EFFICIENT ALGORITHMS FOR SOLVING THREE DIMENSIONAL PARABOLIC INTERFACE PROBLEM WITH VARIABLE COEFFICIENTS

by

ZHIHAN WEI
SHAN ZHAO, COMMITTEE CHAIR
LAYACHI HADJI
MOJDEH RASOULZADEH
ROBERT BROOKS
WEI ZHU

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

2018
ABSTRACT

The dissertation consists of two parts, in the first part, a new matched alternating direction implicit (ADI) method is proposed for solving three-dimensional (3D) parabolic interface problems with discontinuous jumps, piecewise constant diffusion coefficients and complex interfaces. This scheme inherits the merits of its ancestor of two-dimensional problems, while possesses several novel features, such as a non-orthogonal local coordinate system for decoupling the jump conditions, two-side estimation of tangential derivatives at an interface point, and a new Douglas-Rachford ADI formulation that minimizes the number of perturbation terms, to attack more challenging 3D problems. In time discretization, this new ADI method is found to be first order and stable in numerical experiments. In space discretization, the matched ADI method achieves a second order of accuracy based on simple Cartesian grids for various irregularly-shaped surfaces and spatial-temporal dependent jumps. Computationally, the matched ADI method is as efficient as the fastest implicit scheme based on the geometrical multigrid for solving 3D parabolic equations, in the sense that its complexity in each time step scales linearly with respect to the spatial degree of freedom $N$, i.e., $O(N)$. Furthermore, unlike iterative methods, the ADI method is an exact or non-iterative algebraic solver which guarantees to stop after a certain number of computations for a fixed $N$. Therefore, the proposed matched ADI method provides an efficient tool for solving 3D parabolic interface problems.

In the second part, instead of constant diffusion coefficients, improved schemes for variable diffusion coefficient are also performed in the work. A comparison of proposed ADI method with different other time splitting methods, including Trapezoidal Splitting (TS) method, locally one-dimensional implicit Euler (LOD-IE) and locally one-dimensional Crank-Nicolson (LOD-CN) will be implemented, coupled with different variation of matched interface and boundary (MIB) method in spatial discretization. These large scale computational studies
facilitate the further development of matched ADI algorithms for 3D parabolic interface problems.

**Keyword:** Parabolic interface problem; Matched alternating direction implicit (ADI) method; Douglas-Rachford ADI scheme; Locally one-dimensional implicit Euler (LOD-IE) method; Locally one-dimensional Crank-Nicolson (LOD-CN) method; Ghost Fluid method (GFM); Trapezoidal Splitting (TS) method; Matched interface and boundary (MIB).
DEDICATION

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

I am deeply indebted to my advisor Dr. Shan Zhao for his fundamental role in my doctoral work. Dr. Zhao provided me with every bit of guidance, assistance, and expertise that I needed during my entire Ph.D. career. Dr. Zhao gave me the freedom to do whatever I wanted, at the same time continuing to contribute valuable feedback, advice and encouragement. In addition to our academic collaboration, I greatly value the close personal rapport that Dr. Zhao and I have forged over the years. I quite simply cannot imagine a better advisor.

I would like to thank Dr. Wei Zhu, Dr. Layachi Hadji, Dr. Mojdeh Rasoulzadeh and Dr. Robert Brooks for their support to accomplish my dissertation work. Meanwhile, I am really appreciated the knowledge I have learned in their classes.

I would like to thank all the staffs, faculties and graduate students in the math department of The University of Alabama, expecially Dr. Mingwei Sun, Dr. Xin Luo and Ms. Natalie Lau. It is not hard to imagine how life could easily go wrong without their help.

I am very grateful to have my cat Chestnut, in addition to the times he messed up my code and paper writing. I still have a fresh memory how lovely the kitten was when I took him all the way from Kansas City. Thanks for the time he spent with me in my office and home and all the joys he gave.

To my father Jianrui Wei and my mother Ying Zhu, words and sentences are meaningless here to express my love to them and how I appreaciate the way they love me. They are not only my family members, but also my mentors, my greatest friends. Thanks to my cousin Jie Ning, her brilliant always set an example to me from my childhood. Thanks to my grandmother Shufen Shen, thanks to the way she attached great importance to education, which affect the entire family through all generations.
Last but not least, thanks God. Your word is lamp for my feet and a light for my path (Psalm 119:105).
CONTENTS

ABSTRACT ................................................................. ii
DEDICATION .............................................................. iv
ACKNOWLEDGMENTS ................................................... v
LIST OF TABLES .......................................................... ix
LIST OF FIGURES ........................................................ x
CHAPTER 1 INTRODUCTION ........................................... 1
CHAPTER 2 THREE DIMENSIONAL MATCHED ADI METHOD WITH PIECE-WISE
CONSTANT DIFFUSION COEFFICIENTS ............................ 7
  2.1 ADI semi-discretization .......................................... 7
  2.2 Spatial discretization of the 3D matched ADI method .......... 9
      2.2.1 Local coordinate and non-orthogonal tangential directions .. 9
      2.2.2 Tensor product decomposition of jump conditions ............ 12
      2.2.3 Tangential derivative approximations ....................... 13
      2.2.4 1D jump conditions enforcement ............................ 16
  2.3 A new ADI method with less perturbation terms ................ 18
CHAPTER 3 COMPARISON OF DIFFERENT TIME SPLITTING METHODS WITH
VARIABLE DIFFUSION COEFFICIENTS .......................... 21
  3.1 Enforcing 3D jump conditions in 1D manner .................. 22
      3.1.1 Parabolic interface problem with variable coefficients .... 22
      3.1.2 Euler discretization with central finite difference .......... 23
      3.1.3 Three-dimensional jump condition decomposition .......... 24
  3.2 Different spatial discretization ............................... 25
      3.2.1 Ghost fluid method (GFM) .................................. 26
LIST OF TABLES

4.1 Spatial convergence for ellipsoid surface. ........................................... 40
4.2 Spatial convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 10$. ........................................... 42
4.3 Spatial convergence for cylinder surface with $\alpha^- = 10$ and $\alpha^+ = 1$. ........................................... 43
4.4 Spatial convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 100$. ........................................... 44
4.5 Spatial convergence for molecular surface. ........................................... 45
4.6 Spatial convergence for torus surface. ........................................... 48
4.7 Spatial convergence for tanglecube surface. ........................................... 50
4.8 Spatial convergence for heart surface. ........................................... 53
LIST OF FIGURES

2.1 Non-orthogonal tangential directions η and ζ. Here at an interface point with the normal direction ξ (red), η and ζ are determined by intersecting the tangential plane (turquoise) with the place $x = x_i$ (yellow) and $z = z_k$ (mauve), respectively. 11

2.2 Tangential derivative approximation and MIB interface treatment. (a). The tangential derivative at the interface point (purple square) is approximated through two auxiliary points (purple rectangles). Each auxiliary value will be extrapolated by three supporting nodes from the same side (positive or negative). An optimization procedure is designed to automatically select either positive or negative side for approximating tangential derivatives. The MIB 1D jump condition enforcement is also illustrated, which involves four nodes (black filled circles) and two fictitious nodes (red open circles). (b). In a corner MIB treatment of 1D jump conditions, five nodes (black filled circles) and three fictitious nodes (red open circles) are involved. Here two interface points $(x_i, y_{Γ1}, z_k)$ and $(x_i, y_{Γ2}, z_k)$ are the intersection points of $Γ$ with two dashed lines. 15

3.1 Three types of points: regular, irregular and corner point. (a). Green curve is interface $Γ$ which separates $Ω^-$ and $Ω^+$, two orange nodes lies next to interface $Γ$ are called irregular points and the other two black nodes are regular points. (b). Green curve is still interface $Γ$ with high curvature, which separates inside sub-domain $Ω^-$ and outside sub-domain $Ω^+$. Three purple points next to interface $Γ$ are called corner points and the other two black nodes are regular points. 27

4.1 Implementation illustration based on finite difference in three dimensional space. The surface $Γ$ is a sphere, which separate the two subdomains. In three dimensional space, a straight line cuts through the entire domain $Ω$ at each grid point in each direction, which intersects the surface at some arbitrary node. 38

4.2 Temporal convergence for ellipsoid surface. 40

4.3 Color maps for ellipsoid surface. (a). numerical solution; (b). numerical error. 41

4.4 Temporal convergence for cylinder surface with $α^- = 1$ and $α^+ = 10$. 42

4.5 Temporal convergence for cylinder surface with $α^- = 10$ and $α^+ = 1$. 43

4.6 Temporal convergence for cylinder surface with $α^- = 1$ and $α^+ = 100$. 44

4.7 Color maps for molecular surface. (a). numerical solution; (b). numerical error. 44
4.8 Temporal convergence for molecular surface. ........................................ 45
4.9 Slice plots of the numerical solutions on $z = 0$ plane in Example 3. (a) $t = 0.5$; (b) $t = 1$; (c) $t = 1.5$; (d) $t = 2$. ........................................ 46
4.10 Contour plots of the numerical solutions on $z = 0$ plane in Example 3. (a) $t = 0.5$; (b) $t = 1$; (c) $t = 1.5$; (d) $t = 2$. ........................................ 47
4.11 Temporal convergence for torus surface. ........................................ 48
4.12 Color maps for torus surface. (a). numerical solution; (b). numerical error. .... 49
4.13 Temporal convergence for tanglecube surface. ................................ 50
4.14 Color maps for tanglecube surface. (a). numerical solution; (b). numerical error. .. 51
4.15 CPU time and memory usage of tanglecube (a). CPU time; (b). memory usage. .. 52
4.16 Color maps for heart surface. (a). numerical solution; (b). numerical error. .... 53
4.17 Temporal convergence for heart surface. ........................................ 54
4.18 CPU time and memory usage of heart (a). CPU time; (b). memory usage. .... 54
4.19 Stability test for the heart surface in Example 6. ................................. 55
4.20 Sphere efficiency comparison with rational type variable coefficient. .......... 56
4.21 Temporal $L_{\infty}$ error for sphere interface with rational type variable coefficient. . 57
4.22 Temporal $L_2$ error for sphere interface with rational type variable coefficient. . 57
4.23 Spatial $L_{\infty}$ error for sphere interface with rational type variable coefficient. .... 58
4.24 Spatial $L_2$ error for sphere interface with rational type variable coefficient. .... 58
4.25 Sphere efficiency comparison with trigonometric type variable coefficient. .... 60
4.26 Temporal $L_{\infty}$ error for sphere interface with trigonometric type variable coefficient. 60
4.27 Temporal $L_2$ error for sphere interface with trigonometric type variable coefficient. 61
4.28 Spatial $L_{\infty}$ error for sphere interface with trigonometric type variable coefficient. 61
4.29 Spatial $L_2$ error for sphere interface with trigonometric type variable coefficient. 62
4.30 Sphere efficiency comparison with Gaussian type variable coefficient. ......... 62
4.31 Temporal $L_{\infty}$ error for sphere interface with Gaussian type variable coefficient. 63
4.32 Temporal $L_2$ error for sphere interface with Gaussian type variable coefficient. 63
4.33 Spatial $L_\infty$ error for sphere interface with Gaussian type variable coefficient. 64
4.34 Spatial $L_2$ error for sphere interface with Gaussian type variable coefficient. 64
4.35 Tanglecube efficiency comparison with rational type variable coefficient. 65
4.36 Temporal $L_\infty$ error for tanglecube interface with rational type variable coefficient. 65
4.37 Temporal $L_2$ error for tanglecube interface with rational type variable coefficient. 66
4.38 Spatial $L_\infty$ error for tanglecube interface with rational type variable coefficient. 66
4.39 Spatial $L_2$ error for tanglecube interface with rational type variable coefficient. 67
4.40 Tanglecube efficiency comparison with trigonometric type variable coefficient. 68
4.41 Temporal $L_\infty$ error for tanglecube interface with trigonometric type variable coefficient. 68
4.42 Temporal $L_2$ error for tanglecube interface with trigonometric type variable coefficient. 69
4.43 Spatial $L_\infty$ error for tanglecube interface with trigonometric type variable coefficient. 69
4.44 Spatial $L_2$ error for tanglecube interface with trigonometric type variable coefficient. 70
4.45 Tanglecube efficiency comparison with Gaussian type variable coefficient. 70
4.46 Temporal $L_\infty$ error for tanglecube interface with Gaussian type variable coefficient. 71
4.47 Temporal $L_2$ error for tanglecube interface with Gaussian type variable coefficient. 71
4.48 Spatial $L_\infty$ error for tanglecube interface with Gaussian type variable coefficient. 72
4.49 Spatial $L_2$ error for tanglecube interface with Gaussian type variable coefficient. 72
CHAPTER 1
INTRODUCTION

Parabolic interface problems are a large class of problems modeling the change of a physical or biological quantity across a material interface. Numerous prize-winning models, such as the Pennes model for magnetic fluid hyperthermia therapy of human cancers [47], the time-dependent Poisson-Boltzmann equation for simulating electrostatics on biomolecules immersed in water phase [21, 67], and the cable equation for studying action potential propagating in active human cardiac tissues [29], can all be categorized to this class of interface problems. A typical parabolic interface problem involves a parabolic partial differential equation (PDE) for a function $u$

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha \nabla u) + f, \text{ in } \Omega \subset \mathbb{R}^3,$$

with a proper boundary condition prescribed for $u$ on the boundary $\partial \Omega$. The finite domain $\Omega$ is assumed to be split by a closed interface, $\Gamma = \Omega^+ \cap \Omega^-$, into two subdomains, $\Omega = \Omega^+ \cup \Omega^-$. Here $u(\vec{x}, t)$ is a 3D function of interest with $\vec{x} = (x, y, z)$. The diffusion coefficient $\alpha$ is discontinuous across the interface $\Gamma$, and $f(\vec{x}, t)$ is the source which may be even singular across the interface. On the interface $\Gamma$, jump conditions

$$[u] := u^+ - u^- = \phi(\vec{x}, t) \text{ and } [\alpha u_n] := \alpha^+ \nabla u^+ \cdot \vec{n} - \alpha^- \nabla u^- \cdot \vec{n} = \psi(\vec{x}, t),$$

relate the solution $u$ on both sides of the interface $\Gamma$. In (1.2), $\vec{n}$ is the unit outer normal direction, and the superscripts, $-$ and $+$, denote the limiting value of a function from one side or the other of the interface. With the nonhomogeneous terms $\phi$ and $\psi$ being spatially and temporally dependent, (1.2) takes a quite general form, while for many relatively simple applications, $\phi = \psi = 0.$
The exact solutions of (1.1) - (1.2) are not available, especially when the interface $\Gamma$ is irregularly shaped, while the numerical solutions of parabolic interface problems are highly nontrivial in 3D, with two major bottlenecks being the accuracy and efficiency. The key to a successful numerical method for solving the interface problems is to explicitly incorporating the jump conditions (1.2) in the numerical formulations to solve eqn. (1.1). This philosophy has been adopted by a number of numerical methods developed in the past decades for solving the interface problem. A list of such numerical schemes can be found in [35]. In terms of accuracy, the standard numerical treatments are known to perform poorly for delivering accurate approximations, or even fail to converge, if the jump conditions (1.2) are not treated appropriately in the numerical formulation. This calls for specific treatments to incorporate the jump conditions into the numerical formulation. By using body-fitted grids, the enforcement of jump conditions has been studied in many finite element and finite volume methods [5, 11, 51, 52, 56]. By using a Cartesian grid, jump conditions are more difficult to be satisfied because the interface cuts the grid lines arbitrarily and the normal direction is usually not along the Cartesian directions. The immersed interface method (IIM) [33] is known to be one of the most successful Cartesian grid methods for general interface problems. It achieves the second order of accuracy by rigorous imposing jump conditions in finite difference discretization via local Taylor expansions. Several successful IIM methods have been developed for parabolic problems [2, 7, 8, 30, 31].

The efficiency is especially concerned for solving the parabolic PDE (1.1) in steady state simulations or applications with long time evolution. In such problems, an implicit time stepping is indispensable so that a larger time increment $\Delta t$ or fewer time steps can be employed to save the CPU time. Nevertheless, a linear system has to be solved in every time step in an implicit integration. For 3D problems to be studied in this paper, the computational complexity in solving linear systems is more crucial than that of lower-dimensional problems, because a large spatial degree of freedom $N$ is usually attained for a not dense mesh. By using a generic iterative solver, a typical complexity in solving the sparse systems of various interface schemes is of the order $N^2$, i.e., $O(N^2)$. The acceleration could be achieved with a proper preconditioner or a geometric
multigrid method [26]. Nevertheless, due to special interface treatments, the formulation of the restriction and prolongation in a multigrid cycle is far away from trivial for interface problems. A major achievement in this area is the multigrid IIM method developed by Adams and Li [2], which scales as $O(N)$ computationally.

However, in all iterative methods, the iterations become inevitably longer for algebraic systems with an extremely large size or a very high condition number, due to the nature of iterative algorithms. For such challenging problems, an exact algebraic algorithm becomes more desirable, because the exact solver guarantees to stop with a fixed number of steps for a given $N$. The alternating direction implicit (ADI) methods [15, 46], which solve a multi-dimensional parabolic PDE by converting a multidimensional system to multiple sets of independent one-dimensional (1D) systems, can be regarded as exact solvers from the algebra point of view. Furthermore, for Cartesian grid finite difference methods, the 1D ADI systems are tridiagonal and can be efficiently solved by using the Thomas algorithm [53]. As a consequence, the ADI finite difference method is as efficient as iterative multigrid methods with a complexity $O(N)$, while being an exact algebraic solver. However, without an interface treatment, the central difference approximation is subject to an accuracy reduction in solving parabolic interface problems [21, 67], while the multi-dimensional Cartesian grid interface treatments [2, 7, 8, 30, 31] may not be realized in a 1D manner.

This motivates the development of several interface schemes [35, 38, 39, 41, 42, 68] which aim to maintain the efficiency and stability of the ADI methods, while restoring the accuracy to the second order near the interface. The first rigorous interface method is the IIM-ADI method introduced by Li and Mayo [38], in which the jump conditions (1.2) take a simpler form with $\alpha = 1$. The IIM-ADI method enforces the nonzero function and derivative jumps by introducing correction terms on the right hand sides of the 1D linear systems, without altering the central difference operators. Thus, the dimension decomposition can be simply conducted and the resulting IIM-ADI method has been applied to several problems [41, 42]. For physical jump conditions involving material coefficient $\alpha$ inside the jumps, the construction of the IIM-ADI
scheme through correction terms is intractable [39].

To treat the general jump conditions (1.2) with \( \alpha \) being a piecewisely-defined constant, a completely different strategy has been introduced in the matched ADI method for solving two-dimensional (2D) heat equations [35, 68], in which a tensor product decomposition of 2D jump conditions is carried out to generate 1D jump conditions along \( x \) and \( y \) directions. These 1D jump conditions can then be enforced via the matched interface and boundary (MIB) scheme [63, 64, 69, 77] in the ADI framework, and a fast algebraic algorithm has been developed to solve the resulted perturbed tridiagonal linear system with a complexity of \( O(N) \). Both Douglas [68] and Peaceman-Rachford [35] ADI schemes have been studied, and a spatial second order of accuracy is numerically confirmed.

The goal of this work is to develop a matched ADI method for solving 3D parabolic interface problems with variable diffusion coefficient, and expand the same idea to other time splitting methods like Trapezoidal splitting (TS) method, locally one-dimensional implicit Euler (LOD-IE) method and locally one-dimensionally Crank-Nicolson (LOD-CN) method. However, such a development faces many new challenges, which requires novel numerical schemes for handling much more complicated interfaces and jump conditions. In particular, the generalization of the tensor product decomposition of jump conditions from 2D is 3D is highly non-trivial, because the interface is now a surface, which is geometrically more complicated. Moreover, in the existing matched ADI schemes [35, 68], the MIB corrections depend on function values at different time levels so that the spatial approximation is affected by the temporal error. When generating to 3D, a ADI design with less perturbation errors becomes a pressing issue in the presence of such spatial-temporal error interference. To overcome these difficulties, several major improvement will be conducted in this paper in developing a 3D matched ADI method: (i) In contrast to the 2D schemes [35, 68], one grid line is allowed to intersect the interface more than twice in the proposed 3D scheme. This effort is obviously necessary for handling complex interfaces with fast changing curvature. (ii) The proposed 3D method estimates the tangential derivative from both sides of the interface, based on the local curvature of the interface, so that the
numerical estimation accuracy is well maintained. (iii) More importantly, a non-orthogonal local coordinate system is proposed to decompose the 3D interface jump conditions, so that all tangential derivatives can be approximated in a 2D manner. If an orthogonal local coordinate is employed as in the usual 3D MIB scheme [63], tangential derivative approximations are of 3D nature and become far more complicated. (vi) Finally, appropriate treatments are employed to reduce the computational cost and to maintain the convergence rate and stability.

The work in this direction originates from Zhao’s novel work of a matched alternating direction implicit (ADI) method for solving two dimensional (2D) parabolic interface problems [68]. In Zhao’s method, a series of numerical techniques are elaborately put together, resulting in an effective method, with great potentials for solving most complex interface problems, such as those with most general time-and-space dependent jump conditions imposed on very irregularly-shaped interfaces. Some of the techniques are, for instance, utilizing the Douglas Alternating Direction Implicit (ADI) method [15, 16] so that large time steps can be adopted to obtain equilibrium solutions in a fast pace, introducing a novel matched interface and boundary (MIB) spatial treatment [69, 77] to explicitly impose the jump conditions (1.1) in the finite difference formulas for approximating the spatial derivatives [68], and solving perturbed tri-diagonal linear systems by the combination of the Gaussian Elimination procedure and the Thomas algorithm [53].

Inspired by the success of Zhao’s work, following-up work has been conducted to continue the development of the matched ADI method: an improved MIB scheme with the Peaceman-Rachford ADI scheme [46] is developed for solving 3D parabolic interface problems with higher temporal convergence rate [35], a second order dispersive finite-difference time-domain (FDTD) algorithm is developed for transverse electric Maxwells equations with complex interfaces [45], a group of unconditionally stable time splitting methods is developed for the electrostatic analysis of solvated biomolecules [59], and a spatially second order ADI method is developed for solving three-dimensional (3D) parabolic interface problems [58].

In the process of reviewing fellow researchers’ recent work in this field, a previously
repored method [40, 43], called the Ghost Fluid method (GFM), for solving the Poisson’s equation on irregular domains, caught our attention. One is the popular sharp interface scheme ghost fluid method (GFM) originally developed for treating contact discontinuities in the inviscid Euler equation by Osher and his coworkers [18]. The method is typically first-order accurate for interface problem and the jump conditions are captured implicitly by extending values across the interface into a ghost fluid in the favor of the level set method. By neglecting the tangential derivatives, the GFM is simple and robust which can be applied to complex interface problems with confidence such as 3D moving interface. Comparing the GFM method to the MIB method involved in Zhao’s work, we realized that they share a great deal of commons. Moreover, an interesting observation is that a simple variation of the MIB method can be viewed as an improvement of the GFM, in the sense that an impractical assumption in the GFM is substituted with rigorous mathematical formulations when imposing the jump conditions on the interface. To distinguish the two MIB scheme, the original MIB method is called MIBV2 scheme, the variation MIB method is called MIBV1 scheme.
CHAPTER 2
THREE DIMENSIONAL MATCHED ADI METHOD WITH PIECE-WISE CONSTANT DIFFUSION COEFFICIENTS

Consider an arbitrarily shaped dielectric interface $\Gamma$ enclosed within a cubic domain $\Omega \subset \mathbb{R}^3$. The interface $\Gamma$ is analytically determined as the zero level set of a given function $S(x, y, z)$, i.e., $S(x, y, z) = 0$. We assume a uniform grid partition with mesh size, $\Delta x = h_x$, $\Delta y = h_y$ and $\Delta z = h_z$, resulting in $N_x$, $N_y$, and $N_z$ grids in $x$, $y$, and $z$ direction, respectively. Being aware of the complex shape of the 3D interface $\Gamma$, the grid line is allowed to cross the interface $\Gamma$ more than twice. Moreover, the temporal domain is partitioned with uniform time increment $\Delta t$, and the notation $u_{i,j,k}^m = u(x_i, y_j, z_k, t_m)$ is utilized to denote the value of unknown function on grid $(x_i, y_j, z_k)$ at time step $t_m$.

2.1 ADI semi-discretization

A Douglas-Rachford ADI method [15, 46] will be constructed for solving the parabolic PDE (1.1). For a piecewise constant $\alpha$, we first rewrite (1.1) as

$$\frac{1}{\alpha} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x^2} + \frac{\partial^2 u}{\partial^2 y^2} + \frac{\partial^2 u}{\partial^2 z^2} + \frac{f}{\alpha}, \text{ in } \Omega^- \cup \Omega^+, \quad (2.1)$$

which allows an easier numerical formulation for temporal discretization, while being equivalent to (1.1), subject to the same jump conditions (1.2). Applying the first order Douglas-Rachford
ADI method on equation (2.1) yields

\[
\begin{align*}
\frac{1}{\alpha} - \Delta t \frac{\partial^2}{\partial x^2} u_{i,j,k}^* &= \left( \frac{1}{\alpha} + \Delta t \left( \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) \right) u_{i,j,k}^m + \frac{\Delta t f_{i,j,k}^{m+1}}{\alpha}, \\
\frac{1}{\alpha} - \Delta t \frac{\partial^2}{\partial y^2} u_{i,j,k}^* &= \frac{1}{\alpha} u_{i,j,k}^m - \Delta t \frac{\partial^2}{\partial y^2} u_{i,j,k}^m, \\
\frac{1}{\alpha} - \Delta t \frac{\partial^2}{\partial z^2} u_{i,j,k}^{m+1} &= \frac{1}{\alpha} u_{i,j,k}^m - \Delta t \frac{\partial^2}{\partial z^2} u_{i,j,k}^m,
\end{align*}
\] (2.2)

where \( u^* \) and \( u^{**} \) are two intermediate values. Noticing that the ADI formula (2.2) is derived from the implicit Euler method

\[
\frac{u_{i,j,k}^{m+1} - u_{i,j,k}^m}{\alpha \Delta t} = \frac{\partial^2}{\partial x^2} u_{i,j,k}^m + \frac{\partial^2}{\partial y^2} u_{i,j,k}^m + \frac{\partial^2}{\partial z^2} u_{i,j,k}^m + \frac{f_{i,j,k}^{m+1}}{\alpha}.
\] (2.3)

To see this, one can eliminate \( u^* \) and \( u^{**} \) in (2.2), which gives rise to

\[
\begin{align*}
\left( \frac{1}{\alpha} - \Delta t \frac{\partial^2}{\partial x^2} \right) \left( \frac{1}{\alpha} - \Delta t \frac{\partial^2}{\partial y^2} \right) \left( \frac{1}{\alpha} - \Delta t \frac{\partial^2}{\partial z^2} \right) u_{i,j,k}^{m+1} &= \frac{1}{\alpha^3} u_{i,j,k}^m - \Delta t^3 \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial y^2} \frac{\partial^2}{\partial z^2} u_{i,j,k}^m
+ \frac{\Delta t^2}{\alpha} \left( \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial y^2} \frac{\partial^2}{\partial z^2} \right) u_{i,j,k}^m + \frac{\Delta t f_{i,j,k}^{m+1}}{\alpha^3}, \quad (2.4)
\end{align*}
\]

Reorganizing the terms in (2.4), we have

\[
\begin{align*}
\frac{u_{i,j,k}^{m+1} - u_{i,j,k}^m}{\alpha \Delta t} - \alpha \Delta t \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) u_{i,j,k}^m &= u_{i,j,k}^m + \Delta t f_{i,j,k}^{m+1} - \alpha^2 \Delta t \left( \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial y^2} \frac{\partial^2}{\partial z^2} \right) (u_{i,j,k}^m - u_{i,j,k}^m)
+ \alpha^3 \Delta t^3 \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial y^2} \frac{\partial^2}{\partial z^2} (u_{i,j,k}^{m+1} - u_{i,j,k}^m).
\end{align*}
\] (2.5)

Since the difference \((u_{i,j,k}^{m+1} - u_{i,j,k}^m)\) is on the order of \( \Delta t \), last two terms on the right hand side of (2.5) are higher order perturbation terms. After dropping them, (2.5) is simply (2.3). Thus, the temporal order of the Douglas-Rachford ADI method (2.2) is one, which is the same as the implicit Euler scheme (2.3). Computationally, the Douglas-Rachford ADI scheme is superior to the implicit Euler scheme for the efficiency consideration, because one only needs to solve 1D problems in (2.2), which leads to a faster and easier-to-implement numerical scheme.
2.2 Spatial discretization of the 3D matched ADI method

The spatial derivatives in (2.2) or (2.3) will be approximated by the finite difference operators \( \delta_{xx} \), \( \delta_{yy} \) and \( \delta_{zz} \), which must be treated with great caution due to the presence of the interface \( \Gamma \). Standard central difference formula

\[
\frac{\partial^2}{\partial x^2} u_{i,j,k}^{m+1} \approx \delta_{xx} u_{i,j,k}^{m+1} := \frac{1}{\Delta x^2} (u_{i-1,j,k}^{m+1} - 2u_{i,j,k}^{m+1} + u_{i+1,j,k}^{m+1}),
\]

\[
\frac{\partial^2}{\partial y^2} u_{i,j,k}^{m+1} \approx \delta_{yy} u_{i,j,k}^{m+1} := \frac{1}{\Delta y^2} (u_{i,j-1,k}^{m+1} - 2u_{i,j,k}^{m+1} + u_{i,j+1,k}^{m+1}),
\]

\[
\frac{\partial^2}{\partial z^2} u_{i,j,k}^{m+1} \approx \delta_{zz} u_{i,j,k}^{m+1} := \frac{1}{\Delta z^2} (u_{i,j,k-1}^{m+1} - 2u_{i,j,k}^{m+1} + u_{i,j,k+1}^{m+1}),
\]

(2.6)
can be adopted at the nodes which are far away from \( \Gamma \) (they are called regular nodes), while finite difference formula (2.6) must be modified in order to rigorously impose the jump conditions (1.2) into the numerical formulation at the nodes close to the interface \( \Gamma \) (they are called irregular nodes). The matched ADI method corrects \( \delta_{xx} \), \( \delta_{yy} \) and \( \delta_{zz} \) in two major stages [35, 68]. First, a tensor product decomposition is conducted to transfer 3D jump conditions (1.2) into 1D ones along Cartesian directions, i.e., \([\alpha u_x] = \psi_x\), \([\alpha u_y] = \psi_y\), and \([\alpha u_z] = \psi_z\), where we denote \( u_x = \frac{\partial u}{\partial x} \).

Second, one enforces the 1D jump conditions in each alternating direction of the ADI algorithm via the matched interface and boundary (MIB) scheme [63, 64, 69, 77].

2.2.1 Local coordinate and non-orthogonal tangential directions

A tensor product decomposition of jump conditions will be conducted at every necessary interface point. Denote \((x_\Gamma, y_\Gamma, z_\Gamma)\) as a general intersection point between the interface \( \Gamma \) and one Cartesian grid line. Note that one Cartesian grid line has two fixed coordinate values. For instance, consider a \( y \) grid line with \( x = x_i \) and \( z = z_k \), where \( x_i \) and \( z_k \) are multiples of \( \Delta x \) and \( \Delta z \).

The interface point is then \((x_i, y_\Gamma, z_k)\), where \( y_\Gamma \) is assumed to be an arbitrary value satisfying \( y_{j-1} \leq y_\Gamma \leq y_j \). At this particular interface point, we need to generate a new 1D jump condition \([\alpha u_y] = \psi_y\). Similarly, at every interface point \((x_\Gamma, y_\Gamma, z_\Gamma)\), one needs to construct a new 1D jump condition. This procedure can be described using \( y \)-direction as an example in the following, and
the treatments in the other two directions are similar.

A local coordinate is introduced first. At the interface point IPY\((x_i, y_\Gamma, z_k)\), the outward normal direction \(\vec{n} = (n_x, n_y, n_z)\) can be determined by using the level set function \(S(x, y, z)\). Denote a local coordinate as \((\xi, \eta, \zeta)\) with \(\xi\) varying along \(\vec{n}\) and \(\eta\) and \(\zeta\) being two tangential directions. Non-orthogonal tangential directions will be constructed in the local coordinate so that approximations to the tangential derivatives can be conducted in a 2D Cartesian grid plane, instead of a 3D space, in the tensor product decomposition. This new idea considerably simplifies the proposed algorithm.

The tangential plane through the interface point IPY\((x_i, y_\Gamma, z_k)\) can be represented as

\[
n_x(x - x_i) + n_y(y - y_\Gamma) + n_z(z - z_k) = 0.
\]

(2.7)

We formulate \(\eta\) and \(\zeta\) by intersecting this tangential plane with the place \(x = x_i\) and \(z = z_k\), respectively, (see Fig. 2.1)

\[
\begin{align*}
n_yy + n_zz &= n_yy_\Gamma + n_zz_k, \quad \text{and} \quad x = x_i, \\
n_xx + n_yy &= n_xx_i + n_yy_\Gamma, \quad \text{and} \quad z = z_k.
\end{align*}
\]

(2.8, 2.9)

After choosing a positive direction appropriately, we can define two unit directions on these two intersection lines by

\[
\eta = (0, \frac{n_z}{\sqrt{n_y^2 + n_z^2}}, -\frac{n_y}{\sqrt{n_y^2 + n_z^2}}) \quad \text{and} \quad \zeta = (\frac{n_y}{\sqrt{n_x^2 + n_y^2}}, -\frac{n_x}{\sqrt{n_x^2 + n_y^2}}, 0),
\]

as shown in Figure 2.1. The local coordinate system \((\xi, \eta, \zeta)\) at the interface point IPY can be
related to the global $xyz$ coordinate system by a transformation matrix $P$

\[
\begin{bmatrix}
\xi \\
\eta \\
\zeta
\end{bmatrix} = P
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}.
\] (2.10)

In the $y$–direction, the transformation matrix $P_y$ is provided by

\[
P_y = \begin{bmatrix}
n_x & n_y & n_z \\
0 & -\frac{n_y}{\sqrt{n_x^2 + n_z^2}} & \frac{n_z}{\sqrt{n_x^2 + n_z^2}} \\
\frac{n_z}{\sqrt{n_x^2 + n_y^2}} & \frac{n_x}{\sqrt{n_x^2 + n_y^2}} & 0
\end{bmatrix}.
\] (2.11)
Similarly, one can construct the transformation matrix $P_x$ in the $x$–direction

$$P_x = \begin{bmatrix}
  n_x & n_y & n_z \\
  \frac{n_y}{\sqrt{n_x^2+n_z^2}} & 0 & -\frac{n_z}{\sqrt{n_x^2+n_z^2}} \\
  \frac{n_z}{\sqrt{n_x^2+n_y^2}} & -\frac{n_y}{\sqrt{n_x^2+n_y^2}} & 0 \end{bmatrix} , \quad (2.12)$$

and $P_z$ in the $z$–direction

$$P_z = \begin{bmatrix}
  n_x & n_y & n_z \\
  0 & \frac{n_z}{\sqrt{n_y^2+n_z^2}} & -\frac{n_y}{\sqrt{n_y^2+n_z^2}} \\
  \frac{n_y}{\sqrt{n_x^2+n_y^2}} & 0 & -\frac{n_z}{\sqrt{n_x^2+n_y^2}} \end{bmatrix} , \quad (2.13)$$

respectively, for interface point $(x, y, z)$ and $(x, y, z')$.

### 2.2.2 Tensor product decomposition of jump conditions

After the local $\xi\eta\zeta$–coordinate system at the interface point IPY is obtained, one can differentiate the zeroth order jump condition $[u] = \phi$ in (1.2) along the two tangential directions ($\eta$ and $\zeta$) to obtain two additional jump conditions

$$[u_\eta] = \phi_\eta, \quad \text{and} \quad [u_\zeta] = \phi_\zeta. \quad (2.14)$$

Note that the known jump conditions (1.2) and (2.14) are in normal and tangential directions, and can not be applied in a 1D manner in the ADI algorithm, because they are not along the $x$–, $y$– or $z$–axis.

The desired Cartesian jump conditions, i.e., $[\alpha u_x]$, $[\alpha u_y]$ and $[\alpha u_z]$, can be obtained through coordinate transformation,

$$\begin{pmatrix}
  \psi_x \\
  \psi_y \\
  \psi_z
\end{pmatrix} := \begin{pmatrix}
  [\alpha u_x] \\
  [\alpha u_y] \\
  [\alpha u_z]
\end{pmatrix} = P^{-1} \begin{pmatrix}
  [\alpha u_\xi] \\
  [\alpha u_\eta] \\
  [\alpha u_\zeta]
\end{pmatrix} , \quad (2.15)$$

where $[\alpha u_\xi] = \psi$ is known in (1.2), while $[\alpha u_\eta]$ and $[\alpha u_\zeta]$ are unknown. Notice that the transformation matrix $P$ is singular only when the tangential plane is parallel to one of $xy$–, $xz$–, etc.
or \( yz \)– plane. In such a case, the flux jump condition \([\alpha u_\xi] = \psi\) is simply the desired 1D jump condition in \( z, y \) or \( x \) direction. Besides this degenerate case, \( P^{-1} \) in (2.15) always exists and can be calculated either analytically or numerically.

The unknown jumps of \([\alpha u_\eta]\) and \([\alpha u_\zeta]\) can be calculated with the aid of (2.14)

\[
[\alpha u_\eta] = \alpha^+ \phi_\eta + (\alpha^+ - \alpha^-)u^-_\eta \quad \text{or} \quad [\alpha u_\eta] = \alpha^- \phi_\eta + (\alpha^+ - \alpha^-)u^+_\eta, 
\]

\[
[\alpha u_\zeta] = \alpha^+ \phi_\zeta + (\alpha^+ - \alpha^-)u^-_\zeta \quad \text{or} \quad [\alpha u_\zeta] = \alpha^- \phi_\zeta + (\alpha^+ - \alpha^-)u^+_\zeta. 
\]

Computationally, we will approximate two tangential derivatives \( u_\eta \) and \( u_\zeta \), in either + or − side, for determining \([\alpha u_\eta]\) and \([\alpha u_\zeta]\) numerically. Once \([\alpha u_\eta]\) and \([\alpha u_\zeta]\) are obtained, they are coupled with \([\alpha u_\xi] = \psi\) to deliver 1D jump conditions in Cartesian directions: \([\alpha u_x] = \psi_x\), \([\alpha u_y] = \psi_y\), and \([\alpha u_z] = \psi_z\). Thus, the key in the tensor product decomposition of 3D jump conditions is how to approximate two tangential derivatives accurately.

### 2.2.3 Tangential derivative approximations

A new algorithm is proposed in this paper to approximate tangential derivatives, which is novel in two folds. First, since each of the non-orthogonal tangential directions is contained in a 2D Cartesian grid plane, the tangential derivatives can be computed in a 2D manner. If the orthogonal tangential directions are adopted as in the classical MIB 3D scheme [63], one tangential derivative has to be calculated as a 3D problem. Obviously, the present algorithm is simpler. Second, both positive and negative side information will be employed for calculating tangential derivatives, while in our previous 2D matched ADI methods [35, 68], the tangential derivatives are calculated only from the positive side. In 3D, the interface \( \Gamma \) becomes geometrically more complicated, so that a more robust algorithm has to be developed in this work.

We illustrate the proposed algorithm by considering \( u_\zeta \) at the interface point \((x_i, y_\Gamma, z_k)\), and \( u_\eta \) can be treated similarly. Since the \( \zeta \) direction is contained in a plane \( z = z_k \), all involved points are in this plane. Thus, the numerical scheme can be shown in a \( x\)-\( y \) coordinate, see Fig. 2.2(a). Here we assume \( y_{j-1} \leq y_\Gamma \leq y_j \) and denote the interface point (and other points) simply as \((x_i, y_\Gamma)\)
by dropping \( z = z_k \). If \( \Gamma \) is locally convex with respect to \( \Omega^- \), the approximation of \( u^+_\zeta \) by nearby on-grid values from the outer domain \( \Omega^+ \) produces a better precision, while for a local concave curve, it is better to approximate \( u^-_\zeta \) from \( \Omega^- \). For a situation shown in Fig. 2.2(a) where the interface \( \Gamma \) has an inflection, an optimization procedure is designed to select a better approximation from either + or − side. In fact, we apply the same optimization procedure for all interface points so that one can automatically select \( u^+_\zeta \) or \( u^-_\zeta \) based on the local geometry.

In our algorithm, the auxiliary and supporting nodes for approximating \( u^+_\zeta \) and \( u^-_\zeta \) are first determined. As shown in Fig. 2.2(a), the interface point \((x_i, y_\Gamma)\) is marked with a purple square. Two auxiliary points \((x_{i-1}, y_{\Gamma-1})\) and \((x_{i+1}, y_{\Gamma+1})\) (purple rectangles in Figure 2.2(a)) along the \( \zeta \) line are chosen to form a central difference approximation for the \( u^\pm_\zeta \)

\[ u^\pm_\zeta(x_i, y_\Gamma) \approx \frac{u^\pm(x_{i+1}, y_{\Gamma+1}) - u^\pm(x_{i-1}, y_{\Gamma-1})}{2\Delta \zeta} , \quad \text{(2.18)} \]

where \( \Delta \zeta \) is the distance between \((x_i, y_\Gamma)\) and \((x_{i-1}, y_{\Gamma-1})\). The auxiliary values will be interpolated/extrapolated by using supporting nodal values from the same side. Referring to Figure 2.2(a), \( u^+(x_{i+1}, y_{\Gamma+1}) \) and \( u^+(x_{i-1}, y_{\Gamma-1}) \) will be extrapolated by six supporting nodes below \( \Gamma \) (blue circles), while \( u^-(x_{i+1}, y_{\Gamma+1}) \) and \( u^-(x_{i-1}, y_{\Gamma-1}) \) are approximated by six supporting nodes above \( \Gamma \) (orange circles). The selection of \( u^+_\zeta \) or \( u^-_\zeta \) is according to the total distance between supporting nodes and auxiliary nodes. For Fig. 2.2(a), we define

\[
d^+ = \sum_{J=2}^{4} |y_{\Gamma+1} - y_{j-J}| + \sum_{J=0}^{2} |y_{\Gamma-1} - y_{j-J}|, \quad \text{(2.19)}
\]

\[
d^- = \sum_{J=-1}^{1} |y_{\Gamma+1} - y_{j+J}| + \sum_{J=1}^{3} |y_{\Gamma-1} - y_{j+J}|. \quad \text{(2.20)}
\]

If \( d^+ > d^- \), we calculate \( u^-_\zeta \) by using (2.18) from the negative side. Otherwise, we approximate \( u^-_\zeta \) by using (2.18) from the positive side. Then the auxiliary values are extrapolated by six supporting values.

The time interference is a subtle issue in this process. Note that at this stage of
Figure 2.2: Tangential derivative approximation and MIB interface treatment. (a). The tangential derivative at the interface point (purple square) is approximated through two auxiliary points (purple rectangles). Each auxiliary value will be extrapolated by three supporting nodes from the same side (positive or negative). An optimization procedure is designed to automatically select either positive or negative side for approximating tangential derivatives. The MIB 1D jump condition enforcement is also illustrated, which involves four nodes (black filled circles) and two fictitious nodes (red open circles). (b). In a corner MIB treatment of 1D jump conditions, five nodes (black filled circles) and three fictitious nodes (red open circles) are involved. Here two interface points \((x_i, y_{\Gamma 1}, z_k)\) and \((x_i, y_{\Gamma 2}, z_k)\) are the intersection points of \(\Gamma\) with two dashed lines.

approximation, function values at the future time \(t = t_{m+1}\) are unknown. So, we can only use the known function values at the current time step \(t = t_m\) for the six supporting nodes. Consequently, the calculated tangential derivatives are held at \(t_m\), and we denote them as \((u^-_\zeta)^m\) and \((u^+_\zeta)^m\).

However, the finite difference correction (2.6) is intended to be conducted at \(t_{m+1}\). Hence, in (2.15), (2.16), and (2.17), we specify analytically known jumps at \(t_{m+1}\), i.e., \(\psi_{m+1}, \phi_{\eta}^{m+1}, \) and \(\phi_{\zeta}^{m+1}\), for a better accuracy. Mathematically, the calculated jumps \(\psi_x, \psi_y,\) and \(\psi_\zeta\) can be expressed as a linear combination of some supporting nodal values at \(t_m\), and \(\psi_{m+1}, \phi_{\eta}^{m+1}, \) and \(\phi_{\zeta}^{m+1}\) at \(t_{m+1}\). With hybrid contributions from two time levels, the Cartesian jump values \(\psi_x, \psi_y,\) and \(\phi_\zeta\) mainly
depend on the supporting function values at \( t_m \). Thus, we will denote them as \( \psi^m_x \), \( \psi^m_y \), and \( \psi^m_\zeta \).

This finishes the first stage of the spatial discretization of the matched ADI method.

### 2.2.4 1D jump conditions enforcement

In the second stage of the matched ADI method, we impose the decomposed 1D jump conditions

\[
[u] = \phi^{m+1}, \quad [au_x] = \psi^m_x, \quad [au_y] = \psi^m_y, \quad [au_z] = \psi^m_\zeta, \quad (2.21)
\]

to correct the finite difference operators \( \delta_x \), \( \delta_y \) and \( \delta_z \) in (2.6) by using the MIB scheme [63, 64, 69, 77]. We will also only consider \( \delta_y \) here for simplicity. For two irregular points, \( \delta_y \) at time step \( t_{m+1} \) is modified to be

\[
\delta_y u_{i,j-1,k}^{m+1} := \frac{1}{\Delta y^2}(\tilde{u}_{i,j-1,k}^{m+1} - 2u_{i,j,k}^{m+1} + u_{i,j+1,k}^{m+1}), \quad \text{in } \Omega^-
\]
\[
\delta_y u_{i,j,k}^{m+1} := \frac{1}{\Delta y^2}(u_{i,j-2,k}^{m+1} - 2u_{i,j-1,k}^{m+1} + \tilde{u}_{i,j,k}^{m+1}), \quad \text{in } \Omega^+ \quad (2.22)
\]

where \( \tilde{u}_{i,j-1,k}^{m+1} \) and \( \tilde{u}_{i,j,k}^{m+1} \) are two fictitious value at node \((x_i, y_{j-1}, z_k)\) and \((x_i, y_j, z_k)\), respectively, as shown in Fig. 2.2(a). These two fictitious values \( \tilde{u}_{i,j-1,k}^{m+1} \) and \( \tilde{u}_{i,j,k}^{m+1} \) are determined by discretizing two jump conditions in (2.21).

\[
\omega^+_{0,1} u_{i,j-2,k}^{m+1} + \omega^+_{0,2} u_{i,j-1,k}^{m+1} + \omega^+_{0,3} \tilde{u}_{i,j,k}^{m+1} = \omega^-_{0,1} \tilde{u}_{i,j-1,k}^{m+1} + \omega^-_{0,2} u_{i,j,k}^{m+1} + \omega^-_{0,3} \tilde{u}_{i,j+1,k}^{m+1} + \phi^{m+1}, \quad (2.23)
\]
\[
\alpha^+ (\omega^+_{1,1} u_{i,j-2,k}^{m+1} + \omega^+_{1,2} u_{i,j-1,k}^{m+1} + \omega^+_{1,3} \tilde{u}_{i,j,k}^{m+1}) = \alpha^- (\omega^-_{1,1} \tilde{u}_{i,j-1,k}^{m+1} + \omega^-_{1,2} u_{i,j,k}^{m+1} + \omega^-_{1,3} \tilde{u}_{i,j+1,k}^{m+1}) + \psi^m_y, \quad (2.24)
\]

where \( \omega^+_{i,j} \) and \( \omega^-_{i,j} \) are finite different weights [19] for interface surrounding nodes from outer domain \( \Omega^+ \) or inner domain \( \Omega^- \). The subscript \( i \) represents the interpolation when \( i = 0 \) and first order approximation when \( i = 1 \), and the subscript \( j \) represents the indices of related grid nodes.

Solving (2.23) and (2.24) allows \( \tilde{u}_{i,j-1,k}^{m+1} \) and \( \tilde{u}_{i,j,k}^{m+1} \) to be written as linear combinations of \( u_{i,j-2,k}^{m+1} \), \( u_{i,j-1,k}^{m+1} \), \( u_{i,j,k}^{m+1} \), and two jump values \( \phi^{m+1} \) and \( \psi^m_y \). By substituting \( \tilde{u}_{i,j-1,k}^{m+1} \) and \( \tilde{u}_{i,j,k}^{m+1} \) into (2.22), \( \delta_y u_{i,j,k}^{m+1} \) and \( \delta_y u_{i,j-1,k}^{m+1} \) are represented by four surrounding on-grid values at \( t_{m+1} \), i.e., \( u_{i,j-2,k}^{m+1} \), \( u_{i,j-1,k}^{m+1} \), \( u_{i,j,k}^{m+1} \), four jump values at \( t_{m+1} \), i.e., \( \phi^{m+1} \), \( \psi^m_y \), \( \phi^{m+1} \), and \( \phi^{m+1} \), six values at nodes
on plane \( z = z_k \) at \( t_m \), and six values at nodes on plane \( x = x_i \) at \( t_m \).

It is possible that the interface \( \Gamma \) changes rapidly and cuts one grid line twice within a short distance. If there are at least two grid nodes in between these two intersection points, the above mentioned approximation can be conducted. When there is only one node (it is called a corner point), a MIB corner treatment has to be adopted. For instance, in Fig. 2.2(b), a corner treatment is needed for \((x_{i-1}, y_j, z_k)\) and \((x_i, y_j, z_k)\) along \( y \) direction, while the usual MIB scheme can be used for other irregular points in \( \Omega^- \). Since the present jump condition enforcement is of 1D nature, the corner treatment developed in our previous 2D matched ADI method [68] can be simply utilized for 3D problems. Essentially, we need to enforce jump conditions at two interface points simultaneously. We omit the details but briefly describe the idea here by considering \((x_i, y_j, z_k)\) in Fig. 2.2(b). At this corner point, three fictitious values, \( \tilde{u}_{i,j-1,k}^{m+1} \), \( \tilde{u}_{i,j,k}^{m+1} \) and \( \tilde{u}_{i,j+1,k}^{m+1} \), are calculated together with one additional fictitious value, either \( \tilde{u}_{i,j-2,k}^{m+1} \) or \( \tilde{u}_{i,j+2,k}^{m+1} \) depending on which node is closer, in order to maintain the convergence rate. The four fictitious values are solved by a system of four equations, built from imposing the jump conditions at the interface points on both sides of the corner point, following similar formulation as in (2.23)-(2.24). By substituting fictitious values into (2.22), \( \delta_{yy} u_{i,j,k}^{m+1} \) is represented by five surrounding on-grid values at \( t_{m+1} \), i.e., \( u_{i,j-2,k}^{m+1}, u_{i,j-1,k}^{m+1}, u_{i,j+1,k}^{m+1} \) and \( u_{i,j+2,k}^{m+1} \), eight jump values at \( t_{m+1} \), i.e., \( \Phi_{x}^{m+1}, \Phi_{y}^{m+1}, \Phi_{\eta}^{m+1}, \Phi_{\zeta}^{m+1} \), and \( \Phi_{\xi}^{m+1} \) at two interface points \((x_i, y_{T1}, z_k)\) and \((x_i, y_{T2}, z_k)\), and a total of 24 supporting nodal values at \( t_m \). This finishes the second stage of the spatial discretization of the matched ADI method. In this entire process, a spatially second order of accuracy is guaranteed throughout the domain.

Symbolically, the aforementioned MIB corrections of the finite difference operators \( \delta_{xx} \), \( \delta_{yy} \), and \( \delta_{zz} \) can be expressed as [35, 68]

\[
\begin{align*}
\delta_{xx} u_{i,j,k}^{m+1} &= D_{xx} u_{i,j,k}^{m+1} + B_x u_{i,j,k}^{m} + \Phi_x^{m+1}, \\
\delta_{yy} u_{i,j,k}^{m+1} &= D_{yy} u_{i,j,k}^{m+1} + B_y u_{i,j,k}^{m} + \Phi_y^{m+1}, \\
\delta_{zz} u_{i,j,k}^{m+1} &= D_{zz} u_{i,j,k}^{m+1} + B_z u_{i,j,k}^{m} + \Phi_z^{m+1}.
\end{align*}
\]

\( (2.25) \quad (2.26) \quad (2.27) \)
where $D_{xx}$, $D_{yy}$, and $D_{zz}$ are perturbed operators which have a band-width 4 for a usual irregular node, and a band-width 5 for a corner node. The operators $B_x$, $B_y$, and $B_z$ are due to tangential derivative approximations, which have a band-width 12 for a usual irregular node, and a band-width 24 for a corner node. The jump terms $\Phi^{m+1}_x$, $\Phi^{m+1}_y$, and $\Phi^{m+1}_z$ are the sums of 4 and 8 jump values, respectively, for a usual irregular node and corner node.

### 2.3 A new ADI method with less perturbation terms

Because the MIB corrected finite difference operators in (2.25) - (2.27) involves two time levels $t_m$ and $t_{m+1}$, a direct application of them to approximate derivatives in the standard ADI scheme (2.2) is questionable. In particular, the intermediate functions $u^\ast$ and $u^{\ast\ast}$ are at unknown fractional time instants between $t_m$ and $t_{m+1}$, so that one does not know how to specify a time for them. One possible choice here is to use $t_{m+1}$ for approximating derivatives of $u^\ast$ and $u^{\ast\ast}$ in (2.2), while derivatives of $u^m$ on the right hand sides of (2.2) are evaluated at $t_m$. As a consequence, when one eliminates the intermediate functions $u^\ast$ and $u^{\ast\ast}$, similar to (2.4) and (2.5) in the semi-discretized form, the difference between such an ADI scheme and the implicit Euler scheme involves many perturbation terms.

To overcome the aforementioned difficulty, we propose a new ADI method which minimizes the number perturbation terms when comparing it with the implicit Euler scheme. Our idea is to apply the finite difference operators (2.25) - (2.27) to the implicit Euler scheme (2.3), which then can be rewritten as

\[
\frac{u_{i,j,k}^{m+1} - u_{i,j,k}^{m}}{\alpha \Delta t} = D_{xx} u_{i,j,k}^{m+1} + D_{yy} u_{i,j,k}^{m+1} + D_{zz} u_{i,j,k}^{m+1} + F_{i,j,k}^{m+1},
\]

(2.28)

where $F_{i,j,k}^{m+1} = (B_x + B_y + B_z) u_{i,j,k}^{m+1} + \Phi^{m+1}_x + \Phi^{m+1}_y + \Phi^{m+1}_z + \frac{1}{\alpha} f_{i,j,k}^{m+1}$. The Douglas-Rachford ADI
method for the implicit Euler scheme (2.28) is simply given as

\begin{align*}
\frac{1}{\alpha} - \Delta t D_{xx} & \quad u_{i,j,k}^* = \left( \frac{1}{\alpha} + \Delta t (D_{yy} + D_{zz}) \right) u_{i,j,k}^m + \Delta t F_{i,j,k}^{m+1} , \\
\frac{1}{\alpha} - \Delta t D_{yy} & \quad u_{i,j,k}^{**} = \frac{1}{\alpha} u_{i,j,k}^* - \Delta t D_{yy} u_{i,j,k}^m , \\
\frac{1}{\alpha} - \Delta t D_{zz} & \quad u_{i,j,k}^{m+1} = \frac{1}{\alpha} u_{i,j,k}^{**} - \Delta t D_{zz} u_{i,j,k}^m ,
\end{align*}

(2.29)

Similarly, we can show the difference between such an ADI and the implicit Euler schemes is

\begin{align*}
-\alpha^2 \Delta t^2 (D_{xx} D_{yy} + D_{xx} D_{zz} + D_{yy} D_{zz}) (u_{i,j,k}^{m+1} - u_{i,j,k}^m) & + \alpha^3 \Delta t^3 D_{xx} D_{yy} D_{zz} (u_{i,j,k}^{m+1} - u_{i,j,k}^m) ,
\end{align*}

(2.30)

which is in consistent with the semi-discretized form. The temporal order of the proposed ADI scheme is obviously one, i.e., \( O(\Delta t) \).

In our computations, the boundary conditions for \( u^* \) and \( u^{**} \) in (2.29) are assumed to be the same as \( u^{m+1} \), i.e., at time step \( t = t_{m+1} \). We note that the matrix coefficients of \( D_{xx}, D_{yy}, D_{zz}, B_x, B_y, \) and \( B_z \) only require to be calculated once before the time evolution, because these spatial discretization coefficients only depend on interface geometry and grid nodes, which are time invariant. Then, these sparse coefficients and their column and row indexes are stored and reused in each time step, without too much computational overhead. Algebraically, the corrected operators \( D_{xx}, D_{yy}, \) and \( D_{zz} \) are perturbed tridiagonal matrices with only a few rows having a band-width four (usual irregular node) or five (corner node). A fast algorithm has been developed in [68] to invert such matrices efficiently at each time step, in which these systems are first converted to tridiagonal ones by row-eliminations and solved by the Thomas algorithm [53].

Taking the advantage of the Thomas algorithm, the proposed 3D matched ADI method has a computational cost of order \( O(N) \) for each time step, where \( N \) is the total spatial degree of freedom.

Neglecting the perturbation terms (2.30), the stability of the ADI scheme (2.29) is essentially governed by that of the implicit Euler scheme (2.28). Denote \( U^m \) as a vector of
dimension $N$ containing all $u$ values at the time $t_m$. One can rewrite the implicit Euler scheme (2.28) into the matrix form

$$DU^{m+1} = BU^m + C,$$  \hspace{1cm} (2.31)

where $D$ and $B$ are matrices of dimension $N \times N$, and $C$ is a vector of dimension $N$. Note that $B$ is due to the approximation of tangential derivatives by some function values of $u$ at time $t_m$. The sparse elements of $B$ are distributed in a rather random fashion - their locations depend on the interface geometry and grid size. Generally speaking, a von Neumann type stability analysis is infeasible. Alternatively, the stability analysis of the proposed ADI scheme can be conducted as in [68], through numerically verifying whether the spectral radius of the magnifying matrix $D^{-1}B$ is not greater than one. However, since $N$ is a very large number for 3D problems, the eigenvalue computation is time consuming, and is thus skipped in the present study. Instead, the stability of the 3D matched ADI scheme is numerically examined in next section.
CHAPTER 3
COMPARISON OF DIFFERENT TIME SPLITTING METHODS WITH VARIABLE DIFFUSION COEFFICIENTS

In the second part, unlike the governing equation in previous chapter, the diffusion coefficients $\alpha$ are now function of space. It’s necessary to rewrite the governing equation, a typical parabolic interface problem involves a parabolic partial differential equation (PDE) for a function $u(\vec{x})$ on both sides of the interface $\Gamma$.

$$\frac{\partial u(\vec{x})}{\partial t} = \nabla \cdot (\alpha(\vec{x})\nabla u(\vec{x})) + f(\vec{x}), \quad \vec{x} \in \Omega \setminus \Gamma, \quad \Omega = \Omega^- \cup \Omega^+ \quad (3.1)$$

with appropriate imposed boundary conditions along the interface,

$$[u](X) := u^+ - u^- = \phi(X, t), \quad [\alpha u_n](X) := \alpha^+(X)\nabla u^+ \cdot \vec{n}(X) - \alpha^-(X)\nabla u^- \cdot \vec{n}(X) = \psi(X, t), \quad \text{with } X \in \Gamma \quad (3.2)$$

where domain $\Omega$ is separated by interface $\Gamma$ into two sub-domains $\Omega^-$ and $\Omega^+$ and $\vec{n}(X)$ is the unit normal direction at a point $X$ on the interface pointing to $\Omega^+$ side. Meanwhile, we adopt $\vec{x}$ as a point to the domain without interface, while $X$ a point on the interface $\Gamma$. The following notations are chosen in this paper. The Cartesian coordinate system is taken into consideration with uniform meshes in a 3D finite rectangular domain $\Omega \subset \mathbb{R}^3$. Respectively, in $x$, $y$, and $z$ direction, the mesh sizes are chosen to be $\Delta x = h_x$, $\Delta y = h_y$ and $\Delta z = h_z$ with grid points $N_x$, $N_y$ and $N_z$. The time increment is denoted as $\Delta t$. The notation $u_{i,j,k}^{m} = u(x_i, y_j, z_k, t_m)$ denotes the value of numerical solution at the specific node $(x_i, y_j, z_k)$ and a time step $t_m$. For all splitting methods, we are interested in updating all grid nodes from a time step $t_m$ to the next time step $t_{m+1} = t_m + \Delta t$. 

21
3.1 Enforcing 3D jump conditions in 1D manner

A well-known numerical method is called operator splitting which separates the original equation into several parts over a time step, computes the solution in each time step separately and then combines the solutions together to form a solution to the original equation. The two biggest advantages of operator splitting are stability and efficiency. It is much faster to compute solutions of the split terms than to compute the solution to the original problem directly. There exists many operator splitting method that can reduce a multidimensional problem into a set of independent one-dimensional problems. We will explore a few commonly used operator splitting methods in this paper and test their performances in the proposed parabolic interface problem with different variable diffusion coefficients. With central finite difference discretization in space, all the mentioned operator splitting methods will benefit from the tridiagonal structures, such that can be solved by Thomas algorithm not only accurately but efficiently as well.

3.1.1 Parabolic interface problem with variable coefficients

One of the contribution in this section comparing with previous section is the capability of solving parabolic interface with variable diffusion coefficients in high dimensional space. In (3.2), the diffusion coefficients are no longer only a pair of piecewise constants along two sub-domains \( \Omega^- \) and \( \Omega^+ \) but also denotes piecewise function with respect to the coordinates in three dimension along interface \( \Gamma \). The governing equation (3.1) will be formed as

\[
\frac{\partial u(x,y,z)}{\partial t} = \frac{\partial}{\partial x} (\alpha(x,y,z) \frac{\partial u(x,y,z)}{\partial x}) + \frac{\partial}{\partial y} (\alpha(x,y,z) \frac{\partial u(x,y,z)}{\partial y}) + \frac{\partial}{\partial z} (\alpha(x,y,z) \frac{\partial u(x,y,z)}{\partial z}) + f(x,y,z), \text{ in } \Omega^- \cup \Omega^+,
\]

in three dimensional space, with diffusion coefficients as piecewise function \( \alpha^+(x,y,z) \) in \( \Omega^+ \) and \( \alpha^-(x,y,z) \) in \( \Omega^- \).
3.1.2 Euler discretization with central finite difference

We are favorable to implicit time evolution schemes to semi-discretise (3.1) in time benefiting from the fact that, the equilibrium solution after long-term iteration are usually more meaningful and desirable. First, considering a simplified situation in which the exact solution is smooth throughout entire domain $\Omega$. Technically, there is no jump along interface $\Gamma$ due to the smoothness of analytical solution which yeilds a trivial jump conditions in (3.2). An classic implicit Euler discretization with standard central finite difference scheme is applied on (3.1) in a time interval $[t^m, t^{m+1}]$,

$$
\frac{u_{i,j,k}^{m+1} - u_{i,j,k}^{m}}{\Delta t} = \delta^{\alpha}_{xx}u_{i,j,k}^{m+1} + \delta^{\alpha}_{yy}u_{i,j,k}^{m+1} + \delta^{\alpha}_{zz}u_{i,j,k}^{m+1} + f^{m+1},
$$

(3.4)

where the superscript $m$ denotes the time step, and the subscript is utilized for spatial discretization at specific mesh node at $(x_i, y_j, z_k)$ in three dimensional space, not only the exact solution $u$ and source term $f$ but also the diffusion coefficient $\alpha$ are spatial dependent in all $x$, $y$ and $z$ directions. In (3.4), Operators $\delta^{\alpha}_{xx}$, $\delta^{\alpha}_{yy}$ and $\delta^{\alpha}_{zz}$ are defined with caution to approximate the second partial derivatives $\frac{\partial}{\partial x}(\alpha \frac{\partial u}{\partial x})$, $\frac{\partial}{\partial y}(\alpha \frac{\partial u}{\partial y})$ and $\frac{\partial}{\partial z}(\alpha \frac{\partial u}{\partial z})$ respectively, due to the spatial dependent diffusion coefficient $\alpha(x, y, z)$,

$$
\delta^{\alpha}_{xx}u_{i,j,k}^{m+1} := \frac{1}{\Delta x^2} (\alpha_{i-\frac{1}{2},j,k}u_{i-1,j,k}^{m+1} - (\alpha_{i-\frac{1}{2},j,k} + \alpha_{i+\frac{1}{2},j,k})u_{i,j,k}^{m+1} + \alpha_{i+\frac{1}{2},j,k}u_{i+1,j,k}^{m+1})
$$

$$
\delta^{\alpha}_{yy}u_{i,j,k}^{m+1} := \frac{1}{\Delta y^2} (\alpha_{i,j-\frac{1}{2},k}u_{i,j-1,k}^{m+1} - (\alpha_{i,j-\frac{1}{2},k} + \alpha_{i,j+\frac{1}{2},k})u_{i,j,k}^{m+1} + \alpha_{i,j+\frac{1}{2},k}u_{i,j+1,k}^{m+1})
$$

$$
\delta^{\alpha}_{zz}u_{i,j,k}^{m+1} := \frac{1}{\Delta z^2} (\alpha_{i,j,k-\frac{1}{2}}u_{i,j,k-1}^{m+1} - (\alpha_{i,j,k-\frac{1}{2}} + \alpha_{i,j,k+\frac{1}{2}})u_{i,j,k}^{m+1} + \alpha_{i,j,k+\frac{1}{2}}u_{i,j,k+1}^{m+1})
$$

(3.5)

where values of diffusion coefficients are accessed at half position node on mesh, e.g. $\alpha_{i-\frac{1}{2},j,k} = \alpha(x - \frac{Ax}{2}, y, z)$ and $\alpha_{i+\frac{1}{2},j,k} = \alpha(x + \frac{Ax}{2}, y, z)$. Noticed that in (3.5), all definition actually came from a simplified classic central finite difference scheme at node $u_{i,j,k}$ if a second derivatives are discretized by two nested first derivatives sequentially.
### 3.1.3 Three-dimensional jump condition decomposition

To obtain an accurate solution, it is still not trivial to perfection by applying (3.5) to (3.3) straightforward, because of the discontinuity of analytical solution itself. So special treatment need to be imposed to the interface $\Gamma$ to deliver accurate numerical solutions. Before the introduction to the special treatment, some preprocessing of the three-dimensional jump conditions has to be imposed. Generally, two jump conditions from (3.2) are given physically, the first jump condition $[u]$ can be directly applied, nevertheless, the second jump condition $[au_n]$ is a direction dependent jump condition, it always points to the normal direction along the interface in three-dimensional space which can not be utilized directly. The idea is to transfer 3D jump conditions into 1D ones, i.e., $[au_x]$, $[au_y]$ and $[au_z]$, along Cartesian coordinates using a tensor product decomposition.

At each specific point on interface $\Gamma$ intersected with grid lines in either $x$, $y$ or $z$ directions, a non-orthogonal local coordinate system $(\xi, \eta, \zeta)$ is established with $\xi$ pointing along $\vec{n}$ and $\eta$, $\zeta$ pointing along tangential direction of the interface point. Noted that two tangential directions $\eta$ and $\zeta$ are not necessarily orthogonal so that both directions restricted on raw planes with function either denoted by $x = x_i$, $y = y_j$ or $z = z_k$. After properly define the local $\xi\eta\zeta$–coordinate system, a transformation matrix $P$ is easily conducted, the relation between the two coordinate system is represented by

$$
\begin{bmatrix}
\xi \\
\eta \\
\zeta
\end{bmatrix} = P
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix},
$$

(3.6)

Therefore, it’s trivial to derive the desired jump conditions $[au_x]$, $[au_y]$ and $[au_z]$ from local $\xi\eta\zeta$–coordinate system to original Cartesian coordinate system.

$$
\begin{bmatrix}
\psi_x \\
\psi_y \\
\psi_z
\end{bmatrix} =
\begin{bmatrix}
[au_x] \\
[au_y] \\
[au_z]
\end{bmatrix} = P^{-1}
\begin{bmatrix}
[au_\xi] \\
[au_\eta] \\
[au_\zeta]
\end{bmatrix},
$$

(3.7)
where \([\alpha u_\xi] = [\alpha u_n]\) is defined as the second jump condition. The key to conduct desired jump conditions is calculating \([\alpha u_\eta]\) and \([\alpha u_\zeta]\). From the first given jump condition \([u] = \phi\) in (3.2), two additional jump conditions

\[
[u_\eta] = \phi_\eta, \quad \text{and} \quad [u_\zeta] = \phi_\zeta. \tag{3.8}
\]

are able to be derived along the two tangential directions \(\eta\) and \(\zeta\) in new established local coordinate system. The two unknown jumps of \([\alpha u_\eta]\) and \([\alpha u_\zeta]\) are calculated with the aid of at specific intersection point on interface \(\Gamma\)

\[
[\alpha u_\eta] = \alpha_\Gamma^+ \phi_\eta + (\alpha_\Gamma^+ - \alpha_\Gamma^-)u_\eta^- \quad \text{or} \quad [\alpha u_\eta] = \alpha_\Gamma^- \phi_\eta + (\alpha_\Gamma^+ - \alpha_\Gamma^-)u_\eta^+, \tag{3.9}
\]

\[
[\alpha u_\zeta] = \alpha_\Gamma^+ \phi_\zeta + (\alpha_\Gamma^+ - \alpha_\Gamma^-)u_\zeta^- \quad \text{or} \quad [\alpha u_\zeta] = \alpha_\Gamma^- \phi_\zeta + (\alpha_\Gamma^+ - \alpha_\Gamma^-)u_\zeta^+, \tag{3.10}
\]

where \(u_\eta^+, u_\eta^-, u_\zeta^+,\) and \(u_\zeta^-\) are efficiently approximated by the surrounding on-grid nodes utilizing either extrapolation or interpolation method, giving credit to the well-defined local coordinate system. However, (3.9) and (3.10) are only intended to be conducted at \(t_m\) instead of \(t_{m+1}\) based on the approximation which involves some unavoidable error. Eventually, four decomposed 1D jump conditions are imposed from the involving local \(\xi\eta\zeta\)-coordinate system,

\[
[u] = \phi^{m+1}, \quad [\alpha u_\lambda] = \psi^m_\lambda, \quad [\alpha u_\nu] = \psi^m_\nu, \quad [\alpha u_\zeta] = \psi^m_\zeta, \tag{3.11}
\]

at either time \(t_m\) or \(t_{m+1}\). To obtain a more accurate numerical result, it is preferred to take some advantage of these derived 1D jump conditions for correction of points near interfaces employing implicit Euler scheme in (3.3).

### 3.2 Different spatial discretization

One can see that formulas (3.4) and (3.5) can be applied to all grid nodes directly after the problem has been simplified as a regular parabolic partial differential equation (PDE). However, for interface parabolic interface problem, jump conditions are non-trivial along interface \(\Gamma\), which
technically due to the discontinuity of the analytical solution $u$ and diffusion coefficient $\alpha$. Both
follows different patterns in the two distinguish sub-domains $\Omega^-$ and $\Omega^+$.

Three methods are considered in the work for correction of approximated interface point
near interface $\Gamma$. One is the popular sharp interface scheme ghost fluid method (GFM) originally
developed for treating contact discontinuities in the inviscid Euler equation by Osher and his
coworkers [18]. The method is typically first-order accurate for interface problem and the jump
conditions are captured implicitly by extending values across the interface into a ghost fluid in the
favor of the level set method. By neglecting the tangential derivatives, the GFM is simple and
robust which can be applied to complex interface problems with confidence such as 3D moving
interface. The other two are different schemes of matched interface and boundary (MIB) method
proposed by Y.C. Zhou and his coworkers [77] which is formulated for solving elliptic equation
with the help of involving fictitious points along interface. The MIB method can be utilized to
arbitrary order of accuracy theoretically using only the lowest order jump conditions in an
iterative manner. In two and three dimensional space, the MIB method take full advantage of the
derived tangential derivatives from jump condition, which performs a better accuracy.

3.2.1 Ghost fluid method (GFM)

In this work, we employ the Ghost Fluid method (GFM), to impose the interface condition
(3.2) and correct formulas (3.5) in each $x, y, z$ directions. A brief introduction to the GFM is
provided below, more details could be referred to [18].

There are two major steps in GFM, first, in 2D case (see Fig.3.1(a)), decompose both the
jump condition in normal direction $[\alpha u_n]$ and derived jump condition in tangential direction $[\alpha u_t]$ to $x, y$ directions, respectively, at each interface point. Assuming there is no jump condition along
tangential direction, so that $[\alpha u_t] = 0$. The one can easily obtain

$$[\alpha u_x] = \cos \theta \cdot [\alpha u_n] = \psi_x, \quad [\alpha u_y] = \sin \theta \cdot [\alpha u_n] = \psi_y,$$

where $\theta$ is the angle between normal direction $\vec{n}$ and $x$-axis. In three dimensional space, it follows
Figure 3.1: Three types of points: regular, irregular and corner point. (a). Green curve is interface $\Gamma$ which separates $\Omega^{-}$ and $\Omega^{+}$, two orange nodes lies next to interface $\Gamma$ are called irregular points and the other two black nodes are regular points. (b). Green curve is still interface $\Gamma$ with high curvature, which separates inside sub-domain $\Omega^{-}$ and outside sub-domain $\Omega^{+}$. Three purple points next to interface $\Gamma$ are called corner points and the other two black nodes are regular points.

The same manner to achieve $[\alpha u_x]$, $[\alpha u_y]$ and $[\alpha u_z]$ by assuming the derived jump conditions in two tangential directions $\eta$ and $\zeta$ both zero.

The second major step, is to utilize the jump conditions $[\alpha u_x]$, $[\alpha u_y]$ and $[\alpha u_z]$ to correct finite difference formulas at interface points. Considering $\delta^\eta_{xx}$ as an example and still using Fig.3.1(a) to graphically demonstrate the idea. To simplify the notations, we denote $x_\Gamma - x_{i-1} = \gamma \Delta x$ and thus $x_i - x_\Gamma = (1 - \gamma) \Delta x$, given variable diffusion coefficients as piecewise functions $\alpha^{-}(x, y, z)$ and $\alpha^{+}(x, y, z)$ in the two sub-domains $\Omega^{-}$ and $\Omega^{+}$, one can approximate $\alpha^{-}(u_x^{-})_{\Gamma}^{m+1}$ and $\alpha^{+}(u_x^{+})_{\Gamma}^{m+1}$ by

$$
\alpha^{-}(u_x^{-})_{\Gamma}^{m+1} = \alpha^{-}_{\Gamma} \frac{(u^-)^{m+1}_{\Gamma} - u_{i-1,j,k}^{m+1}}{\gamma \Delta x}, \quad \alpha^{+}(u_x^{+})_{\Gamma}^{m+1} = \alpha^{+}_{\Gamma} \frac{u_{i,j,k}^{m+1} - (u^+)^{m+1}_{\Gamma}}{(1 - \gamma) \Delta x},
$$

(3.13)
at the interface point \( x_I \) at time \( t_{m+1} \) shown in Fig.3.1(a), and substitute into the jump conditions

\[
(u^+)_I^{m+1} - (u^-)_I^{m+1} = [u]^{m+1} = \phi^{m+1},
\]

\[
\alpha^+(u^+_x)_I^{m+1} - \alpha^-(u^-_x)_I^{m+1} = [\alpha u_x]^{m+1} = \psi^{m+1},
\]

solving for \((u^-)_I^{m+1}\) and \((u^+)_I^{m+1}\) as two unknown from the two equations. Accordingly, for the operator \( \delta^\alpha_{xx} \) in (3.5), to correct the finite difference scheme, an extrapolation of \((u^+)_I^{m+1}\), \(u_{i,j,k}^{m+1}\) and \(u_{i+1,j,k}^{m+1}\) can be adopted to correct the value at node \((x_{i+1}, y_j, z_k)\) in sub-domain \(\Omega^+\), while the value at node \((x_i, y_j, z_k)\) in sub-domain \(\Omega^-\) could be corrected by \((u^-)_I^{m+1}\), \(u_{i-1,j,k}^{m+1}\) and \(u_{i-2,j,k}^{m+1}\).

### 3.2.2 MIB-L1 method

Even though the GFM provides a decent approximation to enforce jump conditions at interface point and delivers an accurate numerical solution, it is still negotiable whether the tangential jump conditions should be assumed to be trivial at the beginning. To address this concern, the new three-dimensional jump conditions decomposition technique introduced in section 2 can be carefully advocated in matched interface and boundary (MIB) method.

After decomposed the jump conditions into one-dimensional ones \([\alpha u_x]\), \([\alpha u_y]\) and \([\alpha u_z]\) using the technique we already discussed in section 3.1.3. Two kinds of point are introduced as regular point and irregular point, the distance of these two is the distance of grid point to interface, if a grid point is just next to the interface \(\Gamma\) which means there is no other grid point between the point and interface, it is called an irregular point no matter which sub-domain they are in, otherwise, it is a regular point. As it is shown on Fig.3.1(a), both \(u_{i,j,k}\), \(u_{i-1,j,k}\) are irregular point and \(u_{i+1,j,k}\) and \(u_{i-2,j,k}\) are regular point. For regular point, the operators \(\delta^\alpha_{xx}, \delta^\alpha_{yy}\) and \(\delta^\alpha_{zz}\) are calculated from (3.5), while in terms of irregular point, based on the sub-domain, a similar correction like GFM should be accomplished, \(\delta^\alpha_{xx}\) is modified to be

\[
\delta^\alpha_{xx} u_{i+1,j,k}^{m+1} = \frac{1}{\Delta x^2} \left( (\alpha^-_{i-\frac{1}{2},j,k} u_{i-1,j,k}^{m+1} - (\alpha^-_{i-\frac{1}{2},j,k} + \alpha^-_{i-\frac{1}{2},j,k})u_{i-1,j,k}^{m+1} + \alpha^-_{i-\frac{1}{2},j,k} u_{i+1,j,k}^{m+1} \right),
\]

\[
\delta^\alpha_{xx} u_{i,j,k}^{m+1} = \frac{1}{\Delta x^2} \left( (\alpha^+_{i+\frac{1}{2},j,k} u_{i+1,j,k}^{m+1} - (\alpha^+_{i+\frac{1}{2},j,k} + \alpha^+_{i+\frac{1}{2},j,k})u_{i+1,j,k}^{m+1} + \alpha^+_{i+\frac{1}{2},j,k} u_{i,j,k}^{m+1} \right),
\]
at the two irregular point \( u_{l-1, jk}, u_{i, jk} \) at time \( t_{m+1} \), where \( \tilde{u}_{l-1, jk} \) and \( \tilde{u}_{i, jk} \) are two fictitious values at node \((x_{i-1}, y_j, z_k)\) and \((x_i, y_j, z_k)\), respectively, which are determined by discretizing two jump conditions in (3.14) and (3.15),

\[
\omega_{0,1}^+ u_{i, jk} + \omega_{0,2}^+ \tilde{u}_{l-1, jk} = \omega_{0,1}^- \tilde{u}_{i, jk} + \omega_{0,2}^- u_{l-1, jk} + \phi \tag{3.18}
\]

\[
\alpha_{i, jk}^+ \omega_{1,1}^+ u_{i, jk} + \alpha_{l-1, jk}^+ \omega_{1,2}^+ \tilde{u}_{l-1, jk} = \alpha_{i, jk}^- \omega_{1,1}^- \tilde{u}_{i, jk} + \alpha_{l-1, jk}^- \omega_{1,2}^- u_{l-1, jk} + \psi_x \tag{3.19}
\]

where \( \omega_{i,j}^+ \) and \( \omega_{i,j}^- \) are finite difference weights [19] with \( i = 0 \) representing interpolation and \( i = 1 \) representing first order approximation. Solving (3.18) and (3.19) allow the two fictitious values \( \tilde{u}_{l-1, jk} \) and \( \tilde{u}_{i, jk} \) to be written as a linear combination of the two points \( u_{l-1, jk} \) and \( u_{i, jk} \) and two jump conditions \( \phi \) and \( \psi_x \). Due to the fact that \( \psi_x \) is derived from jump conditions \([\alpha u_\xi], [\alpha u_\eta] \) and \([\alpha u_\zeta] \) in (3.7), the two fictitious values actually are written as a linear combinations of

\[
\tilde{u}_{l-1, jk} = [a_0, a_1, a_2, a_3, a_4, a_5] \cdot [u_{l-1, jk}, u_{i, jk}, [u], [\alpha u_\eta], [\alpha u_\xi], [\alpha u_\zeta]]^T \tag{3.20}
\]

\[
\tilde{u}_{i, jk} = [b_0, b_1, b_2, b_3, b_4, b_5] \cdot [u_{l-1, jk}, u_{i, jk}, [u], [\alpha u_\eta], [\alpha u_\xi], [\alpha u_\zeta]]^T, \tag{3.21}
\]

where all \( a_i \) and \( b_i \) are coefficients determined from solving (3.18) and (3.19). The fictitious value calculation is a time independent process, which is relatively cheap and only has to be performed once at the beginning of time iteration. Once all the coefficients \( a_i \) and \( b_i \) are attained, at each time step \( t_{m+1} \), followed by a substitution of the two fictitious values, the correction of \( \delta_{\alpha}^{m+1} \) is written as

\[
\delta_{x\alpha}^{m+1} u_{i, jk}^{m+1} = D_{\alpha x}^{\alpha} u_{l-1, jk}^{m+1} + B_{\alpha}^{\alpha} u_{i, jk}^{m} + \Phi_x^{m+1}, \tag{3.22}
\]

where \( D_{\alpha x}^{\alpha} \) is perturbed operators which have a band-width three at nodes \((x_i - 1, y_j, z_k)\), \((x_i, y_j, z_k)\) and \((x_i + 1, y_j, z_k)\). The operators \( B_{\alpha}^{\alpha} \) is due to tangential derivative approximations, which involves the value of point at previous time step \( t_m \). The jump terms \( \Phi_x^{m+1} \) is the sums of all jump conditions \([u], [\alpha u_\eta], [\alpha u_\xi], [\alpha u_\zeta] \). The exact same idea will be adopted to \( y \) and \( z \) directions.

Despite of the jump conditions for \([\alpha u_\xi], [\alpha u_\eta] \) and \([\alpha u_\zeta] \), the MIB-L1 method is actually
conditions in (3.14) and (3.15) points at nodes (same basic idea of MIB method but achieve a locally second order accuracy for nodes around interface \( \Gamma \)). The value of nodes around interface \( \Gamma \) the mathematical equivalent to the GFM, both employ the one-dimensional jump condition to correct the value of nodes around interface \( \Gamma \) with locally first order.

### 3.2.3 MIB-L2 method

Now, we introduce the second form of MIB method, MIB-L2 method, which inherit the same basic idea of MIB method but achieve a locally second order accuracy for nodes around interface. Similarly, to achieve second order accuracy to evaluate fictitious values, two more points at nodes \((x_{i-2}, y_j, z_k)\) and \((x_{i+1}, y_j, z_k)\) are immersed when discretizing the two jump conditions in (3.14) and (3.15)

\[
\omega_{0,1}^+ u_{i+1,j,k} + \omega_{0,2}^+ u_{i,j,k} + \omega_{0,3}^+ \tilde{u}_{i-1,j,k} = \omega_{0,1}^- \tilde{u}_{i,j,k} + \omega_{0,2}^- u_{i-1,j,k} + \omega_{0,3}^- u_{i-2,j,k} + \phi, \quad (3.23)
\]

\[
\alpha_{i+1,j,k}^+ \omega_{1,1}^+ u_{i+1,j,k} + \alpha_{i,j,k}^+ \omega_{1,2}^+ u_{i,j,k} + \alpha_{i-1,j,k}^+ \omega_{1,3}^+ \tilde{u}_{i-1,j,k} = \alpha_{i,j,k}^- \omega_{1,1}^- \tilde{u}_{i,j,k} + \alpha_{i-1,j,k}^- \omega_{1,2}^- u_{i-1,j,k} + \alpha_{i-2,j,k}^- \omega_{1,3}^- u_{i-2,j,k} + \phi_x. \quad (3.24)
\]

After solving (3.23) and (3.24), \( \tilde{u}_{i-1,j,k} \) and \( \tilde{u}_{i,j,k} \) are written as linear combinations of

\[
\tilde{u}_{i-1,j,k} = [a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7] \cdot [u_{i-2,j,k}, u_{i-1,j,k}, u_{i,j,k}, u_{i+1,j,k}, u, [au_\eta], [au_\xi], [au_\zeta]]^T \quad (3.25)
\]

\[
\tilde{u}_{i,j,k} = [b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7] \cdot [u_{i-2,j,k}, u_{i-1,j,k}, u_{i,j,k}, u_{i+1,j,k}, u, [au_\eta], [au_\xi], [au_\zeta]]^T, \quad (3.26)
\]

where all \( a_i \) and \( b_i \) are still linear coefficients, which only need to be computed once before time iterations. To preserve second order accuracy in space, more situation has to be taken into consideration other than simply extend the scheme from MIB-L1 method. If the interface has a high curvature, see Fig.3.1(b), which make it really sharp or even non-smooth sometime, only one grid point will be found inside the region \( \Omega^- \) no matter how dense the grid mesh is. It is called a corner case, and all the three points around the two side of the interfaces are called corner points. Three fictitious can be calculated located at the three corner points from four interface conditions
at $x_{\Gamma_1}$ and $x_{\Gamma_2}$ in Fig.3.1(b) given by

\[
\omega_{0,1}^+ u_{i+2,j,k} + \omega_{0,2}^+ u_{i+1,j,k} + \omega_{0,3}^+ \bar{u}_{i,j,k} + \omega_{0,4}^+ u_{i-1,j,k} = \\
\omega_{0,1}^- \bar{u}_{i+1,j,k} + \omega_{0,2}^- u_{i,j,k} + \omega_{0,3}^- \bar{u}_{i-1,j,k} + \omega_{0,4}^- \bar{u}_{i-2,j,k} + \phi_{\Gamma_1},
\]

(3.27)

\[
\omega_{1,1}^+ u_{i+2,j,k} + \omega_{1,2}^+ u_{i+1,j,k} + \omega_{1,3}^+ \bar{u}_{i,j,k} + \omega_{1,4}^+ u_{i-1,j,k} = \\
\omega_{1,1}^- \bar{u}_{i+1,j,k} + \omega_{1,2}^- u_{i,j,k} + \omega_{1,3}^- \bar{u}_{i-1,j,k} + \omega_{1,4}^- \bar{u}_{i-2,j,k} + \phi_{\Gamma_2},
\]

(3.28)

\[
\alpha_{i+2,j,k}^+ \omega_{2,1}^+ u_{i+2,j,k} + \alpha_{i+1,j,k}^+ \omega_{2,2}^+ u_{i+1,j,k} + \alpha_{i,j,k}^+ \omega_{2,3}^+ \bar{u}_{i,j,k} + \alpha_{i-1,j,k}^+ \omega_{2,4}^+ u_{i-1,j,k} = \\
\alpha_{i+1,j,k}^- \omega_{2,1}^- \bar{u}_{i+1,j,k} + \alpha_{i,j,k}^- \omega_{2,2}^- \bar{u}_{i,j,k} + \alpha_{i-1,j,k}^- \omega_{2,3}^- \bar{u}_{i-1,j,k} + \alpha_{i-2,j,k}^- \omega_{2,4}^- \bar{u}_{i-2,j,k} + \psi_{x_{\Gamma_1}},
\]

(3.29)

\[
\alpha_{i+2,j,k}^+ \omega_{3,1}^+ u_{i+2,j,k} + \alpha_{i+1,j,k}^+ \omega_{3,2}^+ u_{i+1,j,k} + \alpha_{i,j,k}^+ \omega_{3,3}^+ \bar{u}_{i,j,k} + \alpha_{i-1,j,k}^+ \omega_{3,4}^+ u_{i-1,j,k} = \\
\alpha_{i+1,j,k}^- \omega_{3,1}^- \bar{u}_{i+1,j,k} + \alpha_{i,j,k}^- \omega_{3,2}^- \bar{u}_{i,j,k} + \alpha_{i-1,j,k}^- \omega_{3,3}^- \bar{u}_{i-1,j,k} + \alpha_{i-2,j,k}^- \omega_{3,4}^- \bar{u}_{i-2,j,k} + \psi_{x_{\Gamma_2}},
\]

(3.30)

Note that there is one more extra fictitious values $\bar{u}_{i-2,j,k}$ involved, since we would like to avoid extra degree of freedom, a non-degenerated matrix has to be constructed for solving the fictitious values. All the three fictitious values $\bar{u}_{i-1,j,k}$, $\bar{u}_{i,j,k}$ and $\bar{u}_{i+1,j,k}$ will be eventually represented as a linear combination of all the five grid nodes $u_{i-2,j,k}$, $u_{i-1,j,k}$, $u_{i,j,k}$, $u_{i+1,j,k}$, $u_{i+2,j,k}$ and the four jump conditions $[u]$, $[au_x]$, $[au_y]$, $[au_z]$ as (3.25) and (3.26).

A similar way can be performed in calculating fictitious values in $y$ and $z$ directions, afterward, at each time step $t_{m+1}$, one can just substitute related values of nodes for jump approximation at previous time step $t_m$ and required jump conditions at current time step $t_{m+1}$ to
end up with the final forms of corrected operations

\[
\delta_{xx}^{\alpha} u^{m+1}_{i,j,k} = D_{xx}^{\alpha} u^{m+1}_{i,j,k} + B_{x}^{\alpha} u^{m}_{i,j,k} + \Phi^{m+1}_{x}, \tag{3.31}
\]

\[
\delta_{yy}^{\alpha} u^{m+1}_{i,j,k} = D_{yy}^{\alpha} u^{m+1}_{i,j,k} + B_{y}^{\alpha} u^{m}_{i,j,k} + \Phi^{m+1}_{y}, \tag{3.32}
\]

\[
\delta_{zz}^{\alpha} u^{m+1}_{i,j,k} = D_{zz}^{\alpha} u^{m+1}_{i,j,k} + B_{z}^{\alpha} u^{m}_{i,j,k} + \Phi^{m+1}_{z}, \tag{3.33}
\]

where all node values at time step \( t_{m+1} \) will be solved in the following time iterations and all node values at time step \( t_{m} \) are provided from previous time step, meanwhile, all jump related operators \( \Phi^{m+1}_{x}, \Phi^{m+1}_{y} \) and \( \Phi^{m+1}_{z} \) are either derived or given from jump condition directly.

### 3.3 Operator splitting methods with variable coefficients

During the recent decades, operator splitting methods has gained a lot of reputation for solving high-dimensional PDE problems, benifits from the efficiency and some unconditional stable property. These methods split the spatial dimensions and solve a 2D or 3D problem as two or three consecutive 1D problems. In the section we will introduce four different operator splitting methods Douglas-Rachford alternating direction implicit (D-ADI) method, Locally one-dimensional implicit Euler (LOD-IE) method, Locally one-dimensional Crank-Nicolson (LOD-CN) method and Trapezoidal Splitting (TS) method, respectively, and applied each to the original discretized equation (3.4).

#### 3.3.1 Douglas-Rachford ADI scheme

To construct a first order Douglas-Rachford ADI method [15, 46] based on the implicit Euler schemes of parabolic PDE (3.4), at each grid point \((x, y, z)\) we rewrite (3.4) as

\[
(1 - \Delta t \delta_{xx}^{\alpha}) u_{i,j,k}^{*} = (1 + \Delta t (\delta_{yy}^{\alpha} + \delta_{zz}^{\alpha})) u_{i,j,k}^{m} + \Delta t f_{i,j,k}^{m+1},
\]

\[
(1 - \Delta t \delta_{yy}^{\alpha}) u_{i,j,k}^{**} = u_{i,j,k}^{*} - \Delta t \delta_{yy}^{\alpha} u_{i,j,k}^{m}, \tag{3.34}
\]

\[
(1 - \Delta t \delta_{zz}^{\alpha}) u_{i,j,k}^{m+1} = u_{i,j,k}^{**} - \Delta t \delta_{zz}^{\alpha} u_{i,j,k}^{m}.
\]
where the two terms \( u_{i,j,k}^* \) and \( u_{i,j,k}^{**} \) are intermediate terms. It’s not hard to derive the similarity between Douglas-Rachford ADI scheme and the original implicit Euler scheme (3.4) after dropping some higher order terms. The temporal order of Douglas-Rachford ADI method is the same with implicit Euler scheme, however, in respect of efficiency and computational cost, ADI is much faster than implicit Euler schemes with complexity \( O(N) \), where \( N \) is the total degree of freedom. The original left hand side matrix is not tridiagonal structure because of fictitious values, however, the non-tridiagonal parts are either \( 2 \times 4 \) or \( 3 \times 5 \) block matrices which can easily be simplified to tridiagonal structure by Gaussian elimination. The one-dimensional decomposition build up a tridiagonal matrix on the left hand side of the three equations (3.34) which taking full advantages of Thomas algorithm.

To deliver the optimal accuracy, jump condition corrections are imposed on the original implicit Euler scheme (3.4) with MIB method, the original implicit Euler scheme (3.4) are rewritten as

\[
\frac{u_{i,j,k}^{m+1} - u_{i,j,k}^m}{\Delta t} = D_{xx}^\alpha u_{i,j,k}^{m+1} + D_{yy}^\alpha u_{i,j,k}^{m+1} + D_{zz}^\alpha u_{i,j,k}^{m+1} + F_{i,j,k}^{m+1},
\]

(3.35)

where \( F_{i,j,k}^{m+1} = (B_x^\alpha + B_y^\alpha + B_z^\alpha)u_{i,j,k}^m + \Phi_x^{m+1} + \Phi_y^{m+1} + \Phi_z^{m+1} + f_{i,j,k}^{m+1} \). The Douglas-Rachford ADI method for the implicit Euler scheme (3.35) is simply given as

\[
\begin{align*}
(1 - \Delta tD_{xx}^\alpha)u_{i,j,k}^* &= (1 + \Delta t(D_{yy}^\alpha + D_{zz}^\alpha))u_{i,j,k}^m + \Delta tF_{i,j,k}^{m+1}, \\
(1 - \Delta tD_{yy}^\alpha)u_{i,j,k}^{**} &= u_{i,j,k}^* - \Delta tD_{yy}^\alpha u_{i,j,k}^m, \\
(1 - \Delta tD_{zz}^\alpha)u_{i,j,k}^{m+1} &= u_{i,j,k}^{**} - \Delta tD_{zz}^\alpha u_{i,j,k}^m,
\end{align*}
\]

(3.36)

which keep the semi-descretization and first temporal order, but more perturbation terms are eliminated in the proposed Douglas-Rachford ADI scheme from implicit Euler scheme.

\[3.3.2\] **Locally one-dimensional implicit Euler (LOD-IE) schemes**

The fractional step methods or locally one-dimensional (LOD) methods were first developed by Russian mathematicians [62]. Being multiplicative operator splitting methods too, the LOD methods adopt the similar splitting in alternating directions as in the ADI methods, but
before the numerical discretization, we will formulate two types of temporal discretizations based
on implicit Euler and Crank-Nicolson schemes.

First, we propose the implicit Euler integration in time (3.35), decompose the three
dimensional discretization form into three one-dimensional ones which leads to the LOD-IE
scheme

\[(1 - \Delta t D^\alpha_{xx}) u^*_{i,j,k} = u^m_{i,j,k}, \]

\[(1 - \Delta t D^\alpha_{yy}) u^{**}_{i,j,k} = u^*_{i,j,k}, \] (3.37)

\[(1 - \Delta t D^\alpha_{zz}) u^{***}_{i,j,k} = u^{**}_{i,j,k}, \]

\[u_{i,j,k}^{m+1} = u_{i,j,k}^{***} + \Delta t F^m_{i,j,k}, \] (3.38)

where all operators \(D^\alpha_{xx}, D^\alpha_{yy}, D^\alpha_{zz} \) and source term \(F^m_{i,j,k} \) has been corrected for nodes near
interface \(\Gamma\) spatially. The LOD-IE scheme has exact same right hand side with Douglas-Rachford
ADI scheme (3.36) but a simplified left hand side, therefore, it keep the unconditional stability
property but less operations are expected in the computation.

3.3.3 Locally one-dimensional Crank-Nicolson (LOD-CN) schemes

Minor change can be resulted if we consider a well known Crank-Nicolson integration in
time based on governing equation (3.1) with corrected operators in all three directions,

\[\frac{u_{i,j,k}^{m+1} - u_{i,j,k}^m}{\Delta t} = \frac{1}{2} (D^\alpha_{xx} u_{i,j,k}^m + D^\alpha_{yy} u_{i,j,k}^m + D^\alpha_{zz} u_{i,j,k}^m + D^\alpha_{xx} u_{i,j,k}^{m+1} + D^\alpha_{yy} u_{i,j,k}^{m+1} + D^\alpha_{zz} u_{i,j,k}^{m+1} + F_{i,j,k}^m + F_{i,j,k}^{m+1}), \] (3.39)
Utilizing locally one dimensional scheme based on (3.39) leads to

\[
(1 - \frac{\Delta t}{2} D_{xx}^\alpha) u_{i,j,k}^* = (1 + \frac{\Delta t}{2} D_{xx}^\alpha) u_{i,j,k}^m,
\]

\[
(1 - \frac{\Delta t}{2} D_{yy}^\alpha) u_{i,j,k}^{**} = (1 + \frac{\Delta t}{2} D_{xx}^\alpha) u_{i,j,k}^*,
\]

\[
(1 - \frac{\Delta t}{2} D_{zz}^\alpha) u_{i,j,k}^{***} = (1 + \frac{\Delta t}{2} D_{xx}^\alpha) u_{i,j,k}^{**},
\]

\[
u_{i,j,k}^{m+1} = u_{i,j,k}^{**} + \frac{1}{2}(\Delta t F_{i,j,k}^{m+1} + \Delta t F_{i,j,k}^m).
\]  

(3.40) 

One may think the scheme is second-order in time, nonetheless, because the jump approximation is only first order accuracy, the scheme still keep first order in time. Fortunately, unconditional stable is still well preserved on this LOD-CN scheme.

### 3.3.4 Trapezoidal Splitting (TS) schemes

Inspired by [27] about trapezoidal splitting, one splitting method which is based on the trapezoidal rule,

\[
u^{m+1} = u^m + \frac{1}{2}\Delta t G(t_m, u^m) + \frac{1}{2}\Delta t G(t_{m+1}, u^{m+1}),
\]  

(3.42)

where the function \(G\) can be treated as the entire right hand side of the governing equation (3.1).

There are three direction and source term involved in the governing equation, it is possible to decompose the function \(G\) into four simpler component functions,

\[
G(t, u) = G_1(t, u) + G_2(t, u) + G_3(t, u) + G_4(t, u),
\]  

(3.43)
where $G_1$, $G_2$, $G_3$ are indeed the corrected operators $D_{xx}^\alpha$, $D_{yy}^\alpha$, $D_{zz}^\alpha$ and $G_4$ is the source term $F$, which ultimately leads to a large set of system of equations

\[
\begin{align*}
\nu_{i,j,k}^{(1)} &= (1 + \frac{\Delta t}{2} D_{xx}^\alpha) u_{i,j,k}^m, \\
\nu_{i,j,k}^{(2)} &= (1 + \frac{\Delta t}{2} D_{yy}^\alpha) \nu_{i,j,k}^{(1)}, \\
\nu_{i,j,k}^{(3)} &= (1 + \frac{\Delta t}{2} D_{zz}^\alpha) \nu_{i,j,k}^{(2)}, \\
\nu_{i,j,k}^{(4)} &= \nu_{i,j,k}^{(3)} + \frac{\Delta t}{2} F_{i,j,k}^m, \\
\nu_{i,j,k}^{(5)} &= \nu_{i,j,k}^{(4)} + \frac{\Delta t}{2} F_{i,j,k}^{m+1}. \\
\end{align*}
\]

(3.44)

as the trapezoidal splitting method with corrected operators. One may notice that the system is not entirely symmetric due to the fifth equation in (3.44), which is because of the efficiency reason. To take a symmetric form in the fifth equation of (3.44), source term $F_{i,j,k}^{m+1} \nu_{i,j,k}^{(4)}$ has to be replaced by $F_{i,j,k}^m \nu_{i,j,k}^{(3)}$. Therefore, when derived jump conditions will involve an implicitly part represented $(B_x^\alpha + B_y^\alpha + B_z^\alpha) \nu_{i,j,k}^5$, which should move to the left hand side to solve the equation, the matrix structure will be detracted from a tridiagonal matrix and has to be solved by some expensive iterative algorithm instead of Thomas algorithm with $O(N)$.

In the computations, all boundary conditions are Dirichlet boundary conditions which assumed to be at the next time step $t_{m+1}$ when calculating the intermediate terms. If more than first order of accuracy is considered in space, technique [70] using MIB method in high order central finite difference can be utilized.
CHAPTER 4
NUMERICAL EXPERIMENTS

The numerical performance of all proposed methods is measured by two error measurements, $L_\infty$ and $L_2$ norms, which are defined as

$$L_\infty = \max_{i,j,k} | u(x_i, y_j, z_k) - u_h(x_i, y_j, z_k) |,$$

$$L_2 = \sqrt{\frac{\sum_{i,j,k} (u(x_i, y_j, z_k) - u_h(x_i, y_j, z_k))^2}{N_x \times N_y \times N_z}}.$$  \hfill (4.1)

(4.2)

where $u_h(x_i, y_j, z_k)$ is the numerical solution, and $N_x$, $N_y$ and $N_z$ represent the numbers of grid nodes on $x$, $y$ and $z$ direction. The order of convergence for both $L_2$ and $L_\infty$ is reported in all tested cases. For the surface solution and error maps, the values on the interface points are mapped by the closest grid node from exterior sub-domain $\Omega^+$, and the numbers of mesh grid are chosen to be large so that the exterior value of the surface will be extremely close to its closest outside nodal value.

To better understand the implementation in three dimensional space, a visualization of sphere is proposed to illustrate the finite different process in Fig. 4.1. For other surface we used in numerical experiments, the idea is the same.

4.1 Matched ADI with constant piece-wise coefficients

In this section, we examine both temporal and spatial convergence rates of the proposed 3D matched ADI algorithm with various shapes of interfaces and jump conditions. The code is written in C++ based on the idea of objected oriented programming. The data structure “vector” in C++ STL is selected to perform the linked-list feature. With such a new data structure, the grid
Figure 4.1: Implementation illustration based on finite difference in three dimensional space. The surface $\Gamma$ is a sphere, which separate the two subdomains. In three dimensional space, a straight line cuts through the entire domain $\Omega$ at each grid point in each direction, which intersects the surface at some arbitrary node.

A straight line can cross the interface many times without restriction, while in our previous 2D matched ADI methods [35, 68], it allowed to cross at most twice. The present data structure enables us to treat more complicated geometry in 3D. Reported CPU time is recorded on the master node of SGI Ultraviolet in Alabama Supercomputer Center with a single Xeon E5-4640 CPU core operating at 2.4 GHz clock speed.

In all examples, a piecewisely defined analytical solution $u(x, y, z)$ is constructed so that both the initial condition at time $t = 0$ and Dirichlet boundary conditions on the boundary of a cubic computational domain $\Omega$ can be obtained from the constructed analytical solution directly.

Unless specified individually, the integration will be carried out until a stopping time $t = 1$ and the diffusion coefficients are selected as $\alpha^- = 1$ and $\alpha^+ = 10$. The domain size is chosen as a
non-integer so that one can avoid the situation, in which the interface intersects the Cartesian grid lines through one grid node, because such a situation rarely happens for a complex geometry in practice. Moreover, by taking the domain size very close to an integer, such as 1.99, the corner interface case could be encountered in a coarse mesh. This enables us to fully validate the proposed matched ADI algorithm.

**Example 1.** The first example has a simple smooth ellipsoid surface, which is defined as the zero level set of

\[ S(x, y, z) = \frac{x^2}{4} + y^2 + z^2 - 1. \]  

(4.3)

This interface is regular and convex so that each grid line cuts the interface at most twice. The analytical solution to eqn. (1.1) is constructed to be

\[
 u(x, y, z, t) = \begin{cases} 
 10e^{-x^2}e^{-y^2}e^{-z^2} - e^{-a}, & \text{in } \Omega^- \ \\
 5e^{-x^2}e^{-y^2}e^{-z^2} + e^{-a}, & \text{in } \Omega^+ 
\end{cases}
\]  

(4.4)

with \( a = 2 \). The source term is

\[
 f(x, y, z, t) = \begin{cases} 
 -10\alpha (-6 + 4(x^2 + y^2 + z^2))e^{-x^2}e^{-y^2}e^{-z^2} - e^{-a}, & \text{in } \Omega^- \ \\
 -5\alpha^+ (-6 + 4(x^2 + y^2 + z^2))e^{-x^2}e^{-y^2}e^{-z^2} + e^{-a}. & \text{in } \Omega^+ 
\end{cases}
\]  

(4.5)

so that the zeroth order jump condition on the interface is time-dependent, while the first order jump condition is time-independent. The computational domain is fixed to be

\([-3.99, 3.99] \times [-1.99, 1.99] \times [-1.99, 1.99]\).

The proposed matched ADI algorithm is stable and convergent for all tested combinations of \( h \) and \( \Delta t \). In order to demonstrate the spatial convergence rate, \( \Delta t = 10^{-4} \) is set to be small. Numerical results obtained by varying the number of grids per direction are demonstrated in Table 4.1, and it is obvious that the spatial convergence rate is close to 2 for both \( L_{\infty} \) and \( L_2 \) norms. The next set of tests is performed with a dense space mesh \([N_x, N_y, N_z] = [160, 160, 160]\),
and various time steps. The results are depicted in Fig. 4.2 with dashed lines. One can observe that the convergence is not evident for $\Delta t > 10^{-2}$, probably because of the relatively large errors in approximating tangential derivatives for a large $\Delta t$. Once the convergence starts at $\Delta t = 10^{-2}$, the rate is pretty high. To see this, a least-squares fitting [70] is conducted in the log-log scale to analyze the temporal convergence rate, which is demonstrated by the solid lines in Figure 4.2. Their slopes shown in the legend of Figure 4.2 represent the temporal convergence rate of the scheme. It shows that both solid lines for $L_2$ and $L_\infty$ errors have slopes about 40% higher than
their theoretical orders (one).

The color maps of numerical solutions and errors of Example 1 are depicted in Fig. 4.3 for 

\[ [N_x, N_y, N_z] = [160, 160, 160] \text{ and } \Delta t = 10^{-4}. \]

It is found that large errors occur in the places where the solutions have large magnitudes. This is natural due to the simple shape of the ellipsoid surface.

![Color maps for ellipsoid surface](image)

Figure 4.3: Color maps for ellipsoid surface. (a). numerical solution; (b). numerical error.

**Example 2.** The second example has a simple cylinder surface, which is defined as an implicit function

\[
\frac{x^2}{4} + \frac{y^2}{4} = 1, \quad -3 \leq z \leq 3.
\]

(4.6)

In two-dimensional views, this interface is either a rectangle or a circle. The same analytical solution (4.4) and source term (4.5) with \( a = 1 \) as in Example 1 are studied. The computational domain is fixed to be \([-3.99, 3.99] \times [-3.99, 3.99] \times [-3.99, 3.99]\).

We are interested in studying different combinations for the diffusion coefficients for this example. Besides the usual combination with \((\alpha^-, \alpha^+) = (1, 10)\), two more cases with 

\((\alpha^-, \alpha^+) = (10, 1)\) and \((\alpha^-, \alpha^+) = (1, 100)\) are examined. The 3D matched ADI method is found to be stable for all the tested \( h \) and \( \Delta t \) for these three pairs of diffusion coefficients. From Table 4.2 to Table 4.4, to avoid the influence of temporal numerical errors, a small fixed time increment \( \Delta t = 10^{-4} \) is adopted for all spatial convergence tests. The results of all three tables indicate that both \( L_2 \) and \( L_\infty \) norms achieve second order convergence rate regardless of the diffusion.
<table>
<thead>
<tr>
<th>([N_x, N_y, N_z])</th>
<th>(L_{\infty}) Error</th>
<th>(L_{\infty}) Order</th>
<th>(L_2) Error</th>
<th>(L_2) Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>([20, 20, 20])</td>
<td>4.548e-01</td>
<td>1.90</td>
<td>2.262e-02</td>
<td>2.14</td>
</tr>
<tr>
<td>([40, 40, 40])</td>
<td>1.215e-01</td>
<td>2.00</td>
<td>5.122e-03</td>
<td>2.06</td>
</tr>
<tr>
<td>([80, 80, 80])</td>
<td>3.020e-02</td>
<td>2.11</td>
<td>1.227e-03</td>
<td>2.06</td>
</tr>
<tr>
<td>([160, 160, 160])</td>
<td>7.491e-03</td>
<td>2.11</td>
<td>3.035e-04</td>
<td>2.02</td>
</tr>
</tbody>
</table>

Table 4.2: Spatial convergence for cylinder surface with \(\alpha^{-} = 1\) and \(\alpha^{+} = 10\).

Figure 4.4: Temporal convergence for cylinder surface with \(\alpha^{-} = 1\) and \(\alpha^{+} = 10\).

coefficient’s combinations. By fixing the number of grids in each direction to be 160, the temporal convergence is investigated by the least-square fitting analysis in Figure 4.4 to 4.6. It is clear that \(L_2\) and \(L_{\infty}\) errors yield a similar behavior, and both converge at a rate slightly higher than the first order.

**Example 3.** The third example studies the impact of the curvature of the interface on the proposed numerical scheme. To this end, a smooth molecular surface of two atoms is constructed by the zero level set of

\[
S(x, y, z) = (x^2 + y^2 + z^2 + \frac{3}{5})^2 - \frac{7}{2}y^2 - \frac{3}{5}
\]  

(4.7)

within a cubic computational domain \(\Omega\) of dimension.
\[ [N_x, N_y, N_z] \quad \begin{array}{c|c|c|c|c|}
\hline
& L_\infty & & L_2 & \\
\hline
Error & Order & Error & Order \\
\hline
[20, 20, 20] & 5.271e-01 & 2.30 & 3.802e-02 & 2.27 \\
[40, 40, 40] & 1.074e-01 & 2.06 & 7.891e-03 & 1.72 \\
[80, 80, 80] & 2.571e-02 & 1.92 & 2.388e-03 & 1.72 \\
\hline
\end{array} \]

Table 4.3: Spatial convergence for cylinder surface with \( \alpha^- = 10 \) and \( \alpha^+ = 1 \).

Figure 4.5: Temporal convergence for cylinder surface with \( \alpha^- = 10 \) and \( \alpha^+ = 1 \).

\([-1.99, 1.99] \times [-2.99, 2.99] \times [-1.99, 1.99] \), as shown in Figure 4.7. The concave region which connects the two atoms makes it impossible to derive the jump conditions \([\alpha u_\eta]\) and \([\alpha u_\zeta]\) without approximating the tangential derivative \(u_\eta\) and \(u_\zeta\) in both outer and inner domains \((\Omega^+ \text{ and } \Omega^-)\) simultaneously. The same analytical solution (4.4) and source term (4.5) with \( a = 1 \) are used.

Once again, the proposed matched ADI method is stable and converges for all tested \( h \) and \( \Delta t \). To demonstrate the results, \( \Delta t = 10^{-4} \) is fixed for spatial convergence tests and the results are shown in Table 4.5. The results clearly conclude that the spatial convergence rate is two. The results obtained for the temporal convergence test by fixing the number of grids per direction to be 160 are shown in Figure 4.8, together with the results of least-squares error analysis. In this
Table 4.4: Spatial convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 100$.

<table>
<thead>
<tr>
<th>$[N_x, N_y, N_z]$</th>
<th>$L_\infty$ Error</th>
<th>Order</th>
<th>$L_2$ Error</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20, 20, 20]</td>
<td>4.562e-01</td>
<td></td>
<td>2.275e-02</td>
<td></td>
</tr>
<tr>
<td>[40, 40, 40]</td>
<td>1.224e-01</td>
<td>1.90</td>
<td>5.241e-03</td>
<td>2.12</td>
</tr>
<tr>
<td>[80, 80, 80]</td>
<td>3.038e-02</td>
<td>2.01</td>
<td>1.243e-03</td>
<td>2.08</td>
</tr>
<tr>
<td>[160, 160, 160]</td>
<td>7.525e-03</td>
<td>2.01</td>
<td>3.058e-04</td>
<td>2.02</td>
</tr>
</tbody>
</table>

Figure 4.6: Temporal convergence for cylinder surface with $\alpha^- = 1$ and $\alpha^+ = 100$.

Figure 4.7: Color maps for molecular surface. (a). numerical solution; (b). numerical error.
case, both $L_\infty$ and $L_2$ errors decrease in a similar manner. Although it is not very uniform for all $\Delta t$’s, the overall order of the matched ADI method is still higher than one for both error measurements. The color map of numerical solutions and errors are illustrated in Figure 4.7 for $[N_x, N_y, N_z] = [160, 160, 160]$ and $\Delta t = 10^{-4}$. The error distribution pattern is almost the same as that of the solution.

To appreciate the numerical difficulty of the present parabolic interface problem, the slice plots of the matched ADI solutions at the cross section $z = 0$ are depicted in Fig. 4.9. Here, we take $[N_x, N_y, N_z] = [160, 160, 160]$ and carry out the integration from $t = 0$ to $t = 2$ with

<table>
<thead>
<tr>
<th>$[N_x, N_y, N_z]$</th>
<th>$L_\infty$ Error</th>
<th>Order</th>
<th>$L_2$ Error</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20, 20, 20]</td>
<td>2.135e-01</td>
<td>2.30</td>
<td>2.140e-02</td>
<td>2.48</td>
</tr>
<tr>
<td>[40, 40, 40]</td>
<td>4.341e-02</td>
<td>1.94</td>
<td>3.833e-03</td>
<td>2.48</td>
</tr>
<tr>
<td>[80, 80, 80]</td>
<td>1.128e-02</td>
<td>1.94</td>
<td>1.143e-03</td>
<td>1.75</td>
</tr>
</tbody>
</table>

Table 4.5: Spatial convergence for molecular surface.

Figure 4.8: Temporal convergence for molecular surface.
Figure 4.9: Slice plots of the numerical solutions on $z = 0$ plane in Example 3. (a) $t = 0.5$; (b) $t = 1$; (c) $t = 1.5$; (d) $t = 2$.

As shown in Fig. 4.9, a strong jump discontinuity is developed in the solution as time increases. To view the hidden solution in the other side, the corresponding contour plots are shown in Fig. 4.10. Without plotting the interface $\Gamma$ explicitly, these contour lines clearly show the location of $\Gamma$. Even though the solution pattern inside $\Gamma$ is unchanged, the contrast of the solutions inside and outside $\Gamma$ is variant. A strong jump discontinuity can also be sensed in the contour plot at time $t = 2$, because the contour lines now undergo a fast and sharp change near the interface.

Example 4. This example examines the performance of the proposed method on fully time-dependent jump conditions. To this end, we consider a smooth torus, which is given as the zero level set of

$$S(x, y, z) = \left(\frac{3}{2} - \sqrt{x^2 + y^2}\right)^2 + z^2 - \frac{49}{100}$$

Figure 4.10: Contour plots of the numerical solutions on $z = 0$ plane in Example 3. (a) $t = 0.5$; (b) $t = 1$; (c) $t = 1.5$; (d) $t = 2$.

The analytical solution is constructed in the form of

$$
u(x, y, z, t) = \begin{cases} 
\cos(ax) \sin(ay) \cos(az) \cos(t) - 1, & \text{in } \Omega^- \\
\sin(ax) \cos(ay) \sin(az) \cos(t) + 1, & \text{in } \Omega^+ 
\end{cases} \quad (4.9)$$

so that $a = 2$ yields a periodic solution with a period of $\pi$. The source term is given by

$$f(x, y, z, t) = \begin{cases} 
(3\alpha^- a^2 \cos(t) - \sin(t)) \cos(ax) \sin(ay) \cos(az), & \text{in } \Omega^- \\
(3\alpha^+ a^2 \cos(t) - \sin(t)) \sin(ax) \cos(ay) \sin(az), & \text{in } \Omega^+ 
\end{cases} \quad (4.10)$$

In this example, the interface jump conditions are time-and-space dependent, which represents the most general physical jump conditions for parabolic interface problems.

The 3D matched ADI method is stable for this example, and the results obtained by the
spatial convergence tests are presented in Table 4.6. It can be observed that the convergence rate in the $L_2$ norm is still two. However, the $L_\infty$ convergence becomes a little deteriorated in this example, especially for the last entry, even though the overall rate is still about 1.6. The temporal tests are shown in Figure 4.11 which reveals a similar pattern, i.e., the $L_2$ order is similar to the previous examples, while the $L_\infty$ rate is affected somehow. For more detailed insights, the color maps are demonstrated in Figure 4.12. We can see that the error distribution pattern is quite random in this example, which means that some large errors occur on or near the interface $\Gamma$, even when the solution or its gradient is not large. The $L_\infty$ convergence patterns shall be related to such

<table>
<thead>
<tr>
<th>$[N_x, N_y, N_z]$</th>
<th>( L_\infty )</th>
<th>Order</th>
<th>( L_2 )</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>[20, 20, 20]</td>
<td>2.774e-01</td>
<td>2.774e-01</td>
<td>1.908e-02</td>
<td>2.79</td>
</tr>
<tr>
<td>[40, 40, 40]</td>
<td>3.803e-02</td>
<td>2.87</td>
<td>2.767e-03</td>
<td>2.79</td>
</tr>
<tr>
<td>[80, 80, 80]</td>
<td>1.146e-02</td>
<td>1.73</td>
<td>6.554e-04</td>
<td>2.08</td>
</tr>
<tr>
<td>[160, 160, 160]</td>
<td>8.254e-03</td>
<td>0.47</td>
<td>1.602e-04</td>
<td>2.03</td>
</tr>
</tbody>
</table>

Table 4.6: Spatial convergence for torus surface.

Figure 4.11: Temporal convergence for torus surface.
an irregular error distribution.

![Color maps for torus surface. (a) numerical solution; (b) numerical error.](image)

**Example 5.** This example deals with a complicated topological shape of a tanglecube, defined as the zero level set of

\[ S(x, y, z) = x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 10. \]  

(4.11)

embedded in a domain of dimension \([-3.99, 3.99] \times [-3.99, 3.99] \times [-4.99, 4.99]\). The tanglecube surface has several convex and concave regions which pose grant challenges for the tangential derivative approximations. The previous analytical solution (4.9) is reused with a different coefficient \(a = 1\) so that all jump conditions are still time dependent. For this example, one grid line will cut the interface in zero, two or four times.

The matched ADI method is stable for this example. Similar spatial and temporal tests are repeated and the results are shown in Table 4.7 and Figure 4.13, respectively, together with the color maps shown in Figure 4.14. Even though the convergence is not uniform, the averaged spatial convergence rates in both error norms are above two. The temporal convergence rate is similar to what obtained in previous examples. Moreover, the color maps show that the maximum errors occur not only in the regions where the solution achieves maximum values, but also in the regions where the largest jumps take place across the interface.

We also examine the efficiency of the 3D matched ADI method for this example in two aspects, CPU time and memory usage. For the CPU time, the complexity of the proposed method is on the
order of $O(N)$ with $N = N_x \times N_y \times N_z$ per time step. In Figure 4.15(a), the CPU time is plotted against $N$ for evolving 1000 time steps. The mesh grid per direction is chosen from 20 to 220 with an increment of 10. Computationally, the total number of interface points which require special interface treatments is different for different mesh size. However, the interface treatments are conducted only on irregular nodes near the surface. Moreover, they needs to be done only once before the time integration. Thus, the CPU time shown in Figure 4.15(a) is still dominated by the 3D ADI computation, not the MIB interface treatments. Consequently, the execution time scales linearly against $N$ in Figure 4.15(a). A least-square fitting in log-log scale is conducted.
with a slope $r = 0.824$. On the other hand, the memory usage mainly consists of two parts, the structure which stores the interface informations and the sparse matrices for computation at each time step. Obviously, the denser the meshes are, the more interface locations are constructed. Moreover, all discretization matrices are perturbed tridiagonal ones. We only need to store the non-zero elements of these sparse matrices in our computation. Accordingly, the overall storage for both interface locations and ADI matrices increases linearly with respect to the mesh size $N$, as shown in Figure 4.15(b) with slope $r = 0.868$. The present study demonstrates that the ADI method is an extremely efficient non-iterative algebraic solver for parabolic problems.

**Example 6.** The last example considers a heart-shaped interface, which is the zero level set of

$$S(x, y, z) = (x^2 + \frac{9}{4}y^2 + z^2 - 1)^3 - x^2z^3 - \frac{9}{80}y^2z^3$$

(4.12)

embedded in the domain of dimension $[-3.99, 3.99] \times [-1.99, 1.99] \times [-3.99, 3.99]$. The analytical solution is constructed by the formula

$$u(x, y, z, t) = \begin{cases} 
(cos(ax) + sin(ay) + cos(az)) \sin(t), & \text{in } \Omega^- \\
(sin(ax) + cos(ay) + sin(az)) \sin(t), & \text{in } \Omega^+
\end{cases}$$

(4.13)

![Figure 4.14: Color maps for tanglecube surface. (a). numerical solution; (b). numerical error.](image)
where \( a = 2 \) in this example. The source term is given by

\[
f(x, y, z, t) = \begin{cases} 
\alpha^- a^2 (\cos(ax) + \sin(ay) + \cos(az)) \sin(t) + \\
\quad (\cos(ax) + \sin(ay) + \cos(az)) \cos(t), & \text{in } \Omega^- \\
\alpha^+ a^2 (\sin(ax) + \cos(ay) + \sin(az)) \sin(t) + \\
\quad (\sin(ax) + \cos(ay) + \sin(az)) \cos(t), & \text{in } \Omega^+ 
\end{cases}
\]

The significance of this example is that its interface has two singularities, one on the top and the other at the bottom, as shown in Figure 4.16. The singularities have not been considered in previous examples, and it is of great interest to see how well the proposed matched ADI scheme performs on this example. Due to the proper corner treatment, it is found that the singularities on the interface has little impact on the spatial convergence rate, as shown in Table 4.8. The spatial convergence rate is close to 2 in the \( L_2 \) norm, and is slightly reduced for the \( L_\infty \) norm. The temporal convergence test in Figure 4.17 indicates that the temporal order is still higher than one.

The CPU time and memory usage for this example are reported in Figure 4.18, with the same setting as in Example 5. Again both of them are found to be linearly scaled with a slope rate \( r = 0.934 \) and \( r = 0.925 \). The 3D matched ADI method is founded to be stable for the present example with singularities, which is demonstrated in Figure 4.19. By considering a very large
Figure 4.16: Color maps for heart surface. (a). numerical solution; (b). numerical error.

<table>
<thead>
<tr>
<th>([N_x, N_y, N_z])</th>
<th>(L_\infty)</th>
<th>Order</th>
<th>(L_2)</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>[40, 40, 40]</td>
<td>4.490e-02  1.70</td>
<td></td>
<td>8.364e-03  2.09</td>
<td></td>
</tr>
<tr>
<td>[80, 80, 80]</td>
<td>2.158e-02  1.06</td>
<td></td>
<td>2.073e-03  2.01</td>
<td></td>
</tr>
<tr>
<td>[160, 160, 160]</td>
<td>5.701e-03  1.92</td>
<td></td>
<td>5.186e-04  2.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.8: Spatial convergence for heart surface.

time step \(\Delta t = 10\), we calculate \(N_x = N_y = N_z = 20, 40, 80, 160\) in each direction and choose the stopping time varying from \(10^3\) to \(10^5\). It can be seen that the errors do not grow with respect to the time for all mesh sizes. Great stability has been observed in all tested cases.

4.2 Comparison of time splitting methods with variable diffusion coefficients

In this section, we focus on efficiency and accuracy comparison of a total of eight methods, two spatial methods coupled with four temporal methods. Spatially, two variations of matched interface and boundary (MIB) methods are considered, to be simple, since GFM is essentially equivalent to MIB-L1 method with less accurate derived jump condition. For temporal methods, we consider all the four methods Douglas-Rachford alternating direction implicit (ADI) method, locally one-dimensional implicit Euler (LOD-IE), locally one-dimensional Crank-Nicolson (LOD-CN) and Trapezoidal Splitting (TS) method in chapter 3. We coupled the two spacial
methods and the four temporal methods altogether in the experiments. Consequently, to be simple we call these eight methods ADI-L1, ADI-L2, LODIE-L1, LODIE-L2, LODCN-L1, LODCN-L2, TS-L1 and TS-L2. Only variable diffusion coefficients is considered in the experiments since piecewise constant diffusion coefficients is just a special case. The code is written in C++ and compiled with GCC 4.8.5 compiler and C++11 standard, “vector” in C++ STL has been used as the major data structure taking advantage of the guaranteed contiguous memory allocated.
Reported CPU time is recorded on the master node of SGI Ultraviolet in Alabama Supercomputer Center with a single Xeon E5-4640 CPU core operating at 2.4 GHz clock speed.

Three types of variable coefficients are taken into consideration, different functions have different properties. It would be better to test the behavior between different algorithms based on distinct analytical solutions, shapes and types of variable coefficients. The first type of variable coefficient is rational function,

$$
\alpha(x, y, z) = \begin{cases} 
\frac{1}{x^2 + y^2 + z^2 + 1} + 1, & \text{in } \Omega^- \\
\frac{1}{x^2 + y^2 + z^2 + 1}, & \text{in } \Omega^+ 
\end{cases}
$$

(4.15)

The second type of variable coefficient is trigonometric function,

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

(4.16)
The third type of variable coefficient is Gaussian function

\[
\alpha(x, y, z) = \begin{cases} 
    e^{\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} + \frac{z^2}{\sigma_z^2}}, & \text{in } \Omega^- \\
    e^{\frac{x^2}{\overline{\sigma}_x^2} + \frac{y^2}{\overline{\sigma}_y^2} + \frac{z^2}{\overline{\sigma}_z^2}}, & \text{in } \Omega^+ 
\end{cases} 
\]  

(4.17)

with amplitude as constant 1, and variance \( \mu = 0 \), standard deviation \( \sigma_x = \sigma_y = \sigma_z = 3 \) in the region \( \Omega^- \) and \( \overline{\sigma}_x = \overline{\sigma}_y = \overline{\sigma}_z = 4 \) in the region \( \Omega^+ \) in all three directions.

**Example 7.** Consider a simple sphere surface

\[
S(x, y, z) = x^2 + y^2 + z^2 - 1. 
\]  

(4.18)

with analytical solution (4.4) with \( a = 2 \). The source term depends on the first type of variable coefficient which is a pair of rational functions from (4.15) based on the grid locations. The zeroth order jump condition on the interface is kept as time-dependent and the first order jump condition is time-independent. The computational domain is fixed to be 

\([-3.99, 3.99] \times [-1.99, 1.99] \times [-1.99, 1.99].\)
A efficiency comparison is performed in Figure 4.20, the cost of the most expensive algorithm are about two times expensive than the cheapest algorithm. Not surprisingly, TS-L1 and TS-L2 methods are the most expensive one due to more operations in Trapezoidal splitting algorithm. Meanwhile, it is easy to tell that LODIE-L1 and LODIE-L2 are the cheapest
Figure 4.23: Spatial $L_\infty$ error for sphere interface with rational type variable coefficient.

Figure 4.24: Spatial $L_2$ error for sphere interface with rational type variable coefficient.

algorithms. For temporal numerical error, regards both $L_2$ and $L_\infty$, in Figure 4.21 and 4.22, the spacial size are fixed at 160 in all three x, y, z directions. All methods are stable with large time
step except TS-L2, as we mentioned in previous chapter, the scheme of Trapezoidal splitting method used in this work has been modified at the fifth step, it could be a potential reason of the instability. However, TS-L1 is stable for all different time steps, which implies that different MIB scheme could also generate some instability issue based on some specific temporal method. A variation of MIB method leads to a different matrix structure when solving the linear system which might essentially affect the stability. Other than the stability issue with TS-L2, almost all methods stop perform first order convergence rate after some time step. The variable diffusion coefficient somehow restrict the performance of first order convergence no matter what temporal algorithm are adopted. Spatially, in Figure 4.23 and 4.24, time step are picked at \(\Delta t = 1e^{-4}\), with grid size vary from 20 to 160, none of \(L_2\) and \(L_{\infty}\) can perform the overall 2nd order convergence rate in space. The big difference between different algorithms at the beginning is that all methods using MIB-L2 scheme performs slightly more accurate than the methods using MIB-L1 scheme. As grid mesh become finer and finer, the gap are narrowed down expeditiously. For \(L_2\) numerical error, with the finest mesh grid 160, the interesting thing is, all methods using MIB-L1 scheme are converged to a magnitude of \(\Delta t = 1e^{-4}\) and the methods using MIB-L2 scheme only converge to a magnitude of \(\Delta t = 1e^{-3}\).

**Example 8.** The simple sphere surface is still adopted from (4.18) with analytical solution (4.4) with \(a = 2\) in a computational domain \([-3.99, 3.99] \times [-1.99, 1.99] \times [-1.99, 1.99]\). A trigonometric type of variable coefficient is constructed as (4.16).

For trigonometric type of variable coefficient, the efficiency is still the same that TS the most expensive one and LODIE the cheapest one shown in Figure 4.25. For temporal numerical error in Figure 4.26 and 4.27, TS-L2 is the only unstable one, unlike rational variable coefficients, it start convergence at \(\Delta t = 8e^{-4}\) instead of \(\Delta t = 1e^{-3}\). Meanwhile, none of the eight method can perform first order of convergence rate with respect to time. For spatial numerical error in Figure 4.23 and 4.24, time step is still fixed at \(\Delta t = 1e^{-4}\) and different sizes of mesh grid are adopted, from 20 to 40, all algorithms converge with a rate 2, afterwards, none of them keep the convergence rate. Especially for \(L_2\) error, methods with MIB-L2 scheme almost only keep the
Example 9. The last example for the simple sphere interface (4.18) with analytical solution (4.4) and a computational domain $[-3.99, 3.99] \times [-1.99, 1.99] \times [-1.99, 1.99]$ is tested in this example. The variable coefficient pair is referred to the Gaussian type (4.17).
Figure 4.27: Temporal $L_2$ error for sphere interface with trigonometric type variable coefficient.

Figure 4.28: Spatial $L_\infty$ error for sphere interface with trigonometric type variable coefficient.

The last type of variable diffusion coefficient for sphere is Gaussian function. In terms of efficiency in Figure 4.30, TS methods are still the most expensive methods. The temporal numerical errors are shown in Figure 4.31 and 4.32. The two LOD methods, LODIE and LODCN
Figure 4.29: Spatial $L_2$ error for sphere interface with trigonometric type variable coefficient.

Figure 4.30: Sphere efficiency comparison with Gaussian type variable coefficient.

have a better convergence rate at the beginning, since the error obtained from the two methods are relatively large than other methods. ADI method is the most accurate method, the error converge to a smaller range and bounded by the spatial size. For spatial numerical error in Figure 4.33 and
4.34, from 40 to 160, both $L_\infty$ and $L_2$ perform the theoretical rate of convergence equal to 2 in space and take minor improvements after the 40.

**Example 10.** Besides the simple surface like sphere, a more complicated interface is taken into consideration as tanglecube we used in (4.11) in a computational domain of dimension
Figure 4.33: Spatial $L_\infty$ error for sphere interface with Gaussian type variable coefficient.

Figure 4.34: Spatial $L_2$ error for sphere interface with Gaussian type variable coefficient.

Consistent with the conclusion for simple surface, the two Trapezoidal splitting methods are the most inefficient one and the two LODIE methods are the most efficient one shown in Figure 4.35. In Figure 4.36, ADI-L2 has the largest $L_\infty$ numerical error for this complicated interface, while it is the same as simple interface, the largest $L_2$ numerical error is still obtained.
from the two LODIE methods, in the meantime, TS-L2 remained the only conditionally stable methods. From Figure 4.38 and 4.39, it’s hard to figure out the line segment with corresponding methods based on spatial numerical error comparison. The numerical results shows that all
methods with MIB-L1 scheme perform a similar numerical error for both $L_\infty$ and $L_2$ which is significantly smaller than the numerical error generated by methods with MIB-L2 scheme. We have seen some comparable performance for sphere interface as well which indicates that MIB-L1 scheme converge better than MIB-L2 scheme.

Example 11. Considering the same surface and analytical solution as previous example but replace the variable coefficient as a type (4.16).

The efficiency comparison is illustrated in Figure 4.40. With regard to temporal numerical error in Figure 4.41 and 4.42, the same effect has been noticed for this complicated interface with trigonometric type of variable coefficients, ADI-L2 delivers the most inaccurate numerical results, which implies that the complexity of surface has a larger impact on $L_\infty$ than $L_2$. As far as spatial comparison for the eight methods in Figure 4.43 and 4.44, all methods with MIB-L1 schemes performs better than methods with MIB-L2 schemes, and the methods with MIB-L2 scheme stop converge after mesh size of 40.

Example 12. The last example we considered is still tanglecube surface with analytical solution
Figure 4.40: Tanglecube efficiency comparison with trigonometric type variable coefficient.

Figure 4.41: Temporal $L_{\infty}$ error for tanglecube interface with trigonometric type variable coefficient.

(4.9), Gaussian function is studied as (4.17).

In this example, efficiency can be interpreted from Figure 4.45. In Figure 4.46 and 4.47, only TS-L2 failed to converge when time step is relatively large, and all methods start converging after time step $\Delta t = 5e - 3$. Same with sphere interface, the ADI method performs less numerical
Figure 4.42: Temporal $L_2$ error for tanglecube interface with trigonometric type variable coefficient.

Figure 4.43: Spatial $L_\infty$ error for tanglecube interface with trigonometric type variable coefficient.

error than other eventually. For Figure 4.48 and 4.49, this example keeps both characteristics of maintained by complicated interface and Gaussian type of variable coefficient. MIB-L1 scheme
Figure 4.44: Spatial $L_2$ error for tanglecube interface with trigonometric type variable coefficient.

Figure 4.45: Tanglecube efficiency comparison with Gaussian type variable coefficient.

keep performs better than MIB-L2 scheme. Same as sphere, from mesh size 20 to 40, all methods with $L_{\infty}$ and $L_2$ converge with the theoretical rate, after 40, all just take small rate of convergence.
Figure 4.46: Temporal $L_{\infty}$ error for tanglecube interface with Gaussian type variable coefficient.

Figure 4.47: Temporal $L_2$ error for tanglecube interface with Gaussian type variable coefficient.
Figure 4.48: Spatial $L_\infty$ error for tanglecube interface with Gaussian type variable coefficient.

Figure 4.49: Spatial $L_2$ error for tanglecube interface with Gaussian type variable coefficient.
CHAPTER 5
CONCLUSIONS

In this work, we propose a novel matched ADI method for solving 3D parabolic interface problems with discontinuous diffusion coefficient and complex interfaces. The newly developed scheme inherits the merits of its ancestor for solving 2D interface problems, while possesses unique features, such as the non-orthogonal local coordinate system for decoupling the jump conditions, two-side estimation of tangential derivative at the interface point, and the capability of handling arbitrary many interface points on one grid line, for solving more challenging 3D problems. Moreover, a new Douglas-Rachford ADI scheme is designed, which minimizes the perturbation terms, in compared with the implicit Euler scheme, and is of first order in time.

The proposed method is found to converge and maintains stability for various complex-shaped interfaces and time-and-space dependent jump conditions in the numerical experiments. Being highly accurate and efficient, this new ADI method provides a very promising numerical algorithm for solving 3D parabolic PDEs. In terms of accuracy, the matched ADI method yields a spatially second order of accuracy for all tested interface problems. Due to strong stability, a large time increment can be used in the ADI method for an efficient time integration. Moreover, in each time step, the ADI method provides an exact or non-iterative algebraic solver whose complexity scales linearly with respect to the spatial degree of freedom. Therefore, the matched ADI method is definitely one of the fastest implicit algorithms for solving 3D parabolic interface problems.

For variable coefficients, eight methods are proposed as ADI-L1, ADI-L2, LODIE-L1, LODIE-L2, LODCN-L1, LODCN-L2, TS-L1, TS-L2, which are named by a combination of temporal method and spatial method. Three different pairs of variable coefficient are deployed in
numerical experiment with different types of analytical solutions. For efficiency, LODIE method are found to be the most efficient one, ADI method is almost same efficient as LODIE, LODCN method is much more expensive than the previous two methods and TS method is the most expensive one. Meanwhile, in spatial, MIB-L1 scheme is found slightly more efficient than MIBL2 method but with a little sacrifice of accuracy. For accuracy, it's interesting to say that there is not much difference between two spatial scheme MIB-L1 and MIB-L2 scheme. In terms of temporal methods, ADI method and TS method are much better than LODIE method and LODCN method. However, TS method is the only method that considered unstable with large time step, specially, TS method with MIB-L2 scheme, all other methods are stable with variable coefficient and various time step and grid size.

Overall, for solving parabolic interface problem, for the sake of its unconditional stability, high accuracy and extreme efficiency, matched ADI is the first choice to be considered regardless of constant diffusion coefficient or variable diffusion coefficient.
REFERENCES


[16] Douglas Jr, J. On the numerical integration of \( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t} \) by implicit methods. *SIAM Journal on Applied Mathematics* 3, 1 (1955), 42.


79


[70] Zhao, S., and Wei, G. Matched interface and boundary (mib) for the implementation of boundary conditions in high-order central finite differences. *International journal for numerical methods in engineering* 77, 12 (2009), 1690–1730.


