{\beta}\frac{\partial u}{\partial t} = \Delta u + \frac{f}{\beta}, \quad \text{in } \Omega^+ \cup \Omega^- \setminus \Gamma. \tag{6.12}$$

Subject to the same initial, boundary, and interface conditions, the solution to (6.12) is equivalent to that of (6.1). We partition the given rectangular domain $\Omega = [a, b] \times [c, d]$ into to n_x and n_y equally spaced intervals in $x-$ and $y-$ directions respectively. The mesh size in each direction can then be defined as $h_x = (b - a)/n_x$ and $h_y = (d - c)/n_y$. For simplicity, we assume that $h_x = h_y = h$. The grid nodes are hence defined as

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

Moreover, uniform time increment $\triangle t$ is adopted for temporal discretization. Let $t_n = n \cdot \triangle t$

represent the nth time step in the temporal discretization. For ease of exposition, we use notation $u_{i,j}^n$ to denote the function value on grid (x_i, y_j) at time t_n.

We first consider a simple algorithm with FFT algorithm embedded for high efficiency. Suppose $\beta^+ > \beta^-$, then in Ω^+, a predictor-correcter algorithm is constructed. First, implicit Euler scheme is used to discretize the parabolic equation, yielding to

$$\frac{u^{n+1} - u^n}{\triangle t} = \beta^+ Du^{n+1} + f^{n+1}, \tag{6.13}$$

where $Du = (\delta_{xx} + \delta_{yy})u$, and δ denotes second order spatial central difference. On the other hand, in Ω^-, explicit Euler scheme is employed to obtain following with first order temporal truncation error $O(\triangle t)$,

$$\frac{u^{n+1} - u^n}{\triangle t} = \beta^- Du^n + f^n. \tag{6.14}$$

Besides, implicit Euler scheme is also adopted to get another discretization formula

$$\frac{u^{n+1} - u^n}{\triangle t} = \beta^- Du^{n+1} + f^{n+1}, \tag{6.15}$$

which can be rewritten as

$$\frac{1}{\beta^-}u^{n+1} = \frac{1}{\beta^-}u^n + \triangle t Du^{n+1} + \frac{\triangle t}{\beta^-}f^{n+1}.$$

Equivalently, we have

$$\frac{1}{\beta^+}u^{n+1} + (\frac{1}{\beta^-} - \frac{1}{\beta^+})u^{n+1} = \frac{1}{\beta^-}u^n + \triangle t Du^{n+1} + \frac{\triangle t}{\beta^-}f^{n+1}. \tag{6.16}$$

Approximation the u^{n+1} term in (6.16) by (6.14), we have

$$\frac{1}{\beta^+}u^{n+1} + (\frac{1}{\beta^-} - \frac{1}{\beta^+})(u^n + \beta^- \triangle t Du^n + \triangle t f^n) = \frac{1}{\beta^-}u^n + \triangle t Du^{n+1} + \frac{\triangle t}{\beta^-}f^{n+1}. \tag{6.17}$$

As a consequence, above equation can be rewritten as

$$\begin{aligned}
&\frac{1}{\beta^+}u^{n+1} - \triangle t D u^{n+1} \\
&= \frac{1}{\beta^+}u^n + (\frac{\beta^- - \beta^+}{\beta^+})\triangle t D u^n + \triangle t(\frac{1}{\beta^+} - \frac{1}{\beta^-})f^n + \triangle t\frac{1}{\beta^-}f^{n+1} \qquad (6.18)\\
&\triangleq \frac{1}{\beta^+}u^n + F^-. \qquad (6.19)
\end{aligned}$$

Equations (6.15) and (6.19) can be combined into a uniform expression

$$\frac{1}{\beta^+}u^{n+1} - \triangle t D u^{n+1} = \frac{1}{\beta^+}u^n + F, \qquad (6.20)$$

where

$$F = \begin{cases} F^+ := \frac{1}{\beta^+}\triangle t f^{n+1} \\ F^- := (\frac{\beta^- - \beta^+}{\beta^+})\triangle t D u^n + \triangle t(\frac{1}{\beta^+} - \frac{1}{\beta^-})f^n + \triangle t\frac{1}{\beta^-}f^{n+1} \end{cases}, \qquad (6.21)$$

Note that if $\beta^- > \beta^+$, redefine F^+ as $(\frac{\beta^+ - \beta^-}{\beta^-})\triangle t D u^n + \triangle t(\frac{1}{\beta^-} - \frac{1}{\beta^+})f^n + \triangle t\frac{1}{\beta^+}f^{n+1}$ and $F^- = \frac{1}{\beta^-}\triangle t f^{n+1}$. Here we want $|\frac{\beta^- - \beta^+}{\beta^+}| \approx 1$. It is expected that first order truncation error of $O(\triangle t)$ will be preserved. The second order corrected central difference can be adopted to approximate the Laplacian operator when irregular domain is encountered. With such treatment, it can be easily seen that the uniform diagonal coefficient takes place in discretized scheme (6.19). FFT-based AMIB method can be formulated in order to solve above parabolic interface problem with FFT efficiency in solving the desired numerical solution in each step. Numerical experiments have been carried out to test the performance of the proposed algorithm. Extensive experiments have been conducted, showing that such algorithm is implicit and stable enough even in case of high coefficient ratio contrast. Unfortunately, even though second order spatial convergence is observed in the numerical experiments. the temporal convergence of the scheme is only zeroth order as shown in the following example.

We consider a 2D Poisson's problem on a three-leaf shaped interface.

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

with an interface Γ

$$r = 0.5 + 0.1\sin(3\theta),$$

and two exact solutions of Poisson equation are designated by

$$u(x, y) = \begin{cases} e^{-t}e^{x}(\sin(y) + y^2) & \text{inside } \Gamma, \\ e^{-t}\sin(3x)\sin(3y) & \text{otherwise,} \end{cases}$$

on a square domain $[-\frac{\pi}{2}, \frac{\pi}{2}] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$. The coefficients are set as

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

The corresponding source terms q can be determined by the analytical solution and the governing equation.

Table 6.1: Spatial and temporal convergence.

$[n_x, n_y]$	Spatial convergence				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	3.72E-3	–	1.32E-2	–	9.7E-2
[64, 64]	8.13E-4	2.19	2.39E-3	2.47	0.36
[128, 128]	2.51E-4	1.70	9.47E-4	1.34	1.69
[256, 256]	6.84E-5	1.88	2.72E-4	1.80	10.3
[512, 512]	1.92E-5	1.83	7.34E-5	1.89	54.2

Table 6.2: Temporal convergence.

n_t	Temporal convergence				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
5	6.03E-4	–	4.54E-3	–	37.3
10	5.85E-4	0.04	4.40E-3	0.05	67.7
20	5.52E-4	0.08	4.15E-3	0.08	133
40	4.92E-4	0.17	3.70E-3	0.17	244

The above proposed algorithm fails to yield expected first order temporal accuracy. Other high

order explicit temporal scheme like Runge-Kutta method can be used in place of explicit Euler method (6.14) to provide more accurate future solution in (6.17). Or some simple iterative approach can be used to refine solution u^{n+1} obtained from Euler approximation, which can still be seen as explicit approach. However, such treatments will lead the aforementioned algorithm to be no longer stable. Thus new robust and accurate algorithm is expected. In following, a temporally and spatially second order accurate scheme is proposed with the efficiency accelerated by multigrid algorithm.

6.1.1 Temporal discretization

For temporal discretization, the Crank-Nicolson scheme is adopted to form the following time stepping scheme:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\beta_{i,j}\triangle t} = \frac{1}{2}\Delta u_{i,j}^{n} + \frac{1}{2}\Delta u_{i,j}^{n+1} + \frac{1}{2\beta_{i,j}}(f_{i,j}^{n} + f_{i,j}^{n+1}), \tag{6.22}$$

which has the equivalent form below after reorganizing the terms in (6.22)

$$\frac{2}{\beta_{i,j}\triangle t}u_{i,j}^{n+1} - \Delta u_{i,j}^{n+1} = \frac{2}{\beta_{i,j}\triangle t}u_{i,j}^{n} + \Delta u_{i,j}^{n} + \frac{1}{\beta_{i,j}}(f_{i,j}^{n} + f_{i,j}^{n+1}). \tag{6.23}$$

The above Crank-Nicolson scheme admits a second order temporal truncation error. Note that the Laplacian terms at time step $t = t^{n+1}$ and $t = t^{n}$ need spatial discretization such that the iteration process (6.23) can proceed for the discrete solutions at certain time instant.

6.1.2 Spatial discretization

The discussion on spatial discretization for elliptic interface problem can apply to the Laplacian approximation here. The second order corrected differences, the approximation to Cartesian derivative jumps using polynomials built on fictitious values can still contribute to the Laplacian operator approximation. The difference is that the interface jump conditions are time dependent. Hence fictitious values need to be generated in each time step, and the approximations to Cartesian derivative jumps also require corresponding treatment each time.

6.1.3 Formulation of augmented system

Great details have been given in the section on elliptic interface problem to demonstrate how corrected differences are used to approximate the Laplacian operator. The right side of the Crank-Nicolson scheme (6.23) can be easily determined once the information of solution u^n, the interface conditions ϕ, ρ, ψ at current step t^n, and the source term f at t^n and t^{n+1} are gathered. Especially, corrected differences with the the jump approximation method (2.9) are utilized to deal with approximation at irregular points on both sides of the time-stepping scheme (6.23). However, it is not an efficient approach to directly apply iterative solvers to obtain the discrete solution at t^{n+1} from the discretization of the Laplacian operator on the left side of scheme (6.23). It is noticeable that the standard second order central difference matrix structure with some additional diagonal entries is obtained if the corrected difference is neglected. Due to such special matrix structure, a multigrid-based fast solver is established to fulfill the fast algorithm for solving parabolic interface problem. Prior to the introduction of fast multigrid solver, we need to introduce the auxiliary variables in order to take advantage of the special matrix structure as part of augmented system.

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 for spatial discretization, the scheme (6.23) is expanded at all interior nodes as

$$D_h U_{i,j}^{n+1} - L_h U_{i,j}^{n+1} - C_{i,j}^{n+1} = D_h U_{i,j}^{n} + L_h U_{i,j}^{n} + C_{i,j}^{n} + \frac{1}{\beta_{i,j}}(f_{i,j}^{n+1} + f_{i,j}^{n}),$$
$$\text{for } 1 \le i \le n_x - 1, \;\; 1 \le j \le n_y - 1, \tag{6.24}$$

where $C_{i,j}$ is the correction term, and $L_h U_{i,j}$ is the standard five points central difference scheme

$$L_h U_{i,j} := \frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2}, \tag{6.25}$$

$$D_h U_{i,j} := \frac{2}{\beta_{i,j} \triangle t} U_{i,j}. \tag{6.26}$$

Note that the subscript (i, j) indicates the gird node (x_i, y_j), and the superscript n and $n + 1$ imply the time t^n and t^{n+1}, respectively. The degree of freedom of this system is $N = (n_x + 1)(n_y + 1)$. The correction term $C_{i,j}$ only exists at irregular points but vanishes at regular points. The discretizations (6.24) gives rise to the matrix-vector form

$$DU^{n+1} + \hat{A}U^{n+1} + BQ^{n+1} = DU^n - \hat{A}U^n - BQ^n + \hat{F}^n \tag{6.27}$$

where D is a matrix with non-zero entries only along the diagonal, $\hat{A}$ is a N-by-N non-symmetric and diagonally dominant matrix, the unknown U is for solution at all grid nodes and Q is for all the needed auxiliary variables. The diagonal entries of matrix D are non-constant due to the different values of β in different subdomain. As the right hand side of (6.27) can be determined in each time step, we can denote the right hand of above system as F. The above system can be further simplified as

$$AU + BQ = F, \tag{6.28}$$

after we combine the coefficient matrix D and $\hat{A}$ into A, i.e., $A = D + \hat{A}$, and ignore the superscript $n + 1$. Both U and Q are to be solved as time t^{n+1}. Supposed Q is already determined, then the time stepping process continues by updating right side U^n with obtained solution U^{n+1} in (6.27). The Dirichlet boundary condition is imposed in the right side vector F with some rows of matrix A composed by vector $(0, 0, \cdots, 0, 1, 0, \cdots, 0)$, i.e., only the diagonal element is one while other entries are zero. The total number of auxiliary variables Q, denoted as M here, is double 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.

We need to determine Q first to solve for U in (6.28). It is necessary to combine (6.28) and the Cartesian derivative jump approximation

$$CU + IQ = R$$

to form augmented equation system

$$KW = R, \tag{6.29}$$

where

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

Note that the dimensions of matrix C and I match those of A and B. C is a M-by-N sparse matrix, and I is a M-by-M identity matrix.

The matrix structure of A allows for fast inversion with aid of multigrid method, which will be discussed in latter section. Bearing this in mind, we will formulate two efficient solvers for the parabolic interface problem by using the following Schur complement strategy.

Schur complement

In each time step, we aim at solving U from

$$AU = F - BQ, \tag{6.30}$$

after we determine Q, and move all the known quantities to right hand side. Then the solution U could be quickly obtained by applying the multigrid solver on the above equation thanks to the

matrix structure of A. Such step is preceded by the determination of Q from the Schur complement

$$(I - CA^{-1}B)Q = \Phi - CA^{-1}F, \tag{6.31}$$

which is obtained by eliminating U from the augmented system (6.29).
Note that the linear system (6.31) for Q has a much smaller degree of freedom than U, i.e. M. We propose two approaches to solve Q from Eq. (6.31) in terms of iterative solver or the LU decomposition on the coefficient matrix of the left hand side in Eq.(6.31). The following steps are used to illustrate the procedure. One is by iterative algorithm of GMRES method(GMRES approach), while the other is by LU decomposition(LU approach).

- *GMRES approach.*

    1. As a linear system, the right hand side RHS=$\Phi - CA^{-1}F$ is determined by applying multigrid solver on $A^{-1}F$ and some arithmetic operations.
    2. For the left hand side(LHS), we may consider an iteration approach by GMRES method. In this approach, the matrix vector product of $(I - CA^{-1}B)Q$ is fulfilled in a couple of steps. Due to its equivalence to $IQ - CA^{-1}BQ$, a multigird solver is carried out on the product of BQ, i.e., $A^{-1}(BQ)$. Then it is followed by some more arithmetic operations on $IQ - CA^{-1}(BQ)$.
    3. For the first time step $t^0 = 0$, an initial guess $Q = (0, 0, \cdots, 0)^T$ is used to start the GMRES iteration. For the rest time steps, the initial guess of Q is taken as the computed Q from the previous time step. Either the maximal iteration number 5000 or error tolerance equal to 10^{-10} terminates the GMRES iteration. The termination criteria are flexible according to actual computational needs.

- *LU approach.*

    1. The first step is the same as that in the above GMRES approach.

2. For the left hand side(LHS), we will explicitly construct the matrix $I - CA^{-1}B$, where the inverse of A, i.e. A^{-1}, should not be formed, but is applied on the columns of matrix B. Then the explicit matrix is decomposed into the matrix product of X and Y from a LU decomposition solver. Such matrix construction and decomposition need to conducted only once before time stepping.

3. The solution Q in each time step can be simply obtained via the LU forward and backward substitutions.

Remark 6.1.1. *The efficiency of GMRES approach depends on the iteration number for the GMRES iteration process. Note that each iteration involves one multigrid inversion with a complexity $O(N) = C_2N$ for some constant C_2. Fortunately, the iteration is for solving* (6.31)*, whose dimension $M \ll N$. Generally, for any iteration solver, the better conditioned the iteration matrix is, the smaller the iteration number is. The selection of MIB-PULS or MIB-MINUS mentioned in Remark 2.1.4 provides a good conditioning strategy. This makes the iteration number increase moderately as the mesh is refined.*

Remark 6.1.2. *The strategy of LU approach is different from GMRES approach in the sense that the Schur complement matrix is formed with multigrid algorithm applied for each column of matrix B. The more grid lines intersect the interface, the more columns matrix B has. The computing cost is mainly contributed by the multigrid inversion on all the columns, before time stepping. In the time-stepping process, only the forward and backward substitution of LU are applied to solve for auxiliary variable, and one multigrid inversion is deployed on* (6.30) *for solution of u. Moreover, the LU approach is not impacted by the conditioner number of the linear system as it does for GMRES approach, since the LU decomposition is a direct rather than an iterative solver.*

Remark 6.1.3. *The complexities of the two approaches are in the same magnitude. As it is second order accurate for both temporal and spatial discretization, it is reasonable to set time increment Δt proportional to spatial partition size h to ensure the overall second order convergence.*

Suppose the total degree of freedom $N = n^2$, where n is the grid number in each direction. The intersection number M is hence equal to $C_0 n$ for some constant C_0, and the temporal evolution number $N_t = C_3 n$ for another constant C_3. We may roughly have a estimate of the complexities of the two approaches. The complexity of LU approach is

$$\begin{aligned}
&O(N) \times M + O(M^2) \times N_t + \frac{2}{3}M^2 \\
=&C_2 NM + C_1 M^2 N_t + \frac{2}{3}M^2 \\
=&C_1 C_0^2 n^2 C_3 n + C_2 n^2 C_0 n + \frac{2}{3}C_0^3 n^2 \\
=&C_1 C_0^2 C_3 n^3 + C_2 C_0 n^3 + \frac{2}{3}C_0^3 n^2 \\
=&C_0(C_0 C_1 C_3 + C_2 + \frac{2}{3}C_0^2)n^3,
\end{aligned}$$

where the first and second term in the first equality indicates the total sum of matrix generation cost arising from multigrid solver and the cost of multigrid for inversion deployed on (6.30) *and right side of* (6.31)*, and the LU forward and backward substitution cost in all time evolution steps. Besides, the last term in the first equality denotes the LU decomposition for the generated matrix in the first time stepping.*

The computational complexity of GMRES approach is

$$\begin{aligned}
&O(N) \times C_4 \times N_t \\
=&C_2 n^2 C_4 C_3 n \\
=&C_2 C_3 C_4 n^3 \\
=&C_3(C_2 C_4)n^3,
\end{aligned}$$

where C_4 indicates the sum of total times multigrid solver is call in the GMRES subroutine, and multigrid for inversion deployed on (6.30) *and right side of* (6.31)*.*

The computational complexities of the two approach are all of $O(n^3)$. Note that the number C_4 is a constant under the assumption that the iteration number of GMRES solver does not depend on

mesh size. But in the practical implementation of our algorithm, the iteration number depends on the mesh size weakly. This can be seen in the numerical examples of the latter section. This makes AMIB-LU slightly more efficient than AMIB-GMRES when achieving second order accurate solution.

Remark 6.1.4. *When an explicit finite difference method is employed to solve parabolic PDEs, a stability condition with* $\Delta t < Ch^2/\beta_{max}$ *has to be satisfied, where* $\beta_{max} = max_{x\in\Omega}\beta(x)$*. This means that one has to take* $N_t = O(n^2)$ *to ensure stability. It is reasonable to assume the complexity in each time step of this explicit scheme to be on the order of* $O(n^2)$*, since there are* n^2 *degree of freedom. Thus, the overall complexity of explicit finite difference solution of the 2D parabolic interface problems is about* $O(n^4)$*. This means that the proposed AMIB methods could be asymptotically faster than the explicit schemes.*

Multigrid algorithm

In the preceding section, a multigrid approach has been assumed in the discussion of building efficient algorithm. In this section, details on this solver are provided.
The multigrid approach is an efficient algebraic algorithm for solving a linear system

$$AU = B \tag{6.32}$$

with known matrix A and vector B. Recall in our study, A has a special matrix structure: $A = D + \hat{A}$. Thus, Eq. (6.32) can be regarded as obtaining from a discretization to negative of Laplacian operator plus solution with variable coefficient as below:

$$-\Delta u + a(x, y)u = b(x, y), \tag{6.33}$$

where the domain and its partition are the same as that for the problem in this work, the discrete values of $a(x, y)$ on the given partition are the same as diagonal values of D, U is the discrete solution to u in (6.33), and function b has its discrete data equal to B at the interior grid nodes and

for boundary conditions.

To be more clear, at the interior node,

$$a(x_i, y_j) = D((j-1)\cdot nx + i, (j-1)\cdot nx + i),$$
$$b(x_i, y_j) = B((j-1)\cdot nx + i, (j-1)\cdot nx + i),$$

for $1 \le i \le n_x - 1$, and $1 \le j \le n_y - 1$, while at the grid node on boundary of Ω,

$$b(x_i, y_j) = B((j-1)\cdot nx + i, (j-1)\cdot nx + i), \tag{6.34}$$

for (i, j) such that $(x_i, y_j) \cap \partial\Omega \neq \emptyset$.

The discrete problem can then be reformulated as below according to the discrete data,

$$-L_h U_h + D_h U_h = b_h \text{ in } \Omega_h, \tag{6.35}$$
$$B_h U_h = b_h, \qquad \text{on } \partial\Omega_h, \tag{6.36}$$

where L_h and D_h are the discrete versions of Laplacian operator, and the coefficient function $a(x, y)$ as defined in last section, and B_h is the boundary operator. The L_h and D_h in (6.35) share the same stencil structure as in (6.25) and (6.26). When the mesh is refined by doubling h to H, the value of b_H will be sampled directly on the refined interior node, but the boundary value of b_H is collected in the refined grid nodes on the boundary in the same way. The corresponding coefficient matrix for the discrete governing equation (6.35) on coarse mesh H is a submatrix of that for mesh h.

This makes it possible for applying multigrid method to solve the linear system (6.32) efficiently. Once U_h is determined, the solution of (6.32) is set as $U = U_h$.

The main idea of multigrid method lies in reducing the high frequency components of solution error on fine mesh, and that the low frequency errors are solved on coarse grids. Due to the fact that low frequency components of the errors can not be effectively reduced, the residual is

restricted to coarser grids such the low frequency errors are solved therein. Such determined errors are then interpolated onto fine meshes to provide some correction to the already obtained solution on fine meshes. Over the process, relaxation is needed on fine mesh to improve the solution accuracy, while restriction operator is imposed from mesh to coarse mesh and interpolation is need from coarse mesh to fine mesh. Such process is iterated until the solution meets certain accuracy.

The relaxation is used to reduce errors in two aspects. One is for smoothing the high frequency component on the fine mesh. The other is used after the obtained errors are interpolated to add correction on solution on fine mesh. In this case, the relaxation is to reduce the error introduced from interpolation. In this work, we adopt the red-black Gauss-Seidal smoother as the relaxation scheme. The two-grid correction scheme as the simplest multigrid adopts only two level of meshes to achieve the solution correction via iteration between restriction and interpolation. We use such two-grid correction scheme to illustrate the way we apply multigrid scheme to obtain the solution to the linear system (6.32).

1. The initial guess for U_h is set as 0.

2. Relax Eq.(6.35) v_1 times on the fine grid.

3. Compute the residual on fine grid, $r_n = b_h + L_h U_h - D_h U_h$.

4. Restrict the residual to coarse grid $r_H = R_h^H r_h$ with the restriction operator R_h^H, where $H = 2h$.

5. Exactly solve for the solution e_H of the residual problem on coarse grid

$$-L_H e_H + D_H e_H = r_H. \tag{6.37}$$

6. Interpolate the correction to fine grid $e_h = I_H^h e_H$.

7. Apply the correction to get a better approximation $U_h := U_h + e_h$.

8. Relax the Eq.(6.35) v_2 times on the fine grid.

The above process continues until convergence. In our multigrid approach, we generalize the above process to more grid levels to avoid the exact correction solution in the step 5 in case that the cost is still very high on that coarse grid. This promotes the necessity of even coarser grids for correction solutions. The above procedure demonstrates the two-grid correction scheme using v-cycle. In our algorithm, we deploy v-cycle for the multigrid solver rather than the full multigrid approach since $v-$cycle tends to be more efficient in our practical implementation.
Restriction and interpolation are two needed transfer operators in the correction procedure. In our implement, it is sufficient to adopt the following standard full-weighting stencil for restriction,

$$R_h^H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^H, \tag{6.38}$$

and the following standard interpolation scheme

$$I_H^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_H^h. \tag{6.39}$$

For the practical implementation of our multigrid algorithm for problem (6.35)(6.36), we set both v_1 and v_2 equal to 2. For the exact solution on the coarsest grid, we use biconjugate gradient iterative solver [62]. The iteration stopping criteria is set as error tolerance equal to 10^{-10}, and maximal iteration number equal to 5000. Besides, the multigrid stopping criteria is the residual L_2 error less than 10^{-10}. We may use number of levels grid flexibly for the multigrid algorithm based on the mesh refinement.

6.2 Numerical experiments

In this section, numerical experiments are carried out to demonstrate the accuracy, efficiency and stability of the proposed two AMIB algorithms for solving parabolic interface problems. For sake of notation, we name the AMIB method coupled with the GMRES approach on Schur complement as AMIB-GMRES, while the AMIB method associated with LU decomposition is called as AMIB-LU. For comparison purpose, the standard MIB method [85] with Crank-Nicolson scheme for time-stepping is also implemented.

For simplicity, a square domain with uniform number of partitions in each direction is assumed. We consider the number as $N = n_x = x_y$ in each direction. The numerical accuracy and convergence will be tested by comparing the numerical solutions with exact solutions under the L_∞ and L_2 norm.

The iteration solver used in MIB method is biconjugate gradient iterative solver [62]. The maximal iteration number is 5000, and the tolerance error is set as 10^{-10}. The number of mesh levels in the multigrid algorithm for AMIB is chosen as 4 for $n_x = 16$, as 5 for $n_x = 32$, as 6 for $n_x = 64$, and as 7 for all $n_x > 64$. For multigrid algorithm application, we have to set n_x to be 2^k for a positive integer k.

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

6.2.1 Numerical accuracy

Example 1. In this example, we consider a parabolic problem on a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with a circular interface defined by $r^2 := x^2 + y^2 = \frac{1}{4}$. The exact solution to this problem is

$$u(x, y) = \begin{cases} \sin(t) + e^{-x-y}, & r \leq 0.5, \\ \sin(t) + e^{x+y}, & \text{otherwise}, \end{cases} \tag{6.40}$$

with the diffusion coefficients

$$\beta = \begin{cases} 1, & r \le 0.5, \\ 10, & \text{otherwise.} \end{cases}$$

Here we call β as β^+ when it is outside of Γ, and β^- when it is inside of the interface Γ. The interface jump conditions on an interface point $(x, y) = (\frac{1}{2}\cos(\theta), \frac{1}{2}\sin(\theta))$ can be derived as below:

$$\begin{aligned}
[u] =& e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))} - e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))} \\
[\beta u_n] =& \beta^+(\cos(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))} + \sin(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))}) \\
&+ \beta^-(\cos(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))} + \sin(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))}) \\
[u_\tau] =& \beta^+(\sin(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))} - \cos(\theta)e^{\frac{1}{2}(\cos(\theta)+\sin(\theta))}) \\
&+ \beta^-(\sin(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))} - \cos(\theta)e^{-\frac{1}{2}(\cos(\theta)+\sin(\theta))})
\end{aligned}$$

It follows the same principle in below notations. The source term $f(x, y)$ is related to the above designated solution,

$$f(x, y) = \begin{cases} (\cos(t) - 2\beta^- \sin(t))e^{-x-y}, & r \le 0.5, \\ (\cos(t) - 2\beta^+ \sin(t))e^{x+y}, & \text{otherwise.} \end{cases}$$

The initial condition and boundary conditions can be easily determined by the analytical solutions. Beside, we may observe that both the zeroth and first order jump condition on the interface are time-independent. To test spatial convergence rate, we consider the fixed stopping time at $T = 1$ with the time increment $\Delta t = 10^{-3}$ being small enough such that temporal error will not interfere with the examination of the spatial convergence. Table 6.3 clearly reveals that the spatial disretization reaches the second order convergence in space.

On the other hand, the temporal convergence needs to be tested. For such concern, the spatial partition in each direction is set to be $N = 512$, and various time steps are deployed. The results in Table 6.4 indicates that the second order temporal convergence can be monitored from the second

row to the last second row. Such second order convergence starts to occur from $\Delta t = 0.25$. The total CPU time of each time evolution methods is listed in the last column in Table 6.4, where we can compare the efficiency between these method. It can be observed that if the number of time steps is small, AMIB-GMRES is superior than AMIB-LU. However, in the long run, AMIB-LU outperforms both AMIB-GMRES and MIB methods. Take the case of $nt = 128$ for instance, CPU time of MIB is about 12 times larger than that of AMIB-LU. Similar results can also be found in the last row of Table 6.3 of spatial convergence test when partition number is 512 in each direction. The number of time evolution steps is 1000, leads to CPU cost of MIB being 14 times larger than that of AMIB-LU. The inefficiency of MIB or other interface algorithms implemented with iterative solvers for parabolic interface problems leads to the demand for the development of efficient solvers.

Table 6.3: Spatial convergence for example 1.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	2.08E-4	–	8.82E-4	–	0.94
[64, 64]	3.21E-5	2.70	1.69E-4	2.38	5.48
[128, 128]	7.52E-6	2.09	3.13E-5	2.43	26.6
[256, 256]	1.94E-6	1.95	9.07E-6	1.79	175
[512, 512]	5.66E-7	1.95	2.97E-6	1.79	1143
	AMIB-LU				
	L_2		L_∞		
	Error	Order	Error	Order	
[32, 32]	2.08E-4	–	8.82E-4	–	0.30
[64, 64]	3.19E-5	2.70	1.68E-4	2.38	1.49
[128, 128]	7.52E-6	2.09	3.13E-5	2.43	6.50
[256, 256]	1.94E-6	1.95	9.08E-6	1.79	29.8
[512, 512]	5.71E-7	1.95	2.99E-6	1.79	200
	MIB				
	L_2		L_∞		
	Error	Order	Error	Order	
[32, 32]	2.07E-4	–	8.75E-4	–	0.62
[64, 64]	3.19E-5	2.70	1.68E-4	2.38	4.76
[128, 128]	7.50E-6	2.08	3.13E-5	2.42	35.4
[256, 256]	1.94E-6	1.94	9.09E-6	1.79	336
[512, 512]	5.66E-7	1.95	2.97E-6	1.79	2935

Example 2. In this example, we consider a parabolic problem on a square domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$ with a circular interface defined by $r^2 := x^2 + y^2 = \frac{1}{4}$. The exact solution to this problem is

$$u(x, y) = \begin{cases} \sin(kx)\cos(ky)\cos(t), & r \leq 0.5, \\ \cos(kx)\sin(ky)\cos(t), & \text{otherwise}, \end{cases} \tag{6.41}$$

Table 6.4: Temporal convergence for example 1.

nt	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	1.32E-4	–	5.46E-4	–	2.76
4	8.98E-5	0.56	3.25E-4	0.75	5.28
8	2.48E-5	1.86	7.42E-5	2.13	9.73
16	6.21E-6	2.00	1.87E-5	1.99	19.36
32	1.56E-6	1.99	4.88E-6	2.02	37.88
64	6.26E-7	1.32	3.21E-6	0.60	76.9
128	5.60E-7	0.16	2.99E-6	0.10	148.4
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	1.32E-4	–	5.46E-4	–	58
4	8.98E-5	0.56	3.25E-4	0.75	59.4
8	2.48E-5	1.86	7.42E-5	2.13	61.5
16	6.21E-6	2.00	1.87E-5	1.99	63.2
32	1.56E-6	1.99	4.88E-6	1.94	64.7
64	6.34E-7	1.30	3.24E-6	0.59	66.8
128	5.76E-7	0.13	3.04E-6	0.09	74
	MIB				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	1.32E-4	–	5.46E-4	–	21.97
4	8.98E-5	0.56	3.25E-4	0.75	42.34
8	2.48E-5	1.86	7.42E-5	2.14	81.72
16	6.21E-6	2.00	1.87E-5	1.98	153.34
32	1.56E-6	1.99	4.88E-6	1.94	287.03
64	6.27E-7	1.31	3.22E-6	0.60	533
128	5.69E-7	0.14	3.02E-6	0.09	863

The source terms can be correspondingly determined,

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

The zeroth and first order interface can be found to be time and space dependent, representing the most general jump conditions. In this example, we consider $k = 2$.

It is also reasonable to approximate the equation (6.12) with

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\beta_{i,j}\triangle t} = \frac{1}{2}\Delta u_{i,j}^{n} + \frac{1}{2}\Delta u_{i,j}^{n+1} + \frac{1}{\beta_{i,j}} f_{i,j}^{n+\frac{1}{2}}, \tag{6.42}$$

with the difference to Eq.(6.22) on the approximation to f at time $t = t_n + 0.5\Delta t$ by $f_{i,j}^{n+\frac{1}{2}}$ in Eq.(6.42), while $\frac{1}{2}(f_{i,j}^{n} + f_{i,j}^{n+1})$ is used in Eq.(6.23). We are interested in the numerical impacts with such different treatment for the source term f. Table 6.5 shows their numerical comparisons

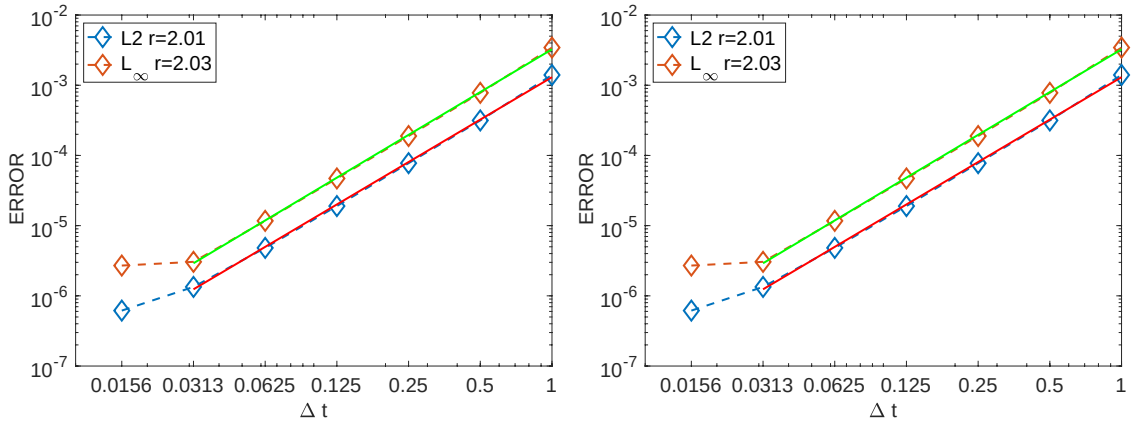

Figure 6.1: The temporal convergence for example 2. The left is for AMIB-GMRES, and the right is for AMIB-LU.

between scheme (6.42) and (6.22) with $f^{n+\frac{1}{2}}$ indicating scheme (6.42) and the other part for scheme (6.22). In this experiment, the partition number in each direction is $N = 512$. The temporal convergence can all be determined according to the results. It can be found that the solution accuracy from (6.22) with $nt = 128$ is comparable to that of (6.42) with $nt = 256$. Such accuracy improvement justifies that scheme (6.22) is superior to (6.42) in terms of accuracy when dealing with the interface jump conditions that are time and space independent. On the other hand, it can be found in Tables 6.6 and 6.7 that the time costs from scheme (6.22) and (6.42) are basically the same for each nt. From this point of view, scheme (6.22) is a better choice for time evolution than (6.42). Thus we will focus on scheme (6.22) in the following discussion to deal with the parabolic interface problem under all kinds of circumstances.

The error versus time partition increment Δt on log-log scale from scheme (6.22) is plotted on Fig.6.1, from which we can observe that the temporal second order accuracy is obtained for both AMIB-GMRES and AMIB-LU methods.

On the other hand, the spatial convergence are investigated on the two methods. The stopping time is fixed as $T = 2$, and the time increment is set as 0.002. The second order spatial convergence of the two approaches concerned with source term f can be observed from the results in Table 6.6. The comparison shows no much difference on spatial accuracy by scheme (6.22) and (6.42).

We then want to study the impact of ratio of $\beta^+ : \beta^-$ on the stability for scheme (6.22). We will fix

Table 6.5: Temporal convergence for example 2.

nt	AMIB-GMRES-$f^{n+\frac{1}{2}}$					AMIB-GMRES				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
2	3.22E-2	–	7.55E-2	–	3.04	1.39E-3	–	3.43E-3	–	2.81
4	5.56E-3	2.53	1.57E-2	2.27	5.9	3.14E-4	2.15	7.77E-4	2.14	5.69
8	9.47E-4	2.55	2.94E-3	2.42	12.5	7.59E-5	2.05	1.88E-4	2.05	11.2
16	1.67E-4	2.50	4.35E-4	2.76	23	1.89E-5	2.01	4.68E-5	2.01	22
32	3.82E-5	2.13	8.32E-5	2.39	45	5.00E-6	1.92	1.21E-5	1.95	45.4
64	9.68E-6	1.99	2.13E-5	1.98	87	1.44E-6	1.80	3.30E-6	1.87	96.3
128	2.579E-6	1.91	6.05E-6	1.82	168	6.60E-7	1.13	2.84E-6	0.22	170
256	9.03E-7	1.53	3.25E-6	0.91	334					
	AMIB-LU-$f^{n+\frac{1}{2}}$					AMIB-LU				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
2	3.22E-2	–	7.55E-2	–	65	1.40E-3	–	3.44E-3	–	57
4	5.56E-3	2.53	1.57E-2	2.27	69	3.15E-4	2.15	7.79E-4	2.15	59.6
8	9.47E-4	2.55	2.94E-3	2.42	62	7.64E-5	2.04	1.89E-4	2.04	58.9
16	1.67E-4	2.50	4.35E-4	2.76	62	1.90E-5	2.01	4.70E-5	2.01	57.7
32	3.82E-5	2.13	8.32E-5	2.39	60.6	4.82E-6	1.98	1.17E-5	2.01	60.7
64	9.65E-6	1.99	2.11E-5	1.98	66	1.34E-6	1.85	3.05E-6	1.94	64.6
128	2.57E-6	1.91	5.98E-6	1.82	75.1	6.16E-7	1.12	2.70E-6	0.18	75.9
256	8.88E-7	1.53	3.18E-6	0.91	91					
512	5.69E-7	0.64	2.76E-6	0.20	129					

Table 6.6: Spatial convergence for example 2.

$[n_x, n_y]$	AMIB-GMRES-$f^{n+\frac{1}{2}}$					AMIB-GMRES				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
$[32, 32]$	1.15E-4	–	5.16E-4	–	1.11	2.25E-4	–	5.16E-4	–	1.19
$[64, 64]$	1.10E-4	1.03	3.42E-4	0.59	7.39	1.10E-4	1.03	3.42E-4	0.59	7.86
$[128, 128]$	1.44E-5	2.93	6.40E-5	2.42	42	1.44E-5	2.93	6.40E-5	2.42	43
$[256, 256]$	2.06E-6	2.82	1.54E-5	2.61	264	2.04E-6	2.82	1.04E-5	2.61	258
$[512, 512]$	5.18E-7	2.01	2.65E-6	1.99	1457	5.05E-7	2.01	2.62E-6	1.99	1463
	AMIB-LU-$f^{n+\frac{1}{2}}$					AMIB-LU				
	L_2		L_∞		CPU	L_2		L_∞		CPU
	Error	Order	Error	Order		Error	Order	Error	Order	
$[32, 32]$	1.15E-4	–	5.16E-4	–	0.34	2.25E-4	–	5.16E-4	–	0.7
$[64, 64]$	1.10E-4	0.06	3.42E-4	0.59	1.66	1.10E-4	1.03	3.42E-4	0.59	1.78
$[128, 128]$	1.44E-5	2.93	6.40E-5	2.42	8.04	1.44E-5	2.93	6.40E-5	2.43	8.76
$[256, 256]$	2.06E-6	2.81	1.05E-5	2.61	39.4	2.05E-6	2.91	1.04E-5	1.79	41.6
$[512, 512]$	5.28E-7	1.96	2.68E-6	1.97	231	5.15E-7	1.99	2.65E-6	1.99	227

one of two β values equal to 1, then vary the value of the other one to make the ratio large or small enough. We fix the partition number in each direction as 128, and the stopping time equal to 2 with time increment 0.002. It can be observed from the Table 6.7 that the two AMIB methods are all stable enough to deal with large contrast of the coefficients. On the other hand, the varying coefficient contrasts seem to have larger impact on the CPU time of AMIB-GMRES than AMIB-LU. The CPU time of AMIB-GMRES become twice expansive when the ratio of $\beta^+ : \beta^-$ is raised from $10 : 1$ to $10000 : 1$. This is also the same case for AMIB-GMRES when ratio of $\beta^+ : \beta^-$ is raised from $1 : 10$ to $1 : 10000$. The milder impact on AMIB-LU is due to the direct

solver by LU decomposition rather than iterative solvers whose condition number can be affected by the ratio of $\beta^+ : \beta^-$. Such conclusion is based on the comparison between the two AMIB methods vertically in Table 6.7.

Note that the"Small β side" indicates the the MIB extrapolation part is fulfilled by the function values from the domain related to the small β. The "Large β" side indicates the domain related to the large β in a similar fashion. From the horizontal comparison for each AMIB method in Table 6.7, MIB fictitious values with the "Small β side" extrapolation is superior to the one with the "Large β side", because the CPU time is less than that in the other way around for AMIB-GMRES. This justifies the selection of β in remark 2.1.4 for concern of being well-conditioned. Hence, it is preferable to adopt the fictitious values with the extrapolation part fulfilled using the function values from domain of "small β." The accuracy of the approximate solutions concerned with the "Small" or "Large" β has no much big difference using AMIB-GMRES or AMIB-LU in this example. In the following examples, we will adopt the "Small β side" strategy for both AMIB schemes.

Table 6.7: Spatial convergence for example 2 with respect to different coefficient contrasts. $N = 128$, $\Delta t = 0.002$.

$[\beta^+,\beta^-]$	AMIB-GMRES					
	Small β side			Large β side		
	L_2	L_∞	CPU time(s)	L_2	L_∞	CPU time(s)
[10000, 1]	1.63E-5	5.11E-5	71.4	1.86E-5	6.70E-5	92.6
[1000, 1]	1.63E-5	5.10E-5	65.4	1.86E-5	6.69E-5	92.8
[100, 1]	1.61E-5	5.04E-5	52.4	1.84E-5	6.60E-5	69.3
[10, 1]	1.47E-5	4.43E-5	31.4	1.65E-5	5.74E-5	33.8
[1, 10]	9.48E-6	4.71E-5	36.2	7.01E-6	3.44E-5	36.2
[1, 100]	6.03E-5	1.56E-4	58.3	6.20E-5	1.45E-4	66.4
[1, 1000]	4.66E-4	9.00E-4	67.8	6.38E-4	1.19E-3	81.6
[1, 10000]	5.26E-4	8.52E-4	71.2	5.62E-3	1.02E-2	89.5
$[\beta^+,\beta^-]$	AMIB-LU					
	Small β side			Large β side		
	L_2	L_∞	CPU time(s)	L_2	L_∞	CPU time(s)
[10000, 1]	1.63E-5	5.11E-5	10.9	1.86E-5	6.72E-5	10.9
[1000, 1]	1.63E-5	5.10E-5	10.58	1.86E-5	6.71E-5	10.46
[100, 1]	1.61E-5	5.04E-5	8.8	1.84E-5	6.61E-5	9.1
[10, 1]	1.47E-5	4.43E-5	6.8	1.66E-5	5.75E-5	6.9
[1, 10]	9.48E-6	4.71E-5	7.0	7.03E-6	3.43E-5	6.9
[1, 100]	6.03E-5	1.56E-4	9.2	6.20E-5	1.45E-4	8.5
[1, 1000]	4.66E-4	9.00E-4	8.6	6.38E-4	1.19E-3	8.6
[1, 10000]	5.29E-4	8.56E-4	9.0	5.63E-3	1.02E-2	8.7

Example 3. We are interested in the performance of the two AMIB methods on complicated interface shape. In this example, we focus on the case of interface shape governed by parametric

function

$$r = 0.5 + 0.1\sin(3\theta), \tag{6.43}$$

embedded in the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The solution and all the related boundary and interface conditions are set as in the last example. The stopping time is given as $T = 1$, and the coefficients are $(\beta^+, \beta^-) = (100, 1)$. The spatial accuracy and convergence are examined with the small time increment $\Delta t = 0.001$. The numerical results in Table 6.8 on the successively refined mesh demonstrate the second order spatial accuracy.

Table 6.8: Spatial convergence for example 3.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	6.70E-4	–	2.17E-3	–	1.67
[64, 64]	1.52E-4	2.14	5.05E-4	2.10	9.82
[128, 128]	3.63E-5	2.07	1.39E-4	1.86	53
[256, 256]	8.62E-6	2.07	3.52E-5	1.98	399
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	6.70E-4	–	2.17E-3	–	0.42
[64, 64]	1.52E-4	2.14	5.05E-4	2.10	1.94
[128, 128]	3.63E-5	2.07	1.39E-4	1.86	8.54
[256, 256]	8.62E-6	2.07	3.52E-5	1.98	45.5

We next examine the temporal order of the two AMIB methods by setting $n_x = 512$, and the stopping time as $T = 1$. As the time evolution scheme (6.22) is efficient enough for the temporal convergence, the numerical results in Table 6.9 show that both AMIB methods obtain their limiting convergence very quickly, leading to the temporal second order observed from $nt = 2$ to $nt = 4$, and the reduced order in further steps.

Since the AMIB algorithm achieves second order accuracy for both spatial and temporal discretization, we test the computing efficiency by refining the spatial and temporal steps simultaneously. See the Table 6.10 with stopping time $T = 1$. We can observe that the second overall accuracy of the three methods are all achieved after the spatial and temporal partition are refined at the same time. Such detected second order accuracy can validate the temporal second order despite the reduced order observation in the Table 6.9. The CPU time comparison of the

Table 6.9: Temporal convergence for example 3.

nt	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	4.83E-5	–	2.18E-4	–	4.1
4	1.39E-5	1.80	5.61E-5	1.96	7.9
8	5.22E-6	1.41	1.92E-5	1.55	15.9
16	2.68E-6	0.96	9.40E-6	1.03	31.5
32	2.15E-6	0.32	8.56E-6	0.14	66.6
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	4.62E-5	–	2.13E-4	–	62.7
4	1.23E-5	1.91	5.46E-5	1.96	63.2
8	4.25E-6	1.53	1.60E-5	1.77	63.5
16	2.57E-6	0.73	9.15E-6	0.81	67.6
32	2.23E-6	0.20	8.88E-6	0.04	68.1

Table 6.10: Efficiency test for example 3.

(nx,nt)	AMIB-GMRES						
	L_2		L_∞		CPU time(s)	time ratio	ANMG
	Error	Order	Error	Order			
$(65,20)$	1.53E-4	–	5.08E-4	–	0.25		31.3
$(129,40)$	3.63E-5	2.08	1.39E-4	1.87	2.22	8.9	35.2
$(257,80)$	8.63E-6	2.07	3.52E-5	1.98	26.7	12.0	48.1
$(513,160)$	2.16E-6	2.00	8.88E-6	1.99	342.9	12.8	57.3
	AMIB-LU						
	L_2		L_∞		CPU time(s)	time ratio	
	Error	Order	Error	Order			
$(65,20)$	1.53E-4	–	5.08E-4	–	0.15		15.2
$(129,40)$	3.63E-5	2.08	1.39E-4	1.87	1.15	7.7	15.2
$(257,80)$	8.63E-6	2.07	3.52E-5	1.98	10.1	8.8	15.2
$(513,160)$	2.14E-6	2.01	8.79E-6	2.00	100.8	10	15.2
	MIB						
	L_2		L_∞		CPU time(s)	time ratio	
	Error	Order	Error	Order			
$(65,20)$	1.47E-4	–	5.05E-4	–	0.43		
$(129,40)$	3.62E-5	2.08	1.42E-4	1.87	5.94	13.8	
$(257,80)$	8.53E-6	2.07	3.56E-5	1.98	113.4	19.1	
$(513,160)$	NA	NA	NA	NA	NA	NA	

three methods further demonstrate the better efficiency of the two AMIB method over MIB method. Besides, the time ratios of MIB method with mesh refinement are higher than the two AMIB methods, meaning the MIB is the most expansive algorithm among the three. AMIB-LU seems a little more efficient than AMIB-GMRES thanks to its relatively smaller time ratio when the mesh is refined. The reason for its better efficiency is mainly due to smaller average number of multigrid(ANMG) solver being used in each time step as recorded in the last column in Table 6.10. It can be seen that the average number for AMIB-GMRES mildly increases as the mesh is refined. This is due to the fact that the iteration number of GMRES solver in solving (6.31) weakly depends on the grid numbers. However, the average number remains constant for

AMIB-LU, since the majority of the number of multigrid solver used is on forming the explicit matrix of left side of (6.31).

It can be noticed that the numerical results for MIB in the last row of Table 6.10 are not available due to failure of convergence from the biconjugate gradient solver [62]. The non-convergence is not caused by stability issue on the algorithm, as a different iterative solver, such as GMRES, can produce convergent iterative solution in our numerical test. This demonstrated that the performance of the AMIB methods does not heavily depend on the iterative solvers, while the MIB method does.

Example 4. We are interested in the stability performance of the two methods on complicated interface shape. In this example, we focus on the case of interface shape governed by parametric function

$$r = 0.5 + 0.1\sin(5\theta), \tag{6.44}$$

embedded in the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$.

The exact solution to this problem is

$$u(x,y) = \begin{cases} \cos(t)e^{-x-y}, & \text{in } \Omega^-, \\ \cos(t)e^{x+y}, & \text{in } \Omega^+, \end{cases} \tag{6.45}$$

with the diffusion coefficients

$$\beta = \begin{cases} 1, & \text{in } \Omega^-, \\ 100, & \text{in } \Omega^+. \end{cases}$$

The spatial convergence is validated by refining the mesh with the stopping time $T = 2$, and time increment $\Delta t = 0.002$. In Table 6.11, the second order accuracy can be observed for both AMIB methods.

We next examine the temporal order of the two AMIB methods by setting $n_x = 512$, and the stopping time as $T = 0.001$. The second order temporal convergence can be observed in Table

Table 6.11: Spatial convergence for example 4.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.08E-3	–	6.81E-3	–	1.67
[64, 64]	6.26E-4	2.70	1.12E-3	2.60	12.5
[128, 128]	2.26E-4	1.47	3.92E-4	1.51	89
[256, 256]	9.01E-5	1.33	1.54E-4	1.35	524
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.08E-3	–	6.81E-3	–	0.48
[64, 64]	6.26E-4	2.70	1.12E-3	2.60	2.50
[128, 128]	2.25E-4	1.47	3.91E-4	1.52	12.4
[256, 256]	8.83E-5	1.33	1.51E-4	1.37	57

6.12 from *nt* equal to 2 up to 16.

Table 6.12: Temporal convergence for example 4.

nt	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	3.28E-3	–	5.60E-3	–	5.2
4	8.22E-4	2.00	1.38E-3	2.02	10.1
8	2.23E-4	1.88	3.65E-4	1.92	20.2
16	7.45E-5	1.58	1.16E-4	1.65	40.1
32	3.75E-5	0.99	6.05E-5	0.94	76.6
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
2	3.28E-3	–	5.60E-3	–	75.8
4	8.21E-4	2.00	1.38E-3	2.02	75.3
8	2.23E-4	1.88	3.64E-4	1.92	77.7
16	7.41E-5	1.59	1.15E-4	1.66	80.1
32	3.75E-5	0.98	6.10E-5	0.94	82.7

To numerically analyze the stability of the two AMIB schemes, we examine different time stepping. We choose $n_x = 128$ and stopping time T equal to $10^4\Delta t$ as shown in Fig. 6.2. All numerical solutions are found to be stable with large temporal increments. And the corresponding numerical errors are bounded as shown in Fig. 6.2. This justifies the reliable stability of our methods for solving parabolic interface problem.

Example 5. We next study a problem with multiple subdomains. Consider a 2D parabolic equation

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

defined in a square domain $\Omega = [-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$, which is divided into three regions by two

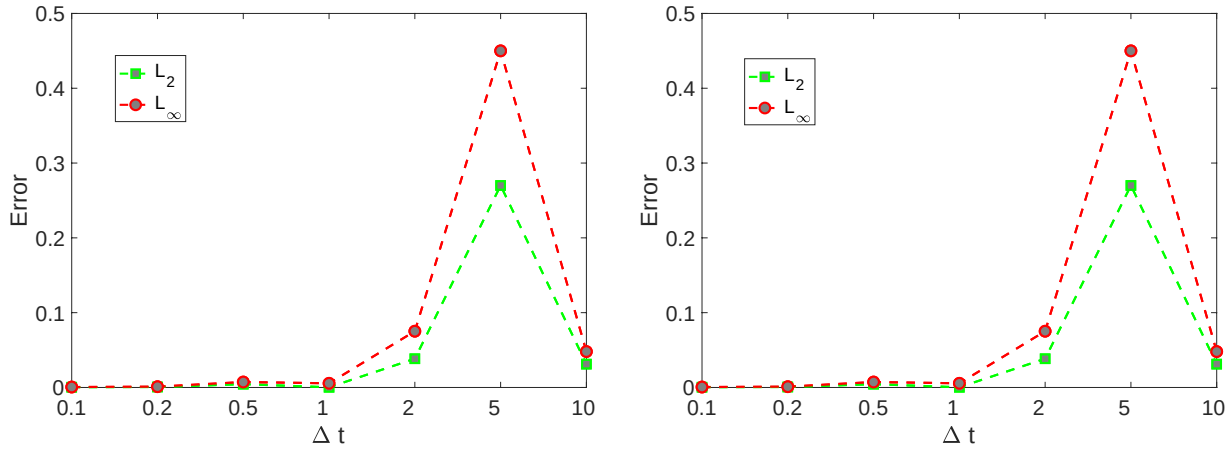

Figure 6.2: Stability test for the two AMIB methods. The left one is for AMIB-GMRES, and the right one is for AMIB-LU. All the numerical errors are bounded with $n_x = 128$, and $T = 10^4 \Delta t$.

ellipse shape interface:

$$\Gamma_L : \left(\frac{x+\frac{\pi}{6}}{\frac{\pi}{9}}\right)^2 + \left(\frac{y+\frac{\pi}{7}}{\frac{\pi}{12}}\right)^2 = 1$$
$$\Gamma_R : \left(\frac{x-\frac{\pi}{6}}{\frac{\pi}{12}}\right)^2 + \left(\frac{y-\frac{\pi}{7}}{\frac{\pi}{9}}\right)^2 = 1$$

Three pieces of solutions are defined in each subdomain

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

where parameter k is set to be 5. 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 second order spatial and temporal accuracy of solutions have been examined through above examples. Besides, we want to further investigate the convergence rate of the gradient recovery. Second order central difference is adopted to approximate the gradients with fictitious values replacing obtained numerical real function values when the stencil cuts through the interface. Table 6.13 shows that second order convergence is achieved for both solutions and gradients from the three methods in solving the multi-domain problem. With the confirmed second orders of the three methods, we pay attention to the comparison of the methods on overall efficiency for solving multi-domain problem . Table 6.13 demonstrates the efficiency test, where it can be observed that AMIB-LU outperform the other methods for solving such problems. The last column of Table 6.13 shows that AMIB-LU has smaller time ratio than AMIB-GMRES or AMIB with mesh refinement. In particular, the computing time by MIB is almost 19 times that of AMIB-LU in dealing the multi-domain problem on the $(nx, nt) = (513, 160)$ mesh, further demonstrating the efficiency improvement from AMIB-LU.

Example 6. We investigate a physical problem modeled by the heat equation without analytical solutions

$$u_t = \nabla \cdot (\beta \nabla u)$$

subject to the homogeneous interface condition

$$[u] = 0 \tag{6.46}$$

$$[\beta u] = 0 \tag{6.47}$$

on the interface $r^2 := x^2 + y^2 = \frac{1}{4}$ embedded in the domain $[-\frac{\pi}{3}, \frac{\pi}{3}] \times [-\frac{\pi}{3}, \frac{\pi}{3}]$. The diffusion coefficients

$$\beta = \begin{cases} 100, & \text{in } \Omega^-, \\ 1, & \text{in } \Omega^+. \end{cases}$$

Table 6.13: Efficiency test for example 5..

(n_x, nt)	AMIB-GMRES					
	Solution					
	L_2		L_∞		CPU time(s)	time ratio
	Error	Order	Error	Order		
(65, 20)	3.00E-2	–	7.43E-2	–	0.37	
(129, 40)	8.96E-3	1.74	2.24E-2	1.73	3.96	10.7
(257, 80)	1.57E-3	2.51	4.02E-3	2.48	44.46	11.2
(513, 160)	3.23E-4	2.28	8.28E-4	2.28	562	12.6
	Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
(65, 20)	9.38E-2	–	0.37	–		
(129, 40)	2.83E-2	1.732	0.12	1.62		
(257, 80)	5.07E-3	2.48	2.15E-2	2.48		
(513, 160)	1.21E-3	2.07	5.29E-3	2.02		
	AMIB-LU					
	Solution					
	L_2		L_∞		CPU time(s)	time ratio
	Error	Order	Error	Order		
(65, 20)	3.00E-2	–	7.43E-2	–	0.15	
(129, 40)	8.95E-3	1.75	2.23E-2	1.74	1.22	8.1
(257, 80)	1.58E-3	2.50	4.05E-3	2.46	11.7	9.6
(513, 160)	3.71E-4	2.09	9.54E-4	2.09	127	10.9
	Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
(65, 20)	9.38E-2	–	0.37	–		
(129, 40)	2.82E-2	1.73	0.12	1.62		
(257, 80)	5.11E-3	2.46	2.17E-2	2.47		
(513, 160)	1.21E-3	2.08	5.29E-3	2.04		
	MIB					
	Solution					
	L_2		L_∞		CPU time(s)	time ratio
	Error	Order	Error	Order		
(65, 20)	3.61E-2	–	9.21E-2	–	0.44	
(129, 40)	1.10E-2	1.74	2.87E-2	1.68	7.38	16.8
(257, 80)	1.41E-3	2.46	3.90E-3	2.88	143	19.4
(513, 160)	2.17E-4	2.09	9.89E-4	1.98	2372	16.6
	Gradient					
	L_2		L_∞			
	Error	Order	Error	Order		
(65, 20)	0.12	–	1.93	–		
(129, 40)	3.53E-2	1.76	0.44	2.13		
(257, 80)	4.51E-3	2.97	0.13	1.76		
(513, 160)	9.73E-4	2.21	2.42E-2	2.43		

The Dirichlet boundary condition equal to zero is imposed at the boundary of given domain. Besides, the initial condition is defined as

$$u(x,y) = \begin{cases} e^{-(x^2+y^2)/\sigma^2} & \text{in } \Omega^-, \\ 0, & \text{in } \Omega^+, \end{cases} \tag{6.48}$$

with parameter $\sigma = 0.1$. As the analytical solution is not readily available, we generate the reference solution on certain mesh to examine the numerical performance of the two AMIB

methods.

The spatial convergence is validated by refining the mesh under time increment $\Delta t = 0.001$ for stopping time $T = 1$. The reference solution is created on the mesh $(nx, \Delta t) = (512, 0.0005)$. The second order accuracy can be observed for both AMIB methods in Table 6.14.

Table 6.14: Spatial convergence for example 6.

$[n_x, n_y]$	AMIB-GMRES				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.09E-5	–	6.66E-5	–	0.65
[64, 64]	2.24E-5	0.87	3.63E-5	0.88	4.02
[128, 128]	5.04E-6	2.15	8.19E-6	2.15	22.3
[256, 256]	9.47E-7	2.41	1.56E-6	2.39	151
	AMIB-LU				
	L_2		L_∞		CPU time(s)
	Error	Order	Error	Order	
[32, 32]	4.09E-5	–	6.66E-5	–	0.25
[64, 64]	2.23E-5	0.88	3.62E-5	0.88	1.36
[128, 128]	5.06E-6	2.14	8.22E-6	2.24	7.06
[256, 256]	9.97E-7	2.34	1.63E-6	2.33	34.9

The temporal convergence is validated in a similar fashion by refining time increment for stopping time $T = 1$, and the spatial discretization is based on mesh $(nx, ny) = (256, 256)$. The reference solution is obtained on the mesh $(nx, \Delta t) = (512, 0.0005)$. Due to the practical physical phenomenon modeled by heat equation, the temperature quickly drops in a very short time, leading to large approximation error when large time step size is adopted as shown in Fig. 6.3. Nevertheless, small time step can give rise to well approximated solution. This is true for both AMIB-GMRES and AMIB-LU method, especially when $nt > 160$ is employed. For such reason, it is difficult to observe the expected second order temporal convergence.

The figure 6.4 shows solution evolution over time for this physical model. As discussed before, the temperature drops in a short time. This can be observed in Fig. 7 when time is from 0 to 0.1. By time equals to 1, the temperature has dropped to a magnitude of 10^{-4}.

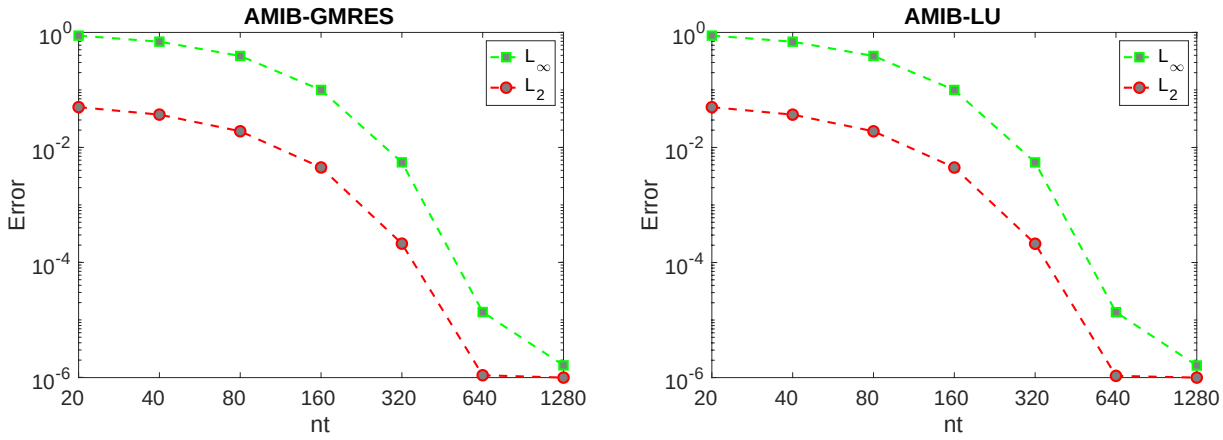

Figure 6.3: The numerical errors from two AMIB methods.

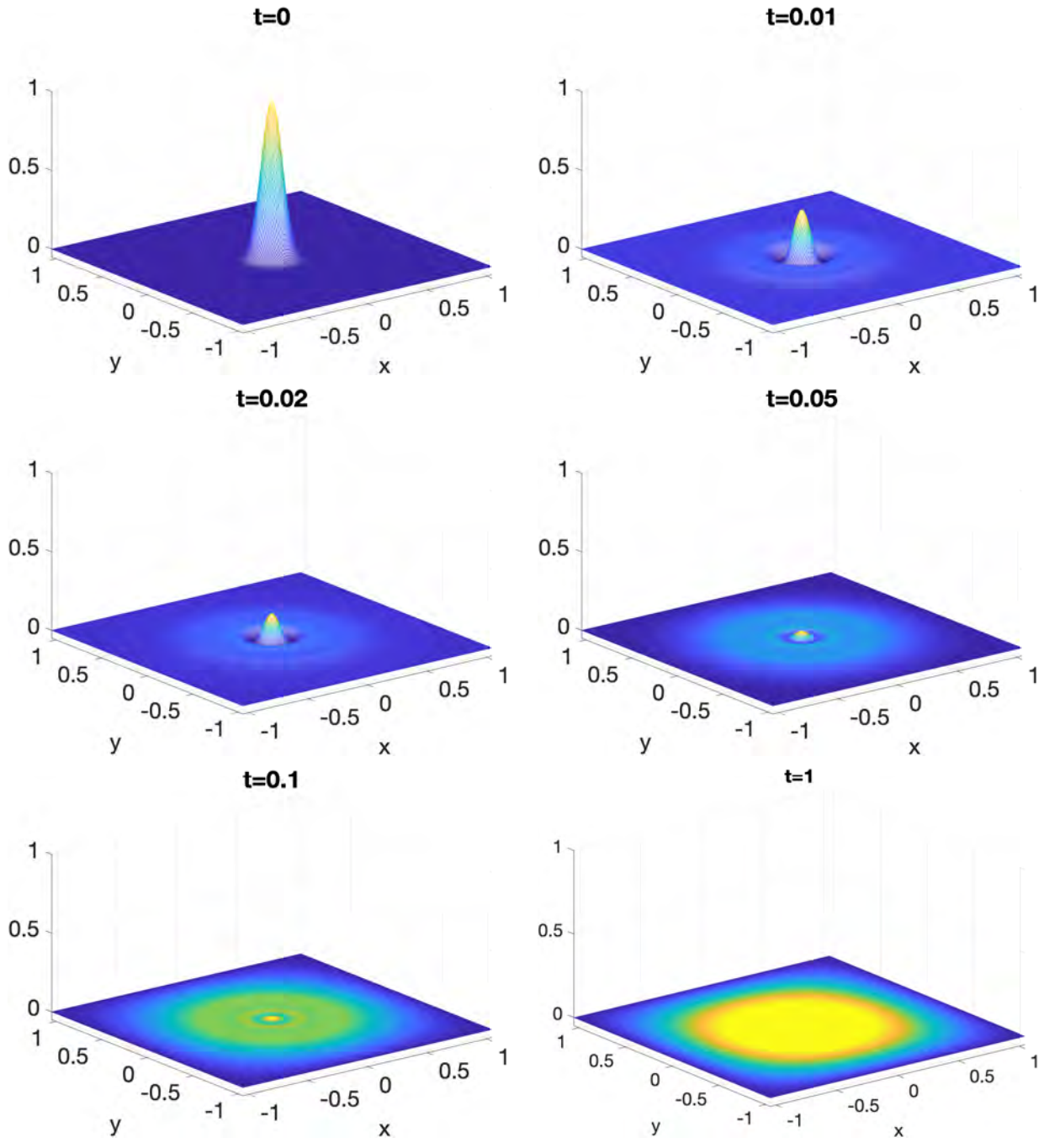

Figure 6.4: The solution of heat equation evolves within time equal to 1. The rapid change during initial period indicates quick temperature drop.

CHAPTER 7

CONCLUSIONS

This dissertation develops an accurate and efficient algorithm, the augmented matched interface and boundary (AMIB) method, for solving various interface problems. The theory of AMIB method is demonstrated, and the efficiency performance of AMIB is exhibited through solving several problems. 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. Such approximations through fictitious values are introduced as auxiliary variables in the augmented MIB (AMIB) formulation. The Schur complement method allows us to iteratively solve these auxiliary variables over a one-dimensionally smaller algebraic system. Consequently, the augmented system in the approach provides an efficient way to solve the interface and boundary value problem.

AMIB method firstly shows its efficiency through the second order algorithm for solving elliptic interface problem with various complex interfaces. It has 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 $O(n \log n)$ for degree of freedom n on the given computational domain.

Secondly, a FFT-based high order fast Poisson solver is developed for a cubic type geometry in multi-dimensions. By introducing a thin fictitious buffer zone or zero-padding region outside the domain, the Poisson boundary value problem is converted into an immersed boundary problem, so that FFT inversion of high order central difference schemes becomes feasible. The boundary conditions then become interface conditions, which are discretized by the matched interface and boundary (MIB) method for generating fictitious nodes. Moreover, the iteration number for the

Schur complement system weakly depends on the degree of freedom N in each dimension. Hence, even though the AMIB method involves an iteration process, which is essentially not a direct fast solver, it still gives a fast Poisson solution. The numerical flop counts of the AMIB method are the order of $O(N^2 \log N)$ for a two-dimensional problem. Traditional high-order fast Poisson solvers are mostly based on compact finite difference or spectral methods. This study is the first of its kind based on high order central differences. As a systematical approach, it can easily handle the usual Dirichlet, Neumann, Robin or any combination of boundary conditions. Moreover, based on tensor product principle, the AMIB method can be implemented in multi-dimensions. Furthermore, the AMIB method can be made to arbitrarily high order in principle. Up to 8th order is demonstrated in this work. As in a similar research filed concerned with Poisson boundary value problem, current AMIB method naturally solve Poisson BVP on staggered boundaries, which has great popularity in mechanical or engineering problem.

Thirdly, fourth order AMIB algorithm is proposed to solve two-dimensional elliptic interface problems with help of proposed fast Poisson BVP solver mentioned above. It is robust and efficient enough for treating smoothly curved interfaces and handling various boundary conditions. At both interfaces and boundaries, fourth order corrected central difference are used for discretization Laplacian operator. The 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.

Fourthly, AMIB method is used to solve parabolic interface problem with second order accuracy

achieved in both time and space. The efficiency is fulfilled with multigrid method built in the Schur complement process in the framework of an augmented system. A simple multigrid solver is designed to solve the linear system due to the special structure of coefficient matrix as part of the augmented system. Two approaches based on the multigrid solver are proposed in the time-step process, with one incorporated in GMRES iteration for each time evolution, and the other one used in the coefficient matrix formation of LU decomposition solver in the first time step. The latter approach with LU decomposition can be more efficient especially when a lot of time-stepping evolution are needed for solving the given parabolic interface problem. Both AMIB methods are significantly faster than the standard MIB method. Moreover, being free of stability constraint, the present implicit method could be asymptotically faster than explicit schemes. Finally, we note that due to broad application of high order central difference schemes, the AMIB method has potential to be applied to many problems with ease to gain a fast solution, including more complicated boundary conditions and boundary value problems over irregular domains. This is the future study plan of AMIB method. With the development of AMIB taking advantage of the maturity and success of MIB method, the further exploration of AMIB in solving practical problems can advance with the growth of MIB method.

REFERENCES

[1] L. Adams, T.P. Chartier, New geometric immersed interface multigrid solvers, SIAM J. Sci. Comput. 25 (2004) 1516–1533.

[2] L. Adams, Z.L. Li, The Immersed Interface/Multigrid Methods for Interface Problems, SIAM J. Sci. Comput. 24(2002) 463-479.

[3] A. Averbuch, M. Israeli, and L. Vozovoi, A fast Poisson solver of arbitrary order accuracy in rectangular regions, SIAM J. Sci. Comput., 19(1998) 933-952.

[4] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, Computing, 5(1970) 207–213.

[5] N.A. Baker, Poisson-Boltzmann methods for biomolecular electrostatics, Meth. Enzymlg. 383(2004) 94-118.

[6] 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.

[7] D. Bochkov and F. Gibou, Solving elliptic interface problems with jump conditions on Cartesian grids, J. Comput. Phys. 407 (2020) 109269.

[8] Jacob Bedrossian, James Brecht, Siwei Zhu and Eftychios Sifakis and Joseph Teran, A Second Order Virtual Node Method for Poisson Interface Problems on Irregular Domains, J. Comput. Phys. 229(2010) 6405–6426.

[9] E. Braverman, M. Israeli, A. Averbuch, and L. Vozovoi, A fast 3D Poisson solver of arbitrary order accuracy, J. Comput. Phys., 144(1998) 109-136.

[10] E. Braverman, M. Israeli, and A. Averbuch, A fast spectral solver for a 3D Helmholtz equation, SIAM J. Sci. Comput., 20(1999) 2237-2260.

[11] O. Bruno and M. Lyon, High-order unconditionally stable FC-AD solvers for general smooth domains I. Basic elements. J. Comput. Phys. 229 (2010) 2009-2033.

[12] A. Bruger, J. Nilsson, W. Kress, A Compact Higher Order Finite Difference Method for the Incompressible Navier–Stokes Equations, J. Sci. Comput. 17 (2002) 551-560.

[13] E. Burman and P. Hansbo, Interior-penalty-stabilized Lagrange multiplier methods for the finite-element solution of elliptic interface problems, IAM J. Numer. Anal. 30(2010) 870–885.

[14] L. Chen, H. Wei, and M. Wen, An interface-fitted mesh generator and virtual element methods for elliptic interface problems, J. Comput. Phys. 334(2017) 327-348.

[15] T. Chen and J. Strang, Piecewise-polynomial discretization and Krylov-accelerated multigrid for elliptic interface problems, J. Comput. Phys. 227(2008) 7503-7542.

[16] Z. Chen, J. Zhou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numerische Mathematik. 79(1998) 175-202.

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

[18] A. Coco, and 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.

[19] J. Douglas, D. Peaceman, Numerical solution of two-dimensional heat-flow problems, AIChEJ.1(4)(1955)505–512.

[20] M. Dryjaa, J. Galvisb, and M. Sarkisb, BDDC methods for discontinuous Galerkin discretization of elliptic problems, Journal of Complexity, 23(2007) 715–739.

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

[22] R. E. Ewing, Z. L. Li, T. Lin, and Y. P. Lin, The immersed finite volume element methods for the elliptic interface problems, Mathematics and Computers in Simulation 50(1999) 63–76.

[23] 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.

[24] H. Feng, G. Long, and S. Zhao, An augmented matched interface and boundary (MIB) method for solving elliptic interface problem. J. Comput. Appl. Math. 361(2019), 426-423.

[25] H. Feng, G. Long, and S. Zhao, FFT-based high order central difference schemes for Poisson's equation with staggered boundaries, J. Sci. Comput., 86 (2021)7.

[26] H. Feng and 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.

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

[28] M. Frigo and S. G. Johnson, The design and implementation of FFTW3, Proc. IEEE, 93(2005) 216-231.

[29] Y. Ge, Multigrid method and fourth-order compact difference discretization scheme with unequal meshsizes for 3D Poisson equation, J. Comput. Phys., 229(2010) 6381-6391.

[30] 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.

[31] G. H. Golub, L. C. Huang, H. Simon, and W. Tang, A fast Poisson solver for the finite difference solution of the incompressible Navier-Stokes equations, Siam J. Comput. 19(1998) 1606-1624.

[32] A. Guittet, M. Lepilliez, S. Tanguy, and F. Gibou, Solving elliptic problems with discontinuities on irregular domains - the Voronoi interface method, J. Comput. Phys. 298(2015) 747-765.

[33] 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.

[34] A. Hansbo and P. Hansbo, An unfitted finite element method, Comput. Methods Appl. Mech. Engng. 191(2002) 5537-5552.

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

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

[37] S. M. Hou, P. Song, L. Q. Wang, and H. K. Zhao, A weak formulation for solving elliptic interface problems without body fitted grid, J. Comput. Phys. 249(2013) 80–959.

[38] 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.

[39] L. N. T. Huynh, N. C. Nguyen, J. Peraire, and B. C. Khoo, A high-order hybridizable discontinuous Galerkin method for elliptic interface problems, Inter. J. Numer. Meth. Enginn. 93(2013) 183–200.

[40] N. A. Kampanis, J. A. Ekaternaris, A staggered grid, high-order accurate method for the incompressible Navier–Stokes equations, J. Comput. Phys. 215(2006) 589-613.

[41] M. C. Lai, A simple compact fourth-order Poisson solver on polar geometry, J. Comput. Phys., 182(2002) 337-345.

[42] 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.

[43] C. Li, Z. Wei, G. Long, C. Campbell, S. Ashlyn, and S. Zhao, Alternating direction ghost-fluid methods for solving the heat equation with interfaces, Comput. Math. with Appl., 80(2020) 714-732.

[44] 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.

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

[46] Z. Li, A.Mayo, ADI methods for heate quations with discontinuous along an arbitrary interface,in:Proc. Sympos. Appl. Math.,48(1993)311–315.

[47] Z. Li, X. Chen, Z. Zhang, On multiscale ADI methods for parabolic pdes with a discontinuous coefficient, SIAM Multiscale Model. Simul. 16 (4) (2018) 1623–1647.

[48] Z.L. Li, K.Ito, Maximum principle preserving schemes for interface problems with discontinuous coefficients, SIAM J. Sci. Comput. 23(2001) 339-361.

[49] Z.L. Li, K.Ito, The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains, siam, 2006/7/1

[50] Z.L. Li, H. Ji, and X. Chen, Accurate solution and gradient computation for elliptic interface problems with variable coefficients, SIAM J. Numer. Anal. 55(2017) 670-697.

[51] C. Liu and C. Hu, A second order ghost fluid method for an interface problem of the Poisson equation, Commun. Comput. Phys.22(2017) 965-996.

[52] J. Liu, Z. Zheng, IIM-Based ADI finite difference scheme for nonlinear convection–diffusion equations with interfaces, Appl. Math. Model. 37 (3) (2013) 1196–1207.

[53] J. Liu, Z. Zheng, A dimension by dimension splitting immersed interface method for heat conduction equation with interfaces, J. Comput. Appl. Math. 261 (2014) 221–231.

[54] X.D. Liu, R.P. Fedkiw, M. Kang, boundary condition capturing method for Poisson's equation on irregular domain, J. Comput. Phys. 160(2000) 151-178.

[55] M. Lyon and 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.

[56] L. Mu, J. Wang, G.W. Wei, X. Ye, and S. Zhao, Weak Galerkin methods for second order elliptic interface problems, J. Comput. Phys. 250(2013) 106–125.

[57] 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.

[58] J.R. Nagel, Solving the Generalized Poisson's Equation Using the Finite-Difference Method (FDM). Department of Electrical and Computer Engineering, University of Utah, Salt Lake City (2011).

[59] D. Paeceman, H. Rachford, The numerical solution of parabolic and elliptic equations, J. Soc. Ind. Appl. Math.3(1955)28–41.

[60] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25(1977) 220-252

[61] C.S. Peskin, Lectures on mathematical aspects of physiology, Lect. Appl. Math. 19(1981) 69-107.

[62] W. H. Press, S. A Teukolsky , Numerical recipes in FORTRAN (2nd ed.): the art of scientific computing, January 1992.

[63] J. Shen, T. Tang, and L. L. Wang, Spectral methods: Algorithm, Analysis and Application, Springer series in computational mathematics, 2011.

[64] D. B. Stein, R. D. Guy, and 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.

[65] J. C. Strikwerda, finite difference schemes and partial differential equations. SIAM.

[66] P. Swarztrauber and R. Sweet, Algorithm 541: Efficient Fortran subprograms for the solution of separable elliptic partial differential equations, ACM T. Math. Software (TOMS), 5(1979), 352-364.

[67] J.W.L. Wan, X.-D. Liu, A boundary condition-capturing multigrid approach to irregular boundary problems, J. Sci. Comput. 25 (2004) 1982–2003.

[68] H. Wang, Y. Zhang, X. Ma, J. Qiu and Y. Liang, An efficient implementation of fourth-order compact finite difference scheme for Poisson's equation with Dirichlet boundary conditions, Comput. Math. Appl., 71(2016) 1843-1860.

[69] Y. Wang and J. Zhang, Sixth order compact scheme combined with multigrid method and extrapolation technique for 2D Poisson equation, J. Comput. Phys., 228(2009)137-146.

[70] H. Wei, L. Chen, Y, Huang, and B. Zheng, Adaptive mesh refinement and superconvergence for two-dimensional interface problems, SIAM J. Sci. Comput. 36(2014) A1478-A1499.

[71] Z. Wei, C. Li, and S. Zhao, A spatially second order alternating direction implicit (ADI) method for three dimensional parabolic interface problems, Computers and Mathematics with Applications, 75(2018) 2173-2192.

[72] A. Wiegmann and 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.

[73] K. N. Xia, M. Zhan, and G.-W. Wei, MIB Galerkin method for elliptic interface problems. Journal of Computational and Applied Mathematics, 272(2014) 195–220.

[74] Y. Xie, and W. Ying, A Fourth-Order Kernel-Free Boundary Integral Method for the Modified Helmholtz Equation, J. Sci. Comput., 78(2019) 1632–1658.

[75] W. J. Ying and W. C. Wang, A kernel-free boundary integral method for implicitly defined surfaces, J. Comput. Phys. 252(2013) 606-624.

[76] K.K.Q. Zhang, B. Shotorban, W.J. Minkowycz,F. Mashayek, A compact finite difference method on staggered grid for Navier-Stokes flow, international journal for numerical methods in fluids, 52(2006) 867–881.

[77] S. Zhao, On the spurious solutions in the high-order finite difference methods, Comput. Method Appl. M., 196(2007) 5031-5046.

[78] S. Zhao, A fourth order finite difference method for waveguides with curved perfectly conducting boundaries, Comput. Method Appl. M., 199(2010) 2655-2662.

[79] 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.

[80] S. Zhao, A matched alternating direction implicit (ADI) method for solving the heat equation with interfaces, J. Sci. Comput., 63 (2015), 118-137.

[81] S. Zhao and G. W. Wei, Matched interface and boundary (MIB) for the implementation of boundary conditions in high order central finite differences, Int. J. Numer. Meth. Eng., 77(2009) 1690-1730.

[82] S. Zhao and G. W. Wei, High-order FDTD methods via derivative matching for Maxwell's equations with material interfaces, J. Comput. Phys., 200(2004) 60-103.

[83] S. Zhao, G. W. Wei, and Y. Xiang, DSC analysis of free-edged beams by an iteratively matched boundary method, J Sound Vib., 284(2005) 487-493.

[84] 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.

[85] 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

[86] Y.C. Zhou, G.W. Wei, On the fictitious-domain and interpolation formulations of the matched interface and boundary (MIB) method, J. Comput. Phys. 219(2006) 228-246.