SONG, PENG. Adaptive Mesh Refinement Strategies for the Immersed Interface Method. (Under the direction of Zhilin Li.)

Adaptive mesh refinement strategies are proposed for the immersed interface method. The immersed interface method (IIM) was developed to solve partial differential equations involving interfaces or discontinuities. The IIM modifies the standard finite difference schemes around the interface by using jump conditions to improve the accuracy. The adaptive mesh refinement (AMR) technique is to deploy a high mesh resolution only around the interface where it is most needed, whereas use a comparatively coarse resolution where the solution is smooth enough. The AMR utilizes the computational resources more economically without loss of accuracy, compared to the mesh refined uniformly. We developed an adaptive version of the IIM to take advantages of both: the numerical accuracy of the IIM and the computational efficiency of AMR. Moreover, we designed the AMR-IIM based on the popular level set method to solve many interface problems.

In this dissertation, we developed a brand new mesh generation method, coupled with an efficient data structure for the AMR-IIM. The local mesh refinement is generated from a narrow tube $|\varphi(x, y, t)| \leq \delta$ around the interface. The linear system of equations derived from the finite difference discretization is solved by an algebraic multigrid solver. The AMR-IIM has been successfully applied to elliptic partial differential equations, Stokes equations, Navier-Stokes equations with fixed or moving interfaces. Numerical results show that the AMR-IIM inherits second order accuracy of the previous IIM on a uniform mesh, and runs faster since the size of linear system of equations is reduced by more than 50%. Implementation details and plenty of numerical experiments are also presented in the dissertation.
Adaptive Mesh Refinement Strategies for the Immersed Interface Method

by

Peng Song

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research

Raleigh, North Carolina

2013

APPROVED BY:

Kazufumi Ito

Negash Medhin

Huixia (Judy) Wang

Zhilin Li
Chair of Advisory Committee
DEDICATION

To my parents
Lisuo Song and Jianhua Zhang
for their endless love.
BIOGRAPHY

Peng Song was born in the city of Tianjin, China in 1985. He became very interested in mathematics when he studied at Xinhua Middle School from 1998 and Nankai High School from 2001. In 2004, he was enrolled to Nankai University in Tianjin, where he received his Bachelor degree in Statistics from the School of Mathematical Sciences in 2008. After that, he came to US and joined the Operations Research Graduate Program in NC State University at Raleigh. He started to do research in numerical mathematics with Dr. Zhilin Li since 2009. Meanwhile, he worked as teaching assistant for Department of Mathematics in NC State University. He received his Master degree in Operations Research and Minor in statistics in 2010. Peng enjoys using mathematics knowledge to solve application problems. He has some internship experiences in US Environmental Protection Agency about statistical models and in SAS Institute Inc. about analytic software testing. In January 2013, he joined the Bank of America as a quantitative operations associate for the development work of risk modeling tools.
ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Dr. Zhilin Li for his guidance and support throughout my PhD study. I benefit so much from the insightful discussions with him. He taught me not only the cutting-edge knowledge but also the correct attitude to be a great researcher. I will never forget his help and encouragement when I was in challenging times in my study and my life.

I would also like to thank my committee members, Dr. Kazufumi Ito, Dr. Negash Medhin, and Dr. Huixia Judy Wang for reviewing my thesis and offering valuable suggestions. Thanks to Ms. Linda Smith for her kind help during my five years in Operations Research program.

Thanks also go to my dear friends in NC State University for their help and collaboration on my study: Sidong Zhang, Guanyu Chen, Ming Jiang, Robert Shih, Shijun Yin, Ran Liu, Yue Li, Ye Tian, Jingyu Zhang, Tao Hong, Song Yang, Yuanhui Zhang, Rajneesh, Alex Gose, Benjamin Perryman, Weining Shen and Dehan Kong.
# TABLE OF CONTENTS

**LIST OF TABLES** ........................................... vii

**LIST OF FIGURES** ........................................... viii

Chapter 1 Introduction ........................................ 1
  1.1 Interface problems and the mathematical models .............. 1
     1.1.1 Some examples of interface problems .................... 1
     1.1.2 PDEs with interfaces ..................................... 3
  1.2 Motivations of the adaptive mesh refinement strategy .......... 5
  1.3 Literature Review .......................................... 8
     1.3.1 Finite difference methods and Cartesian grid methods .... 8
     1.3.2 Review on numerical algorithms for interface problems .... 9
     1.3.3 Review on adaptive mesh refinement techniques .......... 13
     1.3.4 Review on the level set method .......................... 15
  1.4 Contributions and an outline of the dissertation ............. 16

Chapter 2 The AMR-IIM for Elliptic PDEs with interfaces .......... 19
  2.1 The adaptive mesh generation procedure ...................... 20
  2.2 The data structure of the AMR ............................... 25
  2.3 Finite difference schemes on the adaptive meshes ............ 27
  2.4 The Algebraic Multigrid solver .............................. 31
  2.5 Numerical experiments and analysis .......................... 34
     2.5.1 A benchmark elliptic interface problem: AMR-IB vs AMR-IIM . 35
     2.5.2 An elliptic PDE with a variable and discontinuous coefficient . 39
     2.5.3 An elliptic PDE with a discontinuous solution ............ 41
     2.5.4 An elliptic PDE with an elliptic interface ................ 43
     2.5.5 An elliptic PDE with a piecewise constant coefficient ...... 44
     2.5.6 An elliptic PDE with a complicated interface ............ 46

Chapter 3 The AMR-IIM for the Stokes equations with interfaces .... 50
  3.1 The three-Poisson approach for the Stokes equations .......... 52
  3.2 The level set method and interface evolution techniques ....... 56
     3.2.1 The interface evolution by the level set method ........... 57
     3.2.2 Mass conservation techniques ............................ 59
  3.3 Coupling the AMR with the level set method ................... 61
  3.4 Numerical experiments and analysis .......................... 65
     3.4.1 The Stokes equations with a fixed interface ............. 65
     3.4.2 The surface tension problem ............................ 68
     3.4.3 The bubble creeping problem ............................ 74
Chapter 4 The AMR-IIM for the Navier-Stokes equations with interfaces 80
  4.1 The projection-IIM for the Navier-Stokes equations with interfaces . . . . 82
    4.1.1 The projection method for the Navier-Stokes equations . . . . . . 82
    4.1.2 Jump relations and correction terms using the IIM . . . . . . . . 83
  4.2 Implement the projection-IIM method on the AMR mesh . . . . . . . . 87
  4.3 Numerical experiments and analysis . . . . . . . . . . . . . . . . . . . . 90
    4.3.1 A circular flow with a fixed interface . . . . . . . . . . . . . . . 90
    4.3.2 The surface tension problem revisit by the Navier-Stokes equations 96

Chapter 5 Conclusion and future work .............................. 102

References ............................................................... 106
## LIST OF TABLES

Table 2.1 A comparison of grid refinement analysis using Uniform-IB and AMR-IB with the same finest resolution $h$.

Table 2.2 Two and four levels of AMR for the IB method.

Table 2.3 A comparison of grid refinement analysis using Uniform-IIM and AMR-IIM with the same finest resolution $h$. The CPU times spent in algebraic multigrid solver are also recorded.

Table 2.4 Number of grid points (unknowns) to satisfy $\|E_u\|_\infty < \epsilon$ by Uniform-IIM and various AMR-IIM.

Table 2.5 A comparison of grid refinement analysis of Example 2 using Uniform-IIM and AMR-IIM with the same finest resolution $h$.

Table 2.6 A comparison of grid refinement analysis of Example 3 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

Table 2.7 A comparison of grid refinement analysis of Example 4 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

Table 2.8 A comparison of grid refinement analysis of Example 5 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

Table 2.9 A comparison of grid refinement analysis of Example 6 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

Table 3.1 A grid refinement analysis by uniform mesh for Stokes equations with a fixed interface.

Table 3.2 A grid refinement analysis by adaptive mesh for Stokes equations with a fixed interface.

Table 3.3 Critical physical quantities solved by AMR 80(+1)[2] at the four moments corresponding to Figure 3.8 (a)-(d).

Table 3.4 The errors of $p$, $u$, $v$ computed by AMR $m(+1)[2]$ compared to the benchmarks computed by AMR 320(+1)[2] at $t = 0$.

Table 4.1 Case 1, $h(t) = 1$, $\mu = 0.02$, $t = 4s$.

Table 4.2 Case 2, $h(t) = e^{-t}$, $\mu = 0.02$, $t = 4s$.

Table 4.3 Case 3a, $h(t) = \sin(t)$, $\mu = 0.5$, $t = 4s$.

Table 4.4 Case 3b, $h(t) = \sin(t)$, $\mu = 0.005$, $t = 4s$.  
LIST OF FIGURES

Figure 1.1  (a) A star-shaped interface (red) in fluid relaxes to a circular equilibrium shape (blue) due to the surface tension. (b) A bubble creeping in fluid with deformation due to the gravity. ........................................... 2

Figure 1.2  A diagram of a rectangular domain $\Omega = \Omega^+ \bigcup \Omega^-$ with an interface $\Gamma$. The coefficients $\beta(x,y)$ could have a finite jump across the interface $\Gamma$. ................................................................. 3

Figure 1.3  Refine the domain including interface (red) uniformly to reduce solution error at the cost of increasing numbers of grid points: (a) A uniform $20 \times 20$ mesh. (b) A uniform $40 \times 40$ mesh. (c) A uniform $80 \times 80$ mesh. ................................................................. 6

Figure 1.4  Refine the Uniform 20 mesh (a) locally around interface (red) by adding one level AMR (b) and adding two levels of AMR (c) to improve solution accuracy. ................................................................. 7

Figure 1.5  Release the unnecessary grid points far away from interface (red) to reduce computational cost without loss of accuracy. (a), (b), (c) have the same finest mesh resolutions around interface ........... 8

Figure 1.6  Two typical discrete delta functions (a) Discrete hat delta function (b) Discrete cosine delta function ................................................................. 11

Figure 1.7  Roma’s AMR(left) and grid patches(right) ................................................................. 14

Figure 2.1  Adaptive mesh around a circular interface (red solid). Parent grid points (starred) are selected within the tube $|\varphi_{ij}| = |\varphi(x_i, y_j)| \leq h$ (red dashed). Then, square meshes in finer resolution are generated from parent grid points. The refinement ratio $r = 2$, and the refinement tube width $\lambda = 1$ ................................................................. 21

Figure 2.2  Flow chart of a three-level mesh generation and indexing procedure. .................... 23

Figure 2.3  Adding two levels of AMR around circle interface: AMR(+2)[2], with refinement tube width $\lambda = 2$. ................................................................. 24

Figure 2.4  The data structure to book-keep the information of two levels grid points. ................................................................. 26

Figure 2.5  A portion of an adaptive mesh that includes the fine level mesh around an interface (red curve) and the coarse mesh. Ghost points $g_1$ and $g_2$ are used to derive the finite difference equation for node 12. .... 28

Figure 2.6  The solution plot of Example 1 ................................................................. 36

Figure 2.7  The solution plot of Example 2 ................................................................. 40

Figure 2.8  The solution plot of Example 3 ................................................................. 42

Figure 2.9  The solution plot of Example 4 ................................................................. 44
Figure 2.10 The solution plot of Example 5 ............................................. 45
Figure 2.11 The solution plot of Example 6 ............................................. 47
Figure 2.12 Use AMR 80(+1)[2] to cover the flower-shape interface (red curve) 48
Figure 2.13 CPU time distribution in the process of solving an elliptic interface problem by the AMR-IIM ................................................................. 49

Figure 3.1 (a) The local coordinate $\xi$-$\eta$ defined by interface normal direction and tangential direction, where $\theta$ is the angle between the $x$-axis and the normal directions. (b) The singular force $(f_1, f_2)$ in horizontal and vertical directions are transformed to $(\hat{f}_1, \hat{f}_2)$ in normal and tangential directions. .................................................. 53

Figure 3.2 A part of an adaptive mesh including the interface (red curve), the fine level mesh, and the coarse mesh. The $p_y$ at hanging point 12 can be approximated by the average of $p_y$ at 11 and $p_y$ at 13. The $p_y$ at 12 can also be computed after ghost points $g1$ and $g2$ are interpolated. ................................................................. 56

Figure 3.3 Expand the interface by $\epsilon$ in the normal direction to conserve the enclosed area against the initial area $A_0$. ................................. 60

Figure 3.4 The interface (orange dashed), the level set update tube (red), the adaptive refinement tube (blue) and the level set re-initialization tube (green) for moving interface problems. $h$ is the finest the mesh resolution. ................................................................. 63

Figure 3.5 The flow chart of solving Stokes equations on AMR for moving interface problems. ................................................................. 63

Figure 3.6 Pressure color profile ................................................................. 69

Figure 3.7 Velocity field ................................................................. 70

Figure 3.8 Snap-shots of a deformable interface (red-solid in the middle of the fine mesh) relaxing to its equilibrium state computed by the AMR mesh 80(+1)[2]. ................................................................. 71

Figure 3.9 Snap-shots of a deformable interface (red-solid in the middle of the fine mesh) relaxing to its equilibrium state computed by the AMR mesh 40(+2)[2]. ................................................................. 73

Figure 3.10 Creeping bubbles with various surface tension $\gamma$ at time 0s, 0.60s, 1.1s, 1.6s, 2.2s, 2.8s. The loss of mass is less than 0.01% in all four cases. ................................................................. 76

Figure 3.11 A rising bubble (the red solid curve in the middle of the fine mesh) at time $t = 0s, 0.720s, 1.30s, 2.28s, 3.64s$, computed by AMR 80(+1)[2] ................................................................. 77

Figure 3.12 A rising bubble (the red solid curve in the middle of the fine mesh) at time $t = 0s, 0.720s, 1.30s, 2.28s, 3.64s$, computed by AMR 80(+2)[2] ................................................................. 78
Figure 3.13 The bubble interface at $t = 3s$ computed by the uniform mesh $160 \times 160$ and the AMR $80(+1)[2]$.  

Figure 4.1 The local coordinate $\xi-\eta$ defined at $x^*$ on the interface (red). The correction term for grid point $x$ can be derived from the jump conditions at $x^*$.  

Figure 4.2 A part of an adaptive mesh that includes the fine level mesh around an interface (red curve) and the coarse mesh. For hanging node 12, we need to interpolate ghost points $g1$ and $g2$ first, and then plug in the interpolated $g2$ to compute $\nabla_h u, \nabla_h p$ and $\Delta_h u$ for point 12.  

Figure 4.3 The velocity field of circular flow when $h(t) = 1$ with circular interface.  

Figure 4.4 The evolution of the semi-major axes $r_a$ (red) and the semi-minor axes $r_b$ (blue) over time in different viscosity $\mu$ and surface tension coefficient $\gamma$.  

Figure 4.5 Snapshots of the oscillating interface (red solid) around its equilibrium status (blue dashed) when the viscosity $\mu = 0.02$, and the surface tension coefficient $\gamma = 1$.  

Figure 4.6 Snapshots of the oscillating interface (red solid) around its equilibrium status (blue dashed) when the viscosity $\mu = 0.02$, and the surface tension coefficient $\gamma = 1$.  

Chapter 1

Introduction

1.1 Interface problems and the mathematical models

1.1.1 Some examples of interface problems

Interface problems have many applications in our life. For example, when there are two different materials, such as water and oil, or the same material but at different states, such as water and ice, we are dealing with an interface problem. One typical interface problem is the heat propagation in two materials with different heat conductivities. Another example is the growth of crystal which can be tracked by the moving interface between solid state and liquid state.

In this dissertation, we focus on the interface problems in computational fluid dynamics (CFD). In Chapter 3 and 4, we will study an interface immersed in fluid and its motion driven by its surface tension. The distorted interface in a star shape will relax to its equilibrium state, a circle with the minimum circumference, see Figure 1.1(a). Another interesting interface problem is the simulation of a bubble creeping in fluid. The motion
of the bubble can be tracked by the interface between fluid and air, or two different fluids. We study the bubble rising and deformation problem in Chapter 3 where the only driven forces are the gravity and the surface tension, see Figure 1.1(b).

![Figure 1.1: (a) A star-shaped interface (red) in fluid relaxes to a circular equilibrium shape (blue) due to the surface tension. (b) A bubble creeping in fluid with deformation due to the gravity.](image)

Above problems can be used in some mathematical models in biology. For example, the surface tension problem can be used to simulate cell aggregation in the study of tissue formation and interaction [70]. The creeping bubble problem can be converted to a red blood cell model if we modify the problem settings according to physics in human blood vessels.

Interface problems have many other applications besides computational fluid dynamics, such as computer vision, computer-aided design, biophysics and material science. We refer interested readers to read books [39, 52, 60] for more details.
1.1.2 PDEs with interfaces

In this dissertation, we study elliptic PDEs, Stokes equations and Navier-Stokes equations in a 2D rectangular domain with an interface $\Gamma$, see Figure 1.2 for an illustration. The examples we listed in Section 1.1.1 can be solved by Stokes or Navier-Stokes equations, both of which are based on the solution of elliptic equations.

![Figure 1.2: A diagram of a rectangular domain $\Omega = \Omega^+ \cup \Omega^-$ with an interface $\Gamma$. The coefficients $\beta(x, y)$ could have a finite jump across the interface $\Gamma$.](image)

Elliptic PDEs with interfaces

In Chapter 1, we study the elliptic interface problems in the form of

$$\nabla \cdot (\beta \nabla u) = f(x), \quad x \in R,$$  \hspace{1cm} (1.1)
together with jump conditions across interface $\Gamma$

$$[u]_\Gamma = w(s), \quad \left[ \frac{\partial u}{\partial n} \right]_\Gamma = v(s), \quad (1.2)$$

and boundary conditions of $u(x)$ along $\partial R$, where $R$ is a rectangular domain, $\Gamma \in C^2$ is a smooth interface parameterized by a one-dimensional variable $s$, say the arc-length, within the domain $R$; $w(s) \in C^2$ and $v(s) \in C^1$ are two functions defined along the interface $\Gamma$. Note that, when $w(s) = 0$, the problem can be written as

$$\nabla \cdot (\beta \nabla u) = f(x) + \int_\Gamma v(X(s)) \delta(x - X(s)) \, ds, \quad x \in R, \quad (1.3)$$

where $X(s)$ is a point on the interface $\Gamma$.

The elliptic interface problem is the foundation of our study, because both Stokes and Navier-Stokes equation problems can be reduced to solving some elliptic equations.

**The Stokes equations with interfaces**

In Chapter 3, we study the Stokes equations in 2D for an incompressible flow with an immersed interface:

$$\begin{cases}
\nabla p = \mu \Delta u + g + \int_\Gamma f(X(s,t)) \delta(x - X(s,t)) \, ds \\
\nabla \cdot u = 0,
\end{cases} \quad (1.4)$$

in a rectangular domain $R$ involving an interface $\Gamma$ with a known boundary condition on $\partial R$. $u$ is the fluid velocity, $p$ is the fluid pressure, $\mu$ is the fluid viscosity, $g$ is an external force, and $f(X(s,t))$ is a source strength of the force defined only on the interface.
For moving interface problems, we also have

\[ \frac{d\Gamma}{dt} = u(X(s, t)), \]  

(1.5)

The Navier-Stokes equations with interfaces

In Chapter 4, we consider the full Navier-Stokes equations in 2D for an incompressible flow with an immersed interface \( \Gamma \):

\[
\begin{cases}
\rho \left( \frac{\partial u}{\partial t} + (u \cdot \nabla)u \right) + \nabla p = \mu \Delta u + g + \int_{\Gamma} f(X(s, t)) \delta(x - X(s, t)) \, ds \\
\nabla \cdot u = 0
\end{cases}
\]  

(1.6)

in a rectangular domain \( R \) including the interface \( \Gamma \) with a known boundary condition on \( \partial R \). \( u \) is the fluid velocity, \( \rho \) is the fluid density, \( p \) is the fluid pressure, \( \mu \) is the fluid viscosity, \( g \) is an external force, and \( f(X(s, t)) \) is a source strength of the force defined only on the interface.

For moving interface problem, we also have

\[ \frac{d\Gamma}{dt} = u(X(s, t)), \]  

(1.7)

1.2 Motivations of the adaptive mesh refinement s-strategy

There are several numerical methods for above interface problems, such as the immersed boundary method (IB) and the immersed interface method (IIM) described in Section 1.3.2. They are both based on the finite difference method (FD) for numerical solutions
of partial differential equations. In a FD method, we first build a rectangular mesh on the domain, and then rewrite the original differential equation as a discretized finite difference equation for each grid point, and finally convert the original problem to a linear system to approximate the solution.

In a FD method, we usually refine the mesh resolution \( h \), so that the local truncation error of finite difference scheme is reduced at every grid point, and therefore the global solution error is also reduced. For example, the solution error \( E \sim O(h) \) for a first order accurate FD method and \( E \sim O(h^2) \) for a second order accurate FD method. For simplicity, we usually refine the whole domain uniformly to reduce the solution error. The cost is to increase numbers of grid points significantly, see Figure 1.3.

![Figure 1.3: Refine the domain including interface (red) uniformly to reduce solution error at the cost of increasing numbers of grid points: (a) A uniform 20 \( \times \) 20 mesh. (b) A uniform 40 \( \times \) 40 mesh. (c) A uniform 80 \( \times \) 80 mesh.](image)

The interface problems are a little bit different. The FD local truncation errors around the interface are much larger than those on smooth region far away from interface. Thus, we can refine the mesh only around the interface where the solution is discontinuous or non-smooth, and keep the coarse mesh where the solution is smooth enough, see Figure 1.4 for examples. Oftentimes, the main interest of the problem is only in the neighborhood
of the interface, such as the crystal growth problem and the bubble creeping problem. That is the motivation of the adaptive mesh refinement (AMR), which can automatically track the interface and refine the mesh locally around interface.

![Grids with 441, 973, and 2037 grid points](image)

Figure 1.4: Refine the Uniform 20 mesh (a) locally around interface (red) by adding one level AMR (b) and adding two levels of AMR (c) to improve solution accuracy.

Besides, we can also save some computational cost by AMR for the interface problems already well solved by uniformly refined mesh, see Figure 1.5 for example. We expect to use AMR to save the unnecessary computation on the fine mesh far away from interface, without loss of accuracy.

It can been learned from above that the motivation of the AMR is to focus on the interface and distribute the computation resources more economically. The AMR can be used to improve the accuracy by refining mesh locally, or to reduce the unnecessary computation cost without loss of accuracy. This advantage would become more significant for some large problem where it is impossible to refine the whole domain, such as some computational models in cosmology.

The objective of this dissertation is to develop an adaptive mesh refinement version of the immersed interface method for higher solution accuracy or lower computation cost.
1.3 Literature Review

1.3.1 Finite difference methods and Cartesian grid methods

The differential equations listed in Section 1.1.2 are very difficult or even impossible to solve analytically. The finite difference (FD) method is used to approximate the solutions to differential equations. The FD method replaces the derivatives in differential equations with finite difference approximations. The FD schemes approximate the derivatives of function by the function values on discrete points with required accuracy. The Taylor expansion is the basic tool to derive FD schemes and analyze errors. By the FD method, we convert the original differential equation to a large but finite systems of linear equations to be solved by computers. See book [31] for more details about the FD method.

The FD method is usually used on uniform Cartesian grids. The domain is discretized by uniform mesh composed of many unit squares in equal size. There are several advantages of uniform mesh. It is very easy to generate the uniform Cartesian grids and there is no additional cost to index and search the desired grid points. Moreover, it is not necessary to generate or update uniform Cartesian grids in moving interface problems.
Another advantage is that there are many well-developed solvers/packages designed for uniform Cartesian grids, so that the linear systems derived from the FD method can be solved very fast. Some examples include the Fast Poisson Solver [3], dmgd9v multigrid solver [73], the level set method library (LSMLIB) [12], and Clawpack for hyperbolic equations [30] etc.

On the other hand, the uniform mesh may not be efficient for some problems that require high resolution locally in neighborhood of the interface, which is the main interest of many applications. In those cases, the mesh with adaptive refinement is preferred to distribute computation resource more economically. We can use a multi-level mesh with high resolution locally according to the accuracy requirement. And we expect the refinement region can track the moving of interface automatically without much more additional mesh generation cost. There are several challenges of adaptive mesh. First, we need to design how to generate and index grid points for multiple mesh levels, and how to store and access these grid points by efficient data structure. Second, we need to figure out what FD schemes to use for mesh with multiple resolutions, especially on the border of two mesh resolutions. Third, we need to find a proper solver for the linear systems derived from FD schemes on the adaptive mesh. Fortunately, there are some fast solvers flexible for our need, such as the algebraic multigrid method [59], ITPACK for large sparse linear system [1], and multilevel-multigrid method [51]. In this dissertation, we use the algebraic multigrid (AMG) solver for our adaptive mesh refinement.

1.3.2 Review on numerical algorithms for interface problems

It is usually challenging to solve the PDEs for interface problems, because there are many irregularities associated with the equations. The parameters in the governing equations
are typically discontinuous across the interface separating two materials or two states, and
the source terms are often singular to reflect source/sink distributions along interface.
Because of these irregularities, the solutions to the differential equations are typically
non-smooth, or even discontinuous. As a result, many standard numerical methods based
on the assumption of smoothness of solutions do not work for interface problems. Over
years, interface problems have attracted a lot of attention to solve them accurately and
efficiently. Below we review some of the numerical methods related to our study.

**The immersed boundary method**

The immersed boundary method (IB) was developed by Peskin [55] to model blood
flow in the heart, and since then it has been used for many other problems, such as
mathematical biology and computational fluid dynamics [19, 28, 56, 71]. The IB method
can be applied to the elliptic equations, Stokes equations and Navier-Stokes equations
listed in Section 1.1.2 as well. The main idea is to discretize the interface by a set of
control points \( \mathbf{X}_n = (X_n, Y_n), \ n = 1, 2, ..., N_b \), and replace the singular term in (1.3) by a
discrete sum, and also replace the \( \delta \)-function by some discrete approximation \( \delta_h(x) \) with
support related to the mesh width \( h \). For example,

\[
\int_{\Gamma} v(\mathbf{X}(s)) \delta(\mathbf{x} - \mathbf{X}(s)) \, ds \approx \sum_{k=1}^{N_b} v(\mathbf{X}_k) \delta_h(x_i - X_k) \delta_h(y_j - Y_k) \Delta S_k
\]

(1.8)

where \( \delta_\epsilon(x) \) is the discrete delta function (Figure 1.6) defined by

\[
\delta_\epsilon(x) = \begin{cases} 
\frac{\epsilon - |x|}{\epsilon^2} & \text{if } |x| < \epsilon \\
0 & \text{if } |x| \leq \epsilon
\end{cases}
\]

(1.9)
or

\[ \delta_\epsilon(x) = \begin{cases} 
\frac{1}{4\epsilon} (1 + \cos \frac{\pi x}{2\epsilon}) & \text{if } |x| < 2\epsilon \\
0 & \text{if } |x| \leq 2\epsilon 
\end{cases} \] (1.10)

Figure 1.6: Two typical discrete delta functions (a) Discrete hat delta function (b) Discrete cosine delta function

We can see from above that the IB method is basically a smoothing method with a transition region that smears discontinuities. It seems unlikely that any discrete delta function can yield a second-order accurate solution in a point-wise \( L^\infty \) norm for PDEs. An intuitive explanation is that the delta function is independent of the jump condition \( v(s) \) in (2.2) and the curvature of the interface, so that it could not catch the jump across the interface very accurately.

**The immersed interface method**

The immersed interface method (IIM) was first developed for elliptic equations with discontinuous coefficients and singular source terms [32, 37]. In the IIM, the standard
FD scheme is modified only if its FD stencil is cut by the interface, so that the local truncation error is controlled to be $O(h)$ and the global solution is improved to be $O(h^2)$. To be specific, we first derive the interface jump conditions either from the physics of the problem or from the governing differential equations. And then, we add correction terms involving jump information on right hand side of FD schemes around interface to balance the discontinuities or singularities, and also modify coefficients of FD schemes if the equation has a discontinuous coefficient ($\beta$ in elliptic equation and $\mu$ in Stokes and Navier-Stokes equations). In this way, the IIM can sharply capture the jump across the interface, and provide second order accuracy in a point-wise $L^\infty$ norm. The second order accuracy of the IIM has been shown through numerical examples and confirmed by theoretical analysis such as [5, 21, 38].

Since [32], the IIM has been used for parabolic equation [35], hyperbolic equation [34], Stokes equation [33, 40, 42], Navier-Stokes equation [23, 41, 68], three dimensional case [14, 70], and moving interface problems [36]. Meanwhile, the IIM has been used for many application problems involving interface, such as Hele-Shaw problem [20], electro-migration of voids [46], crystal growth problem [45], interfacial flows with surfactant [72], inverse interface problem of shape identification [22], etc. See more details in the monograph [39] about the IIM.

**Others interface numerical approaches**

There are also some other interface numerical approaches besides the IB and the IIM. For example, the Heaviside function (See (3.21)) is used to smooth the discontinuous coefficient $\beta$. That method is easy to implement but may not be accurate. The harmonic averaging is also used for discontinuous coefficient, but we need to calculate the integral accurately, and that approach is not second-order accurate in 2D.
There are some numerical methods based on integral equations. Mayo [50] derived an integral equation for elliptic interface problems with piecewise coefficients and obtained second order accuracy. The addition cost is to solve an integral equation. This method is most effective for homogeneous source terms and certain boundary conditions, and some extra effort is needed for nonhomogeneous source terms.

The ghost fluid method (GFM) is also used for elliptic interface problems [47], and applied to multiphase incompressible flows. The GFM is a sharp interface method because it is built on the jump conditions as the IIM. But the GFM is generally first order accurate in infinity norm. The GFM is easier to implement because it decomposes the flux jump in each axis direction and then the problem can be treated dimension by dimension.

1.3.3 Review on adaptive mesh refinement techniques

The local adaptive mesh refinement technique (AMR) was first developed by Berger [7–9] for the numerical solution of hyperbolic partial differential equations, and since then the AMR has been extended to many other applications. Berger’s AMR algorithm approximates the numerical solutions of PDEs on a hierarchical sequence of nested, logically rectangular grids. The algorithm starts by the estimation of the solution error at every grid point; afterwards, grid points whose error estimates exceed a certain critical value are flagged and clustered together, along with some other non-flagged points. The result is a refined region whose shape is given by a union of rectangular grid patches.

Roma [57,58] combined Peskin’s Immersed Boundary (IB) method and Berger’s AMR to develop an adaptive version of the IB. The accuracy of the IB was enhanced locally by covering the immersed boundary vicinity with a sequence of nested, progressively finer rectangular grid patches. The AMR-IB was implemented in a 2D benchmark elastic band
problem [69], in which there was no significant difference between the solutions obtained on a mesh refined locally around immersed boundary, and the mesh refined uniformly in entire domain with equal finest resolution, see Figure 1.7 for the AMR which Roma used. After that, Griffith [17] extended the adaptive version of the IB based on a formally second order accurate version of the IB [18] (i.e., second order accurate for problems with sufficiently smooth solutions). He presented the application of this method to a 3D simulation of blood flow in the heart and great vessels. Besides, Berger’s AMR can also be applied for two-phase incompressible flow problems, such as [63].

Figure 1.7: Roma’s AMR(left) and grid patches(right)

Roma’s work is the initial motivation of our study. Since both the IB and the IIM are numerical techniques for interface problems, we expect to develop an adaptive version of the IIM, similar to what Roma did for the IB. However, the mesh generation and implementation we designed are quite different from Roma’s. We use the level set method to represent interface whereas he used Lagrangian formulation to represent interface by control points.
Although Berger’s AMR is the preferred technique for flows with shock waves, that is wasteful in the sense that many fine grid cells are introduced in each patch even when only a few may be required. A more efficient method is the quadtree based method in 2D [62] and the octree based method in 3D [49]. The price is a more complicated data structure in algorithm implementation. There are some other adaptive mesh refinement techniques developed for level set method described below. See the review [48].

1.3.4 Review on the level set method

There are several approaches to represent an interface, such as analytic expressions, Lagrangian formulation, and the level set method. The analytic expressions are used only in testing cases where the interface is in simple shape, such as circle, ellipse and parabola. Otherwise the analytic expressions are rarely available, or the interface geometries are too complicated to evaluate.

The Lagrangian formulation represents interface by a set of ordered particles $X_k(t)$, $k = 1, 2, ..., N_b$, or so-called control points. By this approach, there are two different coordinate systems: the uniform Cartesian grid in Eulerian coordinates for finite difference method, and the Lagrangian coordinates for control points defined only on interface. For moving interface problems, the key is to exchange information between two grids accurately, such as Peskin’s interpolation [54] for the IB and its revised version for the IIM [37]. Moreover, a cubic spline interpolation was developed in [35] to use piecewise polynomials to represent interface and compute interface geometries. That approach was shown to be efficient and reliable for some 2D moving interface problems [33]. However, the Lagrangian formulation fails when an interface develops topological changes such as breaking and merging, or in three-dimensional problems. This is the reason why we prefer
the level set method below for our study.

The level set method [53] can implicitly capture the interface on the same Eulerian grid by a zero level set of Lipschitz continuous function $\varphi(x, y, t)$

$$\Gamma = \{(x, y), \varphi(x, y, t) = 0 \}.$$  

by which the interface evolution can be solved by a Hamilton-Jacobi type equation. See details in Section 3.2 or the book [60]. The interface geometries are easy to compute by level set function $\varphi(x, y, t)$ around interface. And the level set can naturally tackle the cases of topological changes or three dimensions.

Since the work of Sussman [66], the level set method has been applied for two-phase incompressible flow problems, in which the topological changes of interface are handled without additional efforts. The weakness of level set, the loss of mass, was also studied intensively such as [64]. There are some recent work for accurate level set method for incompressible flows, such as Hamilton-Jacobi WENO [26], particle level set [15], coupled level set volume of fluid method [65], etc.

### 1.4 Contributions and an outline of the dissertation

This study is the first time that the adaptive mesh refinement (AMR) techniques are applied to immersed interface method (IIM) using finite difference discretizations. The adaptive version of the IIM (AMR-IIM) has been successfully used in 2D elliptic equations, Stokes equations and Navier-Stokes equations involving interface. The numerical results show that the AMR-IIM preserves the second order accuracy of the IIM, and utilizes the computational resources more efficiently by reducing numbers of grid points.
The adaptive mesh is generated based on the level set function, which is different from traditional grid generation techniques using rectangular patches. A brand new mesh generation method, coupled with original bookkeeping procedure and data structure, are developed. They are shown to be accurate and efficient through many benchmark interface problems. Because the AMR-IIM is designed based on the level set method, it possesses the advantages of level set method and it can be conveniently used for moving interface problems.

Chapter 2 introduces the AMR-IIM for elliptic interface problems. This chapter is the foundation of the dissertation. It first describes how to generate the adaptive mesh, how to flag and index grid points, and how to store grid points by efficient data structure. Chapter 2 also describes what finite different schemes to use for various cases among different mesh levels. Moreover, this chapter introduces the usage of an algebraic multigrid solver, which is the main tool we use for the linear systems derived from FD equations. Plenty of examples about elliptic interface problems are tested and the solutions are compared against those by the well-developed IIM on uniform mesh.

Chapter 3 describes the AMR-IIM for Stokes equations with interface, especially in moving interface problems. The Stokes equations is solved by calling the AMR-IIM elliptic solver three times. The level set method and the interface evolution techniques are briefly reviewed, and then an algorithm is designed to couple the AMR-IIM with level set method for moving interface problems. We carefully determine the widths of three tubes for the IIM, the AMR and the level set, so that the computations in the three tubes are compatible with each other. The surface tension problem and the bubble creeping problem are used to show accuracy and efficiency of AMR-IIM.

Chapter 4 is about the AMR-IIM for Navier-Stokes equations with interface. We first review the projection-IIM method for an incompressible flow with interface, and then
develop an algorithm to couple our AMR with the projection-IIM for moving interface problems. A circular flow problem is used to test the accuracy of AMR-IIM, and then the surface tension problem is studied again in the case of large Reynold number so that some different behaviors of flow motion can be observed.

Chapter 5 concludes the dissertation by reviewing some important numerical aspects through the method development and providing some potential extensions.
Chapter 2

The AMR-IIM for Elliptic PDEs with interfaces

In this chapter, we study the adaptive mesh refinement version of immersed interface method (AMR-IIM) used for elliptic interface problems in the form of

\[ \nabla \cdot (\beta \nabla u) = f(x), \quad x \in R, \]  

(2.1)

together with jump conditions across interface $\Gamma$

\[ [u]_{\Gamma} = w(s), \quad \left[ \frac{\partial u}{\partial n} \right]_{\Gamma} = v(s), \]  

(2.2)

and boundary conditions $u(x)$ along $\partial R$, where $R$ is a rectangular domain, $\Gamma \in C^2$ is a smooth interface that can be parameterized by one dimensional variable $s$, say the arc-length, within the domain $R$; $w(s) \in C^2$ and $v(s) \in C^1$ are two functions defined
along the interface $\Gamma$. Note that, when $w(s) = 0$, the problem can be written as

$$\nabla \cdot (\beta \nabla u) = f(x) + \int_{\Gamma} v(X(s)) \delta(x - X(s)) \, ds, \quad x \in \mathbb{R},$$

(2.3)

where $X(s)$ is a point on the interface $\Gamma$.

The AMR-IIM for elliptic interface problems was originally published in [44], in which we described the mesh generation, data structure and results on a preliminary benchmark example. After that, we extended that work to more general elliptic interface problems, such as discontinuous coefficient and complicated interface. We also did more numerical experiments to study the property of AMR-IIM in depth.

### 2.1 The adaptive mesh generation procedure

We assume that the interface problem is defined in a rectangular domain $\Omega = [a, b] \times [c, d]$. We start with a coarse Cartesian grid, $x_i = a + ih$, $y_j = c + jh$, $i = 0, 1, \cdots, m$, $j = 0, 1, \cdots, n$. The interface $\Gamma$ is implicitly represented by the zero level set of a Lipschitz continuous function $\varphi(x,y)$:

$$\Gamma = \left\{ (x, y), \varphi(x, y) = 0 \right\}.$$  

(2.4)

Note that we do not consider time in this chapter so that $\varphi$ is a function of only $x$ and $y$. In the discrete case, $\varphi(x,y)$ is defined at grid points as $\varphi(x_i, y_j)$. Often $\varphi(x, y)$ is the signed distance function from $\Gamma$.

To generate a finer mesh around interface $\Gamma$, we first select parent grid points within
a tube of the interface according to

\[ |\varphi(x, y)| \leq \lambda h, \]  \hspace{1cm} (2.5)

where \( \lambda \) is a control coefficient to adjust the width of the refinement tube. When a grid point \( x_{ij} = (x_i, y_j) \) is selected, we build a refined mesh with higher mesh resolution \( h/r \) (\( r \) is refinement ratio, \( r = 2 \) or 4, for example) within the square: \[ |x - x_i| \leq h \] and \[ |y - y_j| \leq h. \] Generating refined square for every parent grid points yields a refined region around interface. Its width is flexibly controlled by \( \lambda \). Figure 2.1 shows an example of the refinement mesh around a circular interface.

![Figure 2.1: Adaptive mesh around a circular interface (red solid). Parent grid points (starred) are selected within the tube \( |\varphi_{ij}| = |\varphi(x_i, y_j)| \leq h \) (red dashed). Then, square meshes in finer resolution are generated from parent grid points. The refinement ratio \( r = 2 \), and the refinement tube width \( \lambda = 1 \).](image)
If a finer mesh is needed, we can further select second level parents from the newly generated second level grid points by:

\[ |\varphi(x, y)| \leq \lambda h/r, \quad (2.6) \]

and then refine with higher resolution \( h/r^2 \) within \( h/r \times h/r \) squares centered at every selected second level parent. We can repeat this process to get finer and finer levels of adaptive mesh.

Throughout this dissertation, we use the same notations as those used in [57, 58]. AMR \( m(+i)[r] \) means adding \( i \) levels of adaptive meshes based on a uniform \( m \times m \) uniform mesh with the refinement ratio \( r \).

Next we describe how to index grid points from multiple levels and store their information by efficient data structure. Note that the above generation procedure causes large amount of repetitions among different levels of the mesh. For example, some grid points can be considered to be both parents and children multiple times in various levels. Some child grid points can be generated again and again by different parents around them. Although these repetitions do not affect the mesh pattern, they do cause book-keeping issues and waste a lot storage space. To overcome this problem, we have developed a procedure to flag, index, and store multiple levels of grid points without any repetition. Take the case of adding two levels refinement (AMR \( m(+2)[r] \)) as an example. Also see Figure 2.2 for a flow chart of the procedure, and Figure 2.3 for the generated mesh. Following the generation steps described above, we start from the first level uniform mesh, obtain second and then third level of adaptive meshes. The indexing steps are in a reversed order. The third level grid points are indexed first and flagged as ‘3’. When we index grid points within the second level mesh, we can disregard those already flagged
as ‘3’ since the third level mesh is completely within the second level. We just need to index the grid points that are not yet flagged in the second level mesh and flag them as ‘2’. Last step is for the first level. We disregard the grid points in both the second and the third levels and just index what are left and flag them as ‘1’. In each mesh level, we index grid points and record their positions in the domain. Once all grid points in one level are indexed, the required solution space is then allocated for this level according to amount of indexed grid points.

![Flow chart of a three-level mesh generation and indexing procedure.](image)

Figure 2.2: Flow chart of a three-level mesh generation and indexing procedure.

Figure 2.3 shows a sample of the three-level mesh around a circular interface. We summarize some properties of the adaptive mesh generated by above procedure.

- Every generated grid point is flagged as in a unique level. If a point in the domain is never flagged, that means it is not a grid point in any level and it should interpolated by its neighbor grid points for use. With the flagging information, we can determine what finite difference scheme to use for each grid point, and from which level to
Figure 2.3: Adding two levels of AMR around circle interface: AMR(+2)[2], with refinement tube width $\lambda = 2$.

call its neighbors required in that scheme.

- The $(k + 1)$-th level mesh is entirely embedded in the $k$-th level mesh, so that the $(k + 1)$-th level mesh should never directly touch the $(k - 1)$-th level mesh. That is because we would use interpolation FD scheme on border of two consecutive resolutions, rather than mixing three mesh resolutions together in the interpolation. Moreover, grid points on the border of two levels are always flagged as the finer level.

- The interface and all irregular grid points defined by the IIM should be entirely included in the finest level. The FD schemes of irregular grid points (correction terms and perhaps adjusted coefficients) should be exactly the same as what we use for the IIM on uniform mesh, so that our AMR-IIM is able to inherit the second order accuracy of the IIM.
• The width of each AMR tube can be adjusted conveniently by \( \lambda \) according to accuracy requirement. The more the finer level covers, the more accurate the global solution is. When we deal with moving interface problems in Chapter 3, the width of AMR tube can be adjusted easily to be compatible with both the level set updating tube and the level set re-initialization tube.

• In the early stage of our work, such as [44], we restricted that each FD scheme should include grid points from no more than two mesh levels. After that, however, we no longer enforce that requirement because we added a subroutine which can obtain any desired grid point from the correct level it belongs to. Now we are comfortable to use FD schemes involving three mesh levels together.

2.2 The data structure of the AMR

We illustrate the data structure of our AMR in Figure 2.4 to store and get access to the information of all grid points. For simplicity, we assume that the domain is a square. We start with the first level uniform Cartesian mesh of \( m \times m \) mesh, based on which a second level adaptive mesh is generated with refinement ratio \( r \). There are at most \((mr + 1) \times (mr + 1)\) possible grid points in the refined domain (the second level) and the unrefined domain. We use the integer coordinate \((i, j)\), \(0 \leq i, j \leq mr\) to record grid points rather than their actual location \((x_i, y_j)\)'s. Suppose we count \( K \) grid points in the first level and \( L \) grid points in the second level. Two double precision vectors \( U1_{K \times 1} \) and \( U2_{L \times 1} \) are allocated to store the solutions of the two levels. We use an array \( Locate1_{K \times 2} \) to link the \( k \)-th \((1 \leq k \leq K)\) grid point in the first level mesh to its integer coordinate \((i, j)\) in the domain. Similarly we use an array \( Locate2_{L \times 2} \) to link the \( l \)-th \((1 \leq l \leq L)\) grid point in the second level mesh to its integer coordinate. Inversely, integer matrix
\(\text{Pointer}_{(mr+1)\times(mr+1)}\) maps the positions index \((i,j)\) to its corresponding index in the level 1 or 2 meshes. Because some positions index have never been flagged, the matrix \(\text{Pointer}\) can have many undefined entries. Moreover, integer matrix \(\text{Flag}_{(mr+1)\times(mr+1)}\) maps each position index to its flag information: 0 (not a grid point in any level of the mesh), in the level 1 or level 2 mesh. This data structure is also convenient to be extended to more levels of AMR. For example, to add a finer third level, we can add array \(U3\) for the third finest level and array \(\text{Locate}3\) to link solutions and grid point positions. The matrices \(\text{Pointer}\) and \(\text{Flag}\) can include the third level information through the mesh generation procedure.

Notice that except for the solution vectors \(U1\) and \(U2\), all other data structures are integer pointer arrays or matrices to store grids information. This data structure meets our original goal of AMR to reduce the solution space, or, the amount of unknowns to solve in linear system derived from FD equations. The design of double directed pointer by \(\text{Locate}1\), \(\text{Locate}2\) and \(\text{Pointer}\) enables us to find desired grid points or their positions efficiently without additional searching time.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2_4.png}
\caption{The data structure to book-keep the information of two levels grid points.}
\end{figure}
2.3 Finite difference schemes on the adaptive meshes

We describe the finite difference (FD) schemes for solving a Poisson equation with an interface on the generated adaptive mesh

\[ \beta(u_{xx} + u_{yy}) = f \]  \hfill (2.7)

where the right hand side \( f \) could include an singular source and a jump. The coefficient \( \beta \) could also include a jump. The more general Helmholtz equation \( \beta(u_{xx} + u_{yy}) + \lambda u = f \) follows exactly the same cases described below. The FD schemes for these elliptic equations are also used in Chapter 3 and 4 because both Stokes and Navier-Stokes equations can be reduced to several Poisson or Helmholtz equations.

Below we discuss the FD schemes for different types of grid points on AMR. Take Figure 2.5 as an example. It can be considered as a locally zoom-in of the AMR(+1)[2] in Figure 2.1. Suppose the finest resolution is \( h \) so that the coarse resolution is 2\( h \). Starting with the grid points closest to interface, we have four types of FD schemes from the fine mesh to the coarse mesh.

- Irregular grid points such as 10, 16. A grid point \((x_i, y_j)\) is called irregular grid point if the interface cuts through the central 5-point stencil centered at \((x_i, y_j)\).

Using the level set function, a grid point \((x_i, y_j)\) is irregular if \( \varphi_{ij}^{\text{max}} \varphi_{ij}^{\text{min}} \leq 0 \), where

\[
\varphi_{ij}^{\text{max}} = \max \left\{ \varphi_{i-1,j}, \varphi_{i+1,j}, \varphi_{ij}, \varphi_{i,j-1}, \varphi_{i,j+1} \right\}, \quad (2.8)
\]

\[
\varphi_{ij}^{\text{min}} = \min \left\{ \varphi_{i-1,j}, \varphi_{i+1,j}, \varphi_{ij}, \varphi_{i,j-1}, \varphi_{i,j+1} \right\}. \quad (2.9)
\]

An irregular grid point must be in the finest mesh. For irregular grid points, we use the immersed interface method to get the finite difference equation, which is
Figure 2.5: A portion of an adaptive mesh that includes the fine level mesh around an interface (red curve) and the coarse mesh. Hanging nodes 8, 12, 19 lie on border of two mesh levels. Ghost points $g1$ and $g2$ are used to derive the finite difference equation for node 12.

A modification of standard 5-point FD scheme in finest resolution. See [32, 38, 39] for details. For the case where the coefficient $\beta$ is continuous, we use the standard central finite difference discretization but add a correction term $C_{ij}$ on the right hand side to enforce truncation error to be $O(h)$,

$$\beta \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}}{h^2} = f_{ij} + C_{ij} + O(h). \quad (2.10)$$

The correction term $C_{ij}$ is determined from the given jump conditions on the interface $[u]$, $[\beta u_n]$, $[f]$, and the interface curvature $\kappa$. When the coefficient $\beta$ is discontinuous, we used a 9-point maximum principle preserving FD scheme

$$\sum_k \gamma_k u_{i+k,j+k} = f_{ij} + C_{ij} + O(h) \quad (2.11)$$

in which $i_k, j_k = 0, \pm 1$, and coefficients $\gamma_k$ are solved from a quadratic constrained optimization problem to enforce coefficient matrix to be diagonal dominant, and
the correction term $C_{ij}$ is added to guarantee an $O(h)$ truncation error.

- Regular grid points in the fine mesh such as 7, 11, 18, of which the central 5-point stencils are all from one side of the interface. We use the standard central finite difference scheme for these regular grid points.

$$
\beta \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}}{h^2} = f_{ij} + O(h^2).
$$

- Hanging nodes such as 8,12, which lie on the border of two different meshes. We can not use the standard central finite difference scheme since the finite difference stencil lacks some neighbor grid points in certain directions due to different mesh resolutions. As a common practice, we interpolate the so called ghost points such as $g_1$ and $g_2$ to obtain the FD schemes on hanging nodes. Take the hanging node 12 as an example, we use a quadratic interpolation with third order accuracy to approximate $u_{g1}$ and $u_{g2},$

$$
u_{g1} = \frac{3}{8}u_3 + \frac{3}{4}u_4 - \frac{1}{8}u_5 + O(h^3), \quad u_{g2} = -\frac{1}{3}u_{18} + u_{12} + \frac{1}{3}u_{g1} + O(h^3),
$$

and then plug approximated $u_{g2}$ into the discrete 5-point discrete Laplacian on finest resolution $h$ at hanging node 12 to get the finite difference equation for it,

$$
\beta \frac{\frac{2}{3}u_{18} + u_{11} + u_{13} + \frac{1}{8}u_3 + \frac{1}{4}u_4 - \frac{1}{24}u_5 - 3u_{12}}{h^2} = f_{12} + O(h).
$$

Since all hanging nodes take one dimension lower than the whole domain, the global accuracy should not be affected. If a higher accuracy is need, we can use a fourth order interpolation for the ghost points. In the same example about hanging node
12, the ghost point $g2$ can be interpolated by

$$u_{g2} = \frac{1}{4} u_{23} - u_{18} + \frac{3}{2} u_{12} + \frac{1}{4} u_{g1} + O(h^4) \quad (2.15)$$

and $g1$ can be interpolated by:

$$u_{g1} = -\frac{1}{16} u_1 + \frac{9}{16} u_3 + \frac{9}{16} u_4 - \frac{1}{16} u_5 + O(h^4) \quad (2.16)$$

so that the ghost point $g2$ can be approximated with $O(h^4)$ accuracy:

$$u_{g2} = \frac{1}{4} u_{23} - u_{18} + \frac{3}{2} u_{12} - \frac{1}{64} u_1 + \frac{9}{64} u_3 + \frac{9}{64} u_4 - \frac{1}{64} u_5 + O(h^4) \quad (2.17)$$

and the derived FD scheme at 12 has $O(h^2)$ accuracy.

- Regular grid points in coarse resolutions such as 4, 13, 24. We can use the standard 5-point finite difference discretization on course resolution $2h$,

$$\beta \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}}{(2h)^2} = f_{ij} + O(h^2). \quad (2.18)$$

We can also use 9-point compact scheme on the coarse level in resolution $2h$ to reduce its effect on global solution error. We know that the 9-point Laplacian:

$$\nabla_9^2 u_{ij} = \frac{1}{6h^2} [4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} + u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{ij}] \quad (2.19)$$
can be used to approximate $\nabla^2 u$ by:

$$\nabla_{9}^2 u_{ij} = \nabla^2 u_{ij} + \frac{(2h)^2}{12} \nabla^2(\nabla^2 u_{ij}) + O(h^4)$$ (2.20)

Hence we have the 9-point scheme with higher accuracy

$$\nabla_{9}^2 u_{ij} = f_{ij} + \frac{(2h)^2}{12} \nabla^2 f_{ij} + O(h^4)$$ (2.21)

To illustrate above FD schemes, we show the pseudo-code in Algorithm 1 about how to determine proper FD schemes for grid points in AMR $m(+2)[2]$ mesh such as Figure 2.3. In this case, we have three mesh levels: the base uniform $m \times m$ mesh, the first level AMR and second level AMR. Since the refinement ratio is 2, the resolutions of three levels are $4h$, $2h$ and $h$, where $h$ is finest resolution. After the mesh generation steps 2.2, we have counted that there are $L_1$, $L_2$, $L_3$ grid points in three levels. We go through all grid points in each level, obtain their positions by arrays $\text{Locate}_1$, $\text{Locate}_2$, $\text{Locate}_3$, and then check by matrix $\text{Flag}$ to see if the grid points have complete neighbors required by FD schemes. If not, we use interpolation for ghost points. If several FD schemes are all qualified for one grid point, we prefer the scheme with higher accuracy and smaller resolution.

2.4 The Algebraic Multigrid solver

Based on above Section 2.3, we can have a finite difference equation for every grid point in the domain. The next step is to solve the linear system with all grid points as unknowns. The Gauss-Seidel and SOR method are able to solve this case but they converge slowly.
Algorithm 1 Determine FD schemes for three levels of grid points in AMR(+2)[2]

1: for $k = 1$ to $L_1$ (for every grid point on coarsest uniform level) do
2: 
3: Obtain position $(i, j)$ from $i = \text{Locate}1(k, 1), j = \text{Locate}1(k, 2)$
4: 
5: if On boundary then
6: 
7: Call Dirichlet boundary condition
8: 
9: else
10: Use 9-point scheme with resolution $4h$
11: end if
12: end for

13: for $k = 1$ to $L_2$ (for every grid point on first level AMR) do
14: 
15: Obtain position $(i, j)$ from $i = \text{Locate}2(k, 1), j = \text{Locate}2(k, 2)$
16: 
17: if $\text{Flag}(i, j \pm 2), \text{Flag}(i \pm 2, j), \text{Flag}(i \pm 2, j \pm 2)$ are non-zero then
18: 
19: Use 9-point scheme with resolution $2h$
20: 
21: else if $\text{Flag}(i, j \pm 4), \text{Flag}(i \pm 4, j), \text{Flag}(i \pm 4, j \pm 4)$ are non-zero then
22: 
23: Use 9-point scheme with resolution $4h$
24: 
25: else
26: Interpolate the ghost point and then use 5-point scheme on resolution $2h$
27: end if
28: end for

29: for $k = 1$ to $L_3$ (for every grid point on 2nd level AMR, or the finest level covering interface) do
30: 
31: Obtain position $(i, j)$ from $i = \text{Locate}3(k, 1), j = \text{Locate}3(k, 2)$
32: 
33: if Irregular grid point close to interface then
34: 
35: Call correction terms $C_{ij}$ and perhaps modified coefficients computed by the IIM
36: 
37: Use 5-point scheme on resolution $h$ with above corrections
38: 
39: else if $\text{Flag}(i, j \pm 1), \text{Flag}(i \pm 1, j)$ are non-zero then
40: 
41: Use 5-point scheme with resolution $h$
42: 
43: else if $\text{Flag}(i, j \pm 2), \text{Flag}(i \pm 2, j), \text{Flag}(i \pm 2, j \pm 2)$ are non-zero then
44: 
45: Use 9-point scheme with resolution $2h$
46: 
47: else
48: Interpolate the ghost point and then use 5-point scheme on resolution $h$
49: end if
50: end for
The well known Fishpack [3,67] and multigrid dmgd9v [73] solvers are fast, but they are usually designed for 5-point or 9-point FD schemes on uniform mesh. Here we need a fast solver which is flexible enough to deal with standard 5-point or 9-point FD schemes in different resolutions, as well as the interpolation schemes such as (2.14) and (2.17).

The Algebraic Multigrid method (AMG) developed by German National Research Center for Information Technology is suitable for our case and it is available online for research purpose. [59]. The AMG can solve the linear system $L \ast u = f$ where the coefficient matrix belongs "essentially" positive type, i.e.,

- Diagonal entries are all positive
- Most of the off-diagonal entries are less than or equal to zero
- Each row has row sum larger than or equal to zero

All the FD schemes we use in Section 2.3 satisfy above requirements. And the AMG solver runs very fast. For example, it takes no more than 5 seconds on a regular laptop for the solution on a $640 \times 640$ uniform mesh.

There are many parameters needed to set up before calling this solver, and the authors suggested some standard choices of parameters in the manual. To use this solver for our AMR study, one need to pay attention to following cases and make modification as necessary.

- The convergence criteria (parameters "NCYC" and "EPS"). The user could set up the maximum number of iterations or the tolerance for convergence. It is recommended to set up the convergence criteria as strict as possible to obtain an accurate solution and avoid potential program blow up.
• When there is a good first guess of the solution, such as in the moving interface problems in Chapter 3, we can change the parameter ”IFIRST” to save at least half solving time.

• When the Neumann or periodic boundary conditions are used on all four boundaries, such as in the projection method in Chapter 4, it is required to change the parameter ”MATRIX” to tell the solver that the coefficient matrix is singular.

2.5 Numerical experiments and analysis

We did numerical experiments on plenty of elliptic interface problems to test the new AMR-IIM method. In Section 2.5.1, we show a benchmark problem with simple geometry and compare the results by AMR-IB and AMR-IIM. In Sections 2.5.2-2.5.6, we show more general cases, including variable coefficient, non-constant jump, ellipse interface, piecewise constant coefficient and complicated interface.

We use the same notations as those used in [57,58]. AMR $m(+i)[r]$ denotes adding $i$ levels of adaptive meshes with a refinement ratio $r$ based on a uniform $m \times m$ mesh. In all below tables, the column “$\|E_u\|_\infty$” shows the solution error in $L^\infty$ norm. The column “Size” shows the number of grid points, or equivalently the number of unknowns to be solved in the derived linear system.

The AMR-IIM is coded in Fortran 90, except some included classic solvers are in Fortran 77. All results in this study are solved by the Algebraic Multigrid solver described in Section 2.4. We use a regular laptop with Intel Core 2 Duo P9400 2.4Hz processor and 3GB RAM. We use the Compaq Visual Fortran 6.6.0 compiler on a Windows 7 32-bit operating system. Every computation in this section was completed within one or a few seconds. Since the Compaq Visual Fortran is no longer supported by the latest Windows
systems on 64-bit machines, we also compiled all programs by gfortran and ran on a Linux workstation at Department of Mathematics, North Carolina State University. The numerical results by the two compilers agree to at least 6 digits.

2.5.1 A benchmark elliptic interface problem: AMR-IB vs AMR-IIM

This is the first example used in [32] to show the second order accuracy of the IIM. And then, this example was used in [58] to show the performance of the adaptive version of the IB. Here we use this example to show our AMR method works well for both the IB and the IIM. We present the results of one level, two levels and four levels of AMR and show their advantages over uniform mesh. Note that our AMR is generated quite differently from Roma’s AMR in [58]. Roma’s method is based on Berger’s AMR techniques, but ours is based on level set method.

Consider the following Poisson equation in the domain $\Omega : -1 \leq x, y \leq 1$ with a singular source from a circular interface $\Gamma$: $x^2 + y^2 = 1/4$:

$$u_{xx} + u_{yy} = \int_{\Gamma} 2 \delta(x - X(s))\delta(y - Y(s)) \, ds,$$

(2.22)

It can be equivalently rewritten as a Poisson equation with jump conditions

$$u_{xx} + u_{yy} = 0, \quad \text{on } \Omega \backslash \Gamma, \quad [u]_\Gamma = 0, \quad \left[ \frac{\partial u}{\partial n} \right]_\Gamma = 2,$$

(2.23)

The Dirichlet boundary conditions are given from below exact solution. See also Figure 35
2.6.

\[ u(x, y) = \begin{cases} 
1 & \text{if } r \leq 1/2 \\
1 + \log(2r) & \text{if } r > 1/2
\end{cases} \]  

(2.24)

where \( r = \sqrt{x^2 + y^2} \).

Figure 2.6: The solution plot of Example 1

First, we solve the problem by the AMR-IB and show the results in Table 2.1 and 2.2. In Table 2.1, we show the solutions by uniform mesh on left and the solutions by AMR(+1)[4] mesh on right. The results from an equal finest resolution \( h \) are put in the same row. The “ratio” columns show the ratios of two consecutive solution errors in refinement analysis, which should be around 2 for first order method and 4 for second order method. We can observe that the AMR-IB preserves the first order accuracy of the IB method. To obtain a similar accuracy by equal finest resolution, the AMR can save
60% to 90% amount of grid points, that is, the number of unknowns to be solved in the derived linear system.

Table 2.1: A comparison of grid refinement analysis using Uniform-IB and AMR-IB with the same finest resolution $h$.

<table>
<thead>
<tr>
<th>Finest $h$</th>
<th>$m$</th>
<th>Size</th>
<th>$|E_u|_\infty$</th>
<th>Ratio</th>
<th>AMR $m(\pm 1)[4]$, $\lambda = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/80</td>
<td>80</td>
<td>6241</td>
<td>1.34e-2</td>
<td>Ratio</td>
<td>20</td>
</tr>
<tr>
<td>2/160</td>
<td>160</td>
<td>25281</td>
<td>6.74e-3</td>
<td>1.99</td>
<td>40</td>
</tr>
<tr>
<td>2/320</td>
<td>320</td>
<td>101761</td>
<td>3.36e-3</td>
<td>2.01</td>
<td>80</td>
</tr>
<tr>
<td>2/640</td>
<td>640</td>
<td>408321</td>
<td>1.67e-3</td>
<td>2.01</td>
<td>160</td>
</tr>
</tbody>
</table>

We also try multiple levels of AMR to get finer resolution and better results as in [58].

In Table 2.2, up to 4 levels of AMR are generated. The results agree with the argument in [57, 58] about the AMB-IB method that the accuracy attained by refining only the non-smooth region is the same as that if the whole domain is uniformly refined with the same finest resolution in the adaptive grids. This is because for the IB method, the largest error for elliptic interface problems usually occurs in the neighborhood of the interface.

Table 2.2: Two and four levels of AMR for the IB method

<table>
<thead>
<tr>
<th>Finest $h$</th>
<th>Mesh Selection</th>
<th>Size</th>
<th>$|E_u|_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/640</td>
<td>160(+2)[2], $\lambda = 2$</td>
<td>38821</td>
<td>1.46e-3</td>
</tr>
<tr>
<td>2/1280</td>
<td>80(+2)[4], $\lambda = 1$</td>
<td>39981</td>
<td>5.04e-4</td>
</tr>
<tr>
<td>2/1280</td>
<td>80(+4)[2], $\lambda = 2$</td>
<td>38829</td>
<td>4.51e-4</td>
</tr>
</tbody>
</table>

Next, we solve this problem by the AMR-IIM. In general, if the large error occurs near the neighborhood of the interface, we can see the improvement in the accuracy
Table 2.3: A comparison of grid refinement analysis using Uniform-IIM and AMR-IIM with the same finest resolution $h$. The CPU times spent in algebraic multigrid solver are also recorded.

<table>
<thead>
<tr>
<th>Finest h</th>
<th>$m$</th>
<th>Size</th>
<th>$|E_u|_\infty$</th>
<th>Ratio</th>
<th>Time</th>
<th>$m$</th>
<th>Size</th>
<th>$|E_u|_\infty$</th>
<th>Ratio</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/40</td>
<td>40</td>
<td>1521</td>
<td>8.44e-4</td>
<td>0.03</td>
<td>20</td>
<td>805</td>
<td>9.07e-4</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/80</td>
<td>80</td>
<td>6241</td>
<td>2.45e-4</td>
<td>3.4</td>
<td>0.08</td>
<td>40</td>
<td>2745</td>
<td>2.86e-4</td>
<td>3.2</td>
<td>0.06</td>
</tr>
<tr>
<td>2/160</td>
<td>160</td>
<td>25281</td>
<td>6.69e-5</td>
<td>3.7</td>
<td>0.31</td>
<td>80</td>
<td>10201</td>
<td>7.80e-5</td>
<td>3.7</td>
<td>0.19</td>
</tr>
<tr>
<td>2/320</td>
<td>320</td>
<td>101761</td>
<td>1.57e-5</td>
<td>4.3</td>
<td>1.3</td>
<td>160</td>
<td>39225</td>
<td>1.89e-5</td>
<td>4.1</td>
<td>0.67</td>
</tr>
<tr>
<td>2/640</td>
<td>640</td>
<td>408321</td>
<td>3.65e-6</td>
<td>4.3</td>
<td>5.4</td>
<td>320</td>
<td>153805</td>
<td>5.04e-6</td>
<td>3.8</td>
<td>2.5</td>
</tr>
</tbody>
</table>

using the AMR-IIM. If the large error occurs at the region outside of refinement region, then we may not see any improvement in the accuracy using the AMR-IIM. Anyhow, the AMR-IIM will not affect global second order accuracy but reduce the size the linear system of the finite difference equations.

In Table 2.3, we show the results of the AMR-IIM for the same example. Even with a coarse grid $40 \times 40$, the error obtained from the IIM is already smaller than that of the IB with a $640 \times 640$ mesh. We set the width of the refinement tube $\lambda = m/20$ so that the refinement region is thick enough to improve the accuracy. By observing the results horizontally we can see that the AMR-IIM saves 40%-50% of unknowns, with about the same accuracy as the Uniform-IIM on the same row. By observing the results vertically, the ratios of two consecutive errors are around 4, which means the AMR-IIM preserves the second order accuracy of the IIM.

To further show the advantage of our adaptive mesh over a uniform mesh, we designed another series of numerical experiments in Table 2.4. Given a solution error tolerance $\epsilon$, we try to find the minimum number of unknowns needed by various meshes in order to satisfy the accuracy $E_\infty \leq \epsilon$. We first select a proper AMR $m(+i)[r]$ to roughly obtain the required accuracy, and then tune the width $\lambda$ of the refinement tube to slightly control the
error below tolerance. From Table 2.4, we can see that when $\epsilon = 10^{-4}$ the uniform mesh needs about 16900 unknowns, the AMR 80(+1)[2] needs about 7945 unknowns, and the AMR 40(+2)[2] needs about 8353 unknowns. When the error tolerance is lowered to $\epsilon = 2.5 \times 10^{-5}$, the uniform mesh needs at least 65025 unknowns, whereas the AMR 160(+1)[2] needs only 30249 unknowns, and the AMR 80(+2)[2] needs about 28833 unknowns. In general, AMR methods can save about half of the computational cost than that of the uniform mesh for the same accuracy.

### 2.5.2 An elliptic PDE with a variable and discontinuous coefficient

This example from [32, 39] has been used in quite a few paper as a benchmark interface problem. We solve the Poisson equation $(\beta u_x)_x + (\beta u_y)_y = f(x, y)$ on a rectangular domain $[-1, 1] \times [-1, 1]$ with a circular interface $x^2 + y^2 = 1/4$. The coefficient $\beta$ is variable and discontinuous:

$$
\beta(x, y) = \begin{cases} 
x^2 + y^2 + 1 & \text{if } r \leq 1/2, \\
b & \text{if } r > 1/2,
\end{cases}
$$

where $r = \sqrt{x^2 + y^2}$. 

<table>
<thead>
<tr>
<th>Mesh Selection</th>
<th>$|E_u|_{\infty}$</th>
<th>Size</th>
<th>Mesh Selection</th>
<th>$|E_u|_{\infty}$</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform: $m = 131$</td>
<td>1.00e-4</td>
<td>16900</td>
<td>Uniform: $m = 256$</td>
<td>2.59e-5</td>
<td>65025</td>
</tr>
<tr>
<td>AMR: 80(+1)[2], $\lambda = 1$</td>
<td>9.95e-5</td>
<td>7945</td>
<td>AMR: 160(+1)[2], $\lambda = 2$</td>
<td>2.50e-5</td>
<td>30249</td>
</tr>
<tr>
<td>AMR: 40(+2)[2], $\lambda = 5$</td>
<td>8.13e-5</td>
<td>8353</td>
<td>AMR: 80(+2)[2], $\lambda = 9$</td>
<td>2.45e-5</td>
<td>28833</td>
</tr>
</tbody>
</table>
The right hand side \( f \), jump conditions and Dirichlet boundary conditions are all given from the below exact solution. See also Figure 2.7.

\[
u(x, y) = \begin{cases} 
  r^2 & \text{if } r \leq 1/2 \\
  \left(1 - \frac{1}{8b} - \frac{1}{b}\right)/4 + \left(\frac{r^4}{2} + r^2\right)/b + C \log(2r)/b & \text{if } r > 1/2
\end{cases}
\]  

(2.26)

So that the right hand side should be

\[
f(x, y) = 8(x^2 + y^2) + 4
\]

(2.27)

We set up parameters as \( b = 10 \) and \( C = 0.1 \) in this example.

Figure 2.7: The solution plot of Example 2

In Table 2.5, we show a comparison of a grid refinement analysis using the Uniform-IIM and the AMR-IIM. It can be observed that the accuracy is always improved by adding one refinement level AMR \( m(+1)[2] \) based on the uniform mesh \( m \times m \). Moreover, the
AMR retains the second order accuracy when we refine the AMR $m(+1)[2]$ to be AMR $2m(+1)[2]$ below it. With the same finest resolution $h$, the AMR needs a smaller linear system to solve and hence less CPU time.

Table 2.5: A comparison of grid refinement analysis of Example 2 using Uniform-IIM and AMR-IIM with the same finest resolution $h$.

<table>
<thead>
<tr>
<th>Finest $h$</th>
<th>Uniform</th>
<th>AMR $m(+1)[2]$, $\lambda = m/20$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m$</td>
<td>$|E_u|_\infty$</td>
</tr>
<tr>
<td>2/40</td>
<td>40</td>
<td>6.40e-4</td>
</tr>
<tr>
<td>2/80</td>
<td>80</td>
<td>1.40e-4</td>
</tr>
<tr>
<td>2/160</td>
<td>160</td>
<td>2.98e-5</td>
</tr>
<tr>
<td>2/320</td>
<td>320</td>
<td>7.20e-6</td>
</tr>
<tr>
<td>2/640</td>
<td>640</td>
<td>1.75e-6</td>
</tr>
</tbody>
</table>

In this Example 2, we use standard 5-point FD scheme on coarse level, so that there is larger errors compared to those by a uniformly refined mesh. We can further improve the accuracy by using the fourth order 9-point compact FD scheme on the coarse mesh and a higher interpolation scheme at hanging nodes, as in following Examples 3 to 6.

2.5.3 An elliptic PDE with a discontinuous solution

This example is from [32] about the Poisson equation $u_{xx} + u_{yy} = 0$ on a rectangular domain $[-1, 1] \times [-1, 1]$ with a circular interface $x^2 + y^2 = 1/4$. Unlike Example 1 and 2, this one has a discontinuous solution and a non-constant jump across the interface.

The jump conditions and Dirichlet boundary conditions are given from following exact
solution. See also Figure 2.8.

\[ u(x, y) = \begin{cases} 
  e^x \cos y & \text{if } r \leq 1/2 \\
  0 & \text{if } r > 1/2 
\end{cases} \]  

(2.28)

where \( r = \sqrt{x^2 + y^2} \).

\[ \text{Figure 2.8: The solution plot of Example 3} \]

From this example, we also include the results by adding two levels of AMR. Under the same finest resolution \( h \), the AMR \( m(+2)[2] \) can release more grid points on coarse level far away from interface. It is shown in Table 2.6 that the AMR \( m(+2)[2] \) requires almost half numbers of grid points needed by AMR \( m(+1)[2] \), and only 10-20\% of grid points needed by a uniform mesh especially when \( m \) becomes large. But the solution errors are as good as others in the same row. The second order accuracy is also observed for the AMR \( m(+2)[2] \).
Table 2.6: A comparison of grid refinement analysis of Example 3 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

<table>
<thead>
<tr>
<th>Finest $h$</th>
<th>Uniform $m \times m$</th>
<th>AMR $m(+1)[2]$</th>
<th>AMR $m(+2)[2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m$</td>
<td>$|E_u|_\infty$</td>
<td>Size</td>
</tr>
<tr>
<td>2/80</td>
<td>80</td>
<td>2.61e-5</td>
<td>6241</td>
</tr>
<tr>
<td>2/160</td>
<td>160</td>
<td>7.19e-6</td>
<td>2528</td>
</tr>
<tr>
<td>2/320</td>
<td>320</td>
<td>1.63e-6</td>
<td>10176</td>
</tr>
<tr>
<td>2/640</td>
<td>640</td>
<td>4.11e-7</td>
<td>40832</td>
</tr>
</tbody>
</table>

2.5.4 An elliptic PDE with an elliptic interface

This example is from [5] where the second order accuracy of the IIM was analyzed and demonstrated. We solve the Poisson equation $u_{xx} + u_{yy} = f$ in a rectangular domain $[-1,1] \times [-1,1]$ with an ellipse interface:

$$\frac{x^2}{0.75^2} + \frac{y^2}{0.25^2} = 1$$

(2.29)

A re-initialization procedure should be conducted first on the level set function $\varphi(x, y) = \sqrt{\frac{x^2}{0.75^2} + \frac{y^2}{0.25^2}} - 1$ so that the re-scaled $\varphi(x, y)$ satisfies $|\nabla \varphi| \approx 1$ in order to compute interface geometries accurately. See Section for more details about the level set re-initialization.

The right hand side $f$, jump conditions and Dirichlet boundary conditions are given from the following exact solution. See also Figure 2.9.

$$u(x, y) = \begin{cases} 
\sin x \cos y & \text{if } (x, y) \in \Omega^- \\
0 & \text{if } (x, y) \in \Omega^+
\end{cases}$$

(2.30)

Table 2.7 shows the grid refinement analysis by uniform mesh, adding one level AMR
and adding two levels of AMR.

Table 2.7: A comparison of grid refinement analysis of Example 4 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

<table>
<thead>
<tr>
<th>Finest $h$</th>
<th>Uniform $m \times m$</th>
<th>AMR $m(+1)[2]$</th>
<th>AMR $m(+2)[2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m$</td>
<td>$|E_u|_\infty$</td>
<td>Size</td>
</tr>
<tr>
<td>2/80</td>
<td>80</td>
<td>4.32e-5</td>
<td>6241</td>
</tr>
<tr>
<td>2/160</td>
<td>160</td>
<td>8.36e-6</td>
<td>25281</td>
</tr>
<tr>
<td>2/320</td>
<td>320</td>
<td>1.31e-6</td>
<td>101761</td>
</tr>
<tr>
<td>2/640</td>
<td>640</td>
<td>4.02e-7</td>
<td>408321</td>
</tr>
</tbody>
</table>

2.5.5 An elliptic PDE with a piecewise constant coefficient

This example is from [38, 39] where the maximum principle preserving scheme was developed for the case of discontinuous coefficient. We solve the Poisson equation $(\beta u_x)_x +
\((\beta u_y)_y = f(x, y)\) defined in a rectangular domain \([-2, 2] \times [-2, 2]\) including an ellipse interface \(x^2 + 4y^2 = 1\). The coefficient \(\beta\) is piecewise constant: \(\beta^- = 1\) inside the interface and \(\beta^+ = 1000\) outside the interface.

The right hand side \(f\), jump conditions and Dirichlet boundary conditions are given by the below exact solution. See also Figure 2.10.

\[
\begin{align*}
  u(x, y) &= \begin{cases} 
    x^2 - y^2 & \text{if } x^2 + 4y^2 \leq 1 \\
    \sin x \cos y & \text{if } x^2 + 4y^2 > 1 
  \end{cases} 
\end{align*}
\]  

(2.31)
Table 2.8: A comparison of grid refinement analysis of Example 5 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

<table>
<thead>
<tr>
<th></th>
<th>Uniform $m \times m$</th>
<th>AMR $m(+1)[2]$</th>
<th>AMR $m(+2)[2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finest $h$</td>
<td>$m$</td>
<td>$|E_u|_{\infty}$</td>
<td>Size</td>
</tr>
<tr>
<td>4/80</td>
<td>80</td>
<td>2.70e-4</td>
<td>6241</td>
</tr>
<tr>
<td>4/160</td>
<td>160</td>
<td>5.17e-5</td>
<td>25281</td>
</tr>
<tr>
<td>4/320</td>
<td>320</td>
<td>1.09e-5</td>
<td>101761</td>
</tr>
<tr>
<td>4/640</td>
<td>640</td>
<td>2.46e-6</td>
<td>408321</td>
</tr>
</tbody>
</table>

2.5.6 An elliptic PDE with a complicated interface

This example is from [37, 39] where the augmented IIM was developed for the case of piecewise constant coefficient in elliptic interface problems. We consider the Poisson equation $(\beta u_x)_x + (\beta u_y)_y = f(x, y)$ in rectangular domain $[-1, 1] \times [-1, 1]$ including a flower shape interface given by the following equation in polar coordinate

$$r(\theta) = 0.5 + 0.1 \sin(8\theta), \quad 0 \leq \theta < 2\pi.$$ 

The coefficient $\beta$ is a piecewise constant: $\beta^- = 1$ inside the interface and $\beta^+ = 10$ outside the interface.

The right hand side $f$, the Dirichlet boundary conditions, and the jump conditions are all determined from below exact solution. See also Figure 2.11.

$$u(x, y) = \begin{cases} 
\frac{r^2}{\beta^-} & \text{if } (x, y) \in \Omega^- \\
\frac{r^4 - 0.1 \log 2r}{\beta^+} & \text{if } (x, y) \in \Omega^+, 
\end{cases}$$

where $r = \sqrt{x^2 + y^2}$.

This example can be used to set up the surface tension problem in Chapter 3 and 4.
about Stokes equations and Navier-Stokes equations for moving interface problems. The AMR used for this flower-shape interface is shown in Figure 2.12. Table 2.9 shows the grid refinement analysis by uniform mesh, adding one level AMR and adding two levels of AMR.

Table 2.9: A comparison of grid refinement analysis of Example 6 using uniform mesh, AMR(+1)[2] mesh and AMR(+2)[2] mesh with the same finest resolution $h$.

<table>
<thead>
<tr>
<th>Finest $h$</th>
<th>Uniform $m \times m$</th>
<th>AMR $m(+1)[2]$</th>
<th>AMR $m(+2)[2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m$</td>
<td>$|E_u|_{\infty}$</td>
<td>Size</td>
</tr>
<tr>
<td>2/80</td>
<td>80</td>
<td>1.53e-3</td>
<td>6241</td>
</tr>
<tr>
<td>2/160</td>
<td>160</td>
<td>1.16e-4</td>
<td>25281</td>
</tr>
<tr>
<td>2/320</td>
<td>320</td>
<td>2.99e-5</td>
<td>101761</td>
</tr>
<tr>
<td>2/640</td>
<td>640</td>
<td>3.46e-6</td>
<td>408321</td>
</tr>
</tbody>
</table>

In summary, we learn from Example 1 to 6 that AMR-IIM can be successfully applied to all cases of elliptic equations, which are previously solved by the IIM on uniform mesh.
Both AMR(+1)[2] and AMR(+2)[2] inherit the second order accuracy of the IIM method very well through error refinement analysis. That is because our AMR preserves all the IIM treatment on irregular grid points around interface.

We can also see the advantage of AMR-IIM clearly from Examples 1 to 6. When we add one level (AMR m(+1)[2]) or two levels (AMR m(+1)[2]) of refinement based on uniform mesh $m \times m$, the solution error is reduced significantly. If the goal is to obtain a required finest resolution around interface, the AMR can save 50% - 90% amount of grid points, and hence save more than half CPU times in solving linear systems. Figure 2.13 shows the CPU time distribution in solving an elliptic interface problem by the AMR-IIM. The purpose of the AMR is to save the CPU time spent in the multigrid solver (blue), at the cost of a little extra effort in AMR generation (red) and FD schemes setup.
As for accuracy, our numerical results support previous statement about AMR-IB, such as in [57, 58]: the accuracy by refining only the non-smooth region is very close to that by refining the whole domain uniformly with the same finest resolution. Because the IIM controls the truncation error around the interface to be $O(h)$ (It is $O(1)$ in the IB), the FD schemes used in a coarse mesh far away from the interface could also pollute the global accuracy. In the coarse level mesh, we can use the standard 5-point FD scheme for convenience as in Examples 1 and 2, or the 9-point compact FD scheme for a higher accuracy as in Examples 3-6.
Chapter 3

The AMR-IIM for the Stokes equations with interfaces

In this chapter, we use the adaptive version of immersed interface method (AMR-IIM) to solve the Stokes equations with a fixed or moving interface. Consider the 2D Stokes equations for an incompressible flow in a rectangular domain with an interface $\Gamma$:

\[
\begin{align*}
\nabla p &= \mu \Delta u + g + \int_{\Gamma} f(X(s,t))\delta(x-X(s,t)) \, ds \\
\nabla \cdot u &= 0,
\end{align*}
\]  

(3.1)

where $u$ is the fluid velocity, $p$ is the pressure, $\mu$ is the fluid viscosity, $g$ is an external force, and $f(X(s,t))$ is the source strength of the force defined only on the interface. For moving interface problem, we also have

\[
\frac{d\Gamma}{dt} = u(X(s,t)),
\]  

(3.2)
The IIM for Stokes flow problems was first studied in [33], where the jump conditions of Stokes equations were derived. The elastic band problem and creeping bubble problem in Stokes flow were also discussed there. After that, more jump relations were derived in [24, 25] for the case of discontinuous fluid viscosity across interface. The augmented IIM was used for that case because the jump conditions of pressure and velocity are coupled together [40, 42].

In this chapter, we study the Stokes equations with an interface immersed in a fluid with constant viscosity. The Stokes equations can be reduced to three Poisson equations. Each one has jump conditions given by the singular source $f$ or the external force $g$. Hence, we need to call the elliptic solver in Chapter 2 three times. In addition, we need to compute the gradient of computed pressure on the multi-level mesh to connect the three Poisson equations.

We also develop how to couple the AMR-IIM with the level set method for moving interface problems. We briefly review how to use the level set method for the interface evolution, and then describe the implementation of AMR-IIM into the level set framework. We use the surface tension problem and bubble creeping problem to show the performance of AMR-IIM for the moving interface cases.
3.1 The three-Poisson approach for the Stokes equations

Consider the 2D Stokes equations in smooth case:

\[
\begin{cases}
    p_x = \mu (u_{xx} + u_{yy}) + g_1 \\
    p_y = \mu (v_{xx} + v_{yy}) + g_2 \\
    u_x + v_y = 0
\end{cases}
\]  \hspace{1cm} (3.3)

Differentiate the first equation with respect to \( x \), the second equation with respect to \( y \), add them together and eliminate the derivatives of \( u \), \( v \) by the third equation \( u_x + v_y = 0 \) to obtain a Poisson equation of \( p \): \( p_{xx} + p_{yy} = (g_1)_x + (g_2)_y \). Once \( p \) is known, the first and second equations in (3.3) become two independent Poisson equations of \( u \) and \( v \) respectively.

Now consider the Stokes equations with an interface and discontinuities. In [33], the three-Poisson approach was used and the jump conditions were derived for each Poisson equation, so that the IIM can be applied to solve them. See (3.4) to (3.6).

\[
\begin{cases}
    p_{xx} + p_{yy} = (g_1)_x + (g_2)_y \\
    [p] = \hat{f}_1 \\
    [p_n] = \frac{\partial \hat{f}_2}{\partial \eta} + [\mathbf{g} \cdot \mathbf{n}]
\end{cases}
\]  \hspace{1cm} (3.4)
\[
\begin{aligned}
\mu(u_{xx} + u_{yy}) &= p_x - g_1 \\
[u] &= 0 \\
[\mu u_n] &= \hat{f}_2 \sin \theta \\
\end{aligned}
\] (3.5)

\[
\begin{aligned}
\mu(v_{xx} + v_{yy}) &= p_y - g_2 \\
[v] &= 0 \\
[\mu v_n] &= -\hat{f}_2 \cos \theta \\
\end{aligned}
\] (3.6)

Above $\hat{f}_1$ and $\hat{f}_2$ are the singular forces transformed in normal and tangential directions of the interface. We use the interface local coordinate $\xi$-$\eta$ for the normal and tangential directions, see Figure 3.1. The normal direction is denoted as $n = (\cos \theta, \sin \theta)$.

Figure 3.1: (a) The local coordinate $\xi$-$\eta$ defined by interface normal direction and tangential direction, where $\theta$ is the angle between the $x$-axis and the normal directions. (b) The singular force $(f_1, f_2)$ in horizontal and vertical directions are transformed to $(\hat{f}_1, \hat{f}_2)$ in normal and tangential directions.
Usually we know $g_1$, $g_2$, $\hat{f}_1$, and $\hat{f}_2$ either from the partial differential equations or from the physics of problems. Thus we can solve the three Poisson equations for $p$, $u$, $v$ respectively on the AMR mesh generated in Section 2.1 by the FD schemes described in Section 2.3.

**Numerical schemes for gradient $p$**

Since the Stokes equations can be reduced to three Poisson equations, most of the work can be covered by Chapter 2 except how to compute the gradient $p$ in (3.5)-(3.6) on the multi-level mesh with an interface. Take $p_x$ as an example. Once the numerical solution $P_{ij}$ is solved from (3.4), we should pay special attention in computing $(P_x)_{ij}$ when it is close to the interface or it lies on border of two different mesh resolutions.

When we compute $p_x$ around the interface within the finest mesh level in resolution $h$, there are no more than three cases as described in [33, 40].

- If $(x_{i-1}, y_j)$, $(x_i, y_j)$, $(x_{i+1}, y_j)$ are on the same side of the interface, we use standard central difference with $O(h^2)$ accuracy:

  \[
  (P_x)_{ij} \approx \frac{1}{2h} (P_{i+1,j} - P_{i-1,j}) \tag{3.7}
  \]

- If the interface cuts the interval $[x_{i-1}, x_{i+1}]$ only once, we use one-side scheme with $O(h)$ accuracy:

  \[
  (P_x)_{ij} \approx \begin{cases} 
  \frac{1}{h} (P_{ij} - P_{i-1,j}) & \text{if } (x_{i-1}, y_j) \text{ and } (x_i, y_j) \text{ on the same side} \\
  \frac{1}{h} (P_{i+1,j} - P_{i,j}) & \text{if } (x_i, y_j) \text{ and } (x_{i+1}, y_j) \text{ on the same side}
  \end{cases} \tag{3.8}
  \]

- If $(x_{i-1}, y_j)$ and $(x_{i+1}, y_j)$ lie on opposite sides of $(x_i, y_j)$, we have to use the jump
conditions across interface to obtain $O(h)$ accurate approximation of $p_x$.

\[
(P_x)_{ij} \approx \begin{cases} 
\frac{P_{ij} - P_{ij} - [p] - [p_x](x_i - x^*) - [p_y](y_j - y^*)}{x_i - x_l} & \text{if } \varphi(i,j) > 0 \\
\frac{P_{ij} - P_{ij} + [p] + [p_x](x_i - x^*) - [p_y](y_j - y^*)}{x_i - x_l} & \text{if } \varphi(i,j) < 0
\end{cases} \tag{3.9}
\]

where $(x^*, y^*)$ is the projection of the irregular grid point $(x_i, y_j)$ on the interface. $x_l(l = i + 1 \text{ or } i - 1)$ is the grid point closer to projection $(x^*, y^*)$. $[p]$, $[p_x]$ and $[p_y]$ are jumps on $(x^*, y^*)$, which can be obtained from the local coordinate transform:

\[
[p_x] = [p_\xi] \cos \theta - [p_\eta] \sin \theta \tag{3.10}
\]

\[
[p_y] = [p_\xi] \sin \theta + [p_\eta] \cos \theta \tag{3.11}
\]

where $[p_\xi]$, $[p_\eta]$ are the jumps of normal and tangential derivatives of $p$, $(\cos \theta, \sin \theta)$ is the normal direction at $(x^*, y^*)$ on the interface.

The similar idea is also used in Theorem 3 in Chapter 4 for Navier-Stokes equations. In that case, the second order jump conditions are also used so that the accuracy of $(P_x)_{ij}$ or $(P_y)_{ij}$ is improved to $O(h^2)$.

When we compute $p_x$ or $p_y$ for a hanging node where we lack the grid point from one side to approximate $p_x$ or $p_y$, we can alternatively use the average of approximations of its vertical or horizontal neighbor grid points, see Figure 3.2 for example. To compute $p_x$ at the grid point 8, we can use:

\[
(p_x)_8 \approx \frac{1}{2}((p_x)_3 + (p_x)_{11}) \tag{3.12}
\]
To compute $p_y$ at the grid point 12, we can use:

\[
(p_y)_{12} \approx \frac{1}{2}((p_y)_{11} + (p_y)_{13})
\]  

(3.13)

with $O(h^2)$ accuracy. We can also use the ghost point interpolation described in Section 2.3 to obtain the missing grid point, and then plug it into the central FD scheme for $p_x$ or $p_y$.

Figure 3.2: A part of an adaptive mesh including the interface (red curve), the fine level mesh, and the coarse mesh. The $p_y$ at hanging point 12 can be approximated by the average of $p_y$ at 11 and $p_y$ at 13. The $p_y$ at 12 can also be computed after ghost points $g_1$ and $g_2$ are interpolated.

### 3.2 The level set method and interface evolution techniques

The level set method was first proposed in [53] to capture the front of moving interface. After that, the IIM has been successfully used with level set method for many application
problems, such as the Hele-Shaw problem [20], crystal growth problem [45] and the interfacial flows with surfactant problem [72].

3.2.1 The interface evolution by the level set method

In the level set method, the interface \( \Gamma \) is implicitly represented by the zero level set of a Lipschitz continuous function \( \varphi(x, y, t) \),

\[
\Gamma = \left\{ (x, y), \varphi(x, y, t) = 0 \right\}.
\] (3.14)

Because \( \varphi(x, y, t) \equiv 0 \) on the front of interface evolution, we have \( \frac{\partial}{\partial t} \varphi(x, y, t) \equiv 0 \). By chain rule, \( \varphi_t + \varphi_x x_t + \varphi_y y_t = 0 \). Since the velocity of the moving interface \( \mathbf{u} = (x_t, y_t) \), we have the equation for the level set evolution, which is a Hamilton-Jacobi type equation:

\[
\varphi_t + \mathbf{u} \cdot \nabla \varphi = 0.
\] (3.15)

or

\[
\varphi_t + |\nabla \varphi| V_n = 0
\] (3.16)

where \( V_n = \mathbf{u} \cdot \mathbf{n} \) is the velocity in the interface normal direction \( \mathbf{n} = \nabla \varphi / |\nabla \varphi| \).

For a spatial discretization of (3.16), we use the third order WENO(weighted essential non-oscillating) schemes as in [26,27]. For a time discretization, we use the first order Euler scheme for simplicity. We also implement Adams-Bashforth scheme and TVD Runge-Kutta scheme [16] for a higher order temporal accuracy.

As suggested in [2], we update the level set only within a narrow band \( |\varphi(x, y, t)| \leq 6h \)
(h is finest mesh resolution) covering the interface for a fast evolution. We control the time step $\Delta t$ to satisfy the CFL condition

$$
\Delta t \leq \alpha \min \left\{ h, \frac{h}{\|u\|_{\infty}} \right\}
$$

where $\alpha$ is a constant coefficient depending on the problem stability.

After the level set function $\varphi(x, y, t)$ is updated in each time step, it is crucial to re-initialize the level set function to obtain an approximate signed distance function $|\nabla \varphi| \sim 1$. This is to avoid the level set function being too flat or steep, so that the interface geometric quantities are accurately computed by $\varphi(x, y, t)$. As in [66], the re-initialization can be done by solving the following hyperbolic equation to a steady state

$$
\varphi_t + S(\varphi)(1 - |\nabla \varphi|) = 0
$$

where $t$ is an artificial time variable, and $S(\varphi)$ is a signed function. $S(\varphi)$ can be smoothed by

$$
S(\varphi) = \frac{\varphi}{\sqrt{\varphi^2 + \epsilon}}
$$

See also [64] and [26] for more details. Our practice shows that the level set evolution based on an adaptive mesh often need more iterations (up to 10) of re-initialization than a uniform mesh. Sometimes, the insufficient re-initialization could cause a blow up in the computation on AMR.
3.2.2 Mass conservation techniques

One drawback of the level-set method is that the enclosed area (or the mass) is not exactly conserved by the moving interface. Usually, a small error in each time step accumulates so that there would be a large area loss in a long run. There have been a number of efforts to improve the area conservation. In [64], a constrained re-initialization method was proposed. In [15], a hybrid particle level set approach was proposed to conserve mass accurately. In [65], the level-set method and the volume of fluid method were coupled together to achieve the area conservation. In [72], the reason of the area loss is believed to be that the computed velocity field is not exactly divergence free. An correction term $\alpha$ is added to the computed normal velocity to satisfy $\nabla \cdot (u + \alpha n) = 0$. We implement that correction term into our programs but find that there is still up to 5% area loss.

Here we develop a straightforward mass conservation method by adding a correction term directly on the level set function $\phi$ in order to make up the leaked area compared against the initial area. Before the evolution starts, we compute the initial enclosed area as a benchmark. In each time step, we first move the interface by the Hamilton-Jacobi equation (3.16), and then find the area loss against the initial area benchmark. We also compute the length of interface at that step, in order to determine how much correction we need to expand or compress the interface in its normal direction. Finally, we correct the level set within a narrow tube do re-initialization again. See also Figure 3.3.

According to [60], we compute the enclosed area by the following formula

$$A = \int_{\Omega} H(-\varphi(x,y,t))dxdy$$  \hspace{1cm} (3.20)
Figure 3.3: Expand the interface by $\epsilon$ in the normal direction to conserve the enclosed area against the initial area $A_0$.

where $H$ is the Heaviside function defined by

$$H_\epsilon(x) = \begin{cases} 0 & \text{if } x < -\epsilon \\ \frac{1}{2}(1 + \frac{x}{\epsilon} + \frac{1}{\pi} \sin \frac{\pi x}{\epsilon}) & \text{if } -\epsilon \leq x \leq \epsilon \\ 1 & \text{if } x > \epsilon \end{cases}$$

(3.21)

We also compute the length of the interface by the formula

$$L = \int_{\Omega} \delta(\varphi(x, y, t)) |\nabla \varphi(x, y, t)| dxdy$$

(3.22)

where $\delta$ is Peskin’s discrete cosine delta function

$$\delta_\epsilon(x) = \begin{cases} \frac{1}{4\epsilon}(1 + \cos \frac{\pi x}{2\epsilon}) & \text{if } |x| < 2\epsilon \\ 0 & \text{if } |x| \leq 2\epsilon \end{cases}$$

(3.23)

We take $\epsilon$ to be the finest mesh resolution $h$ in above (3.21) and (3.23).
Below we show the procedure for the mass conservation method. Suppose we use the initial enclosed area $A_0$ as a benchmark. In each time step $t_n$:

1. Update the level set by the Hamilton-Jacobi equation (3.16) in a narrow band $|\varphi| \leq 6h$ around the interface.

2. Do the level set re-initialization

3. Compute the enclosed area $A_n$ by (3.20) and the interface length $L_n$ by (3.22)

4. According to the area loss $A_0 - A_n$, we need to expand the computed interface in the normal direction by $\epsilon = \frac{A_0 - A_n}{L_n}$. If $A_0 < A_n$, $\epsilon < 0$, we should compress the interface by $\epsilon$ to reduce the enclosed area.

5. Make the correction in the narrow band $|\varphi| \leq 6h$ by $\varphi_{ij} = \varphi_{ij} - \epsilon$, since $|\nabla \varphi| \sim 1$ after the re-initialization in Step 2

6. Do the level set re-initialization again

Our numerical experiments show that the mass loss after this correction is always less than 0.01% over long run of moving interface for the examples in Section 3.4.2 and 3.4.3.

### 3.3 Coupling the AMR with the level set method

Since our AMR approach is also based on the level set representation of the interface, it can be easily implemented into the above level set algorithms for the interface evolution. The basic idea is to first solve the velocity $(u, v)$ from Stokes equations on the adaptive mesh, then use the velocity to update the level set function $\varphi$ by solving the Hamilton-Jacobi equation, and then generate a new adaptive mesh based on the updated level set function if necessary.
As in Section 3.2.1, the level set is updated only within a narrow tube around interface $|\varphi(x, y, t)| \leq \lambda_1 h$, where $h$ is the finest mesh resolution. To guarantee that the level set is updated within the finest level mesh, we generate the finest level mesh from a wider tube $|\varphi(x, y, t)| \leq \lambda_2 h$ ($\lambda_2 > \lambda_1$) using the method described in Section 2.1. We do not generate new AMR in each time step. We only update the AMR when the level set updating tube $|\varphi(x, y, t)| \leq \lambda_1 h$ tends to move out of the current finest mesh level. There are two advantages to do this. First, we save the efforts to generate a new AMR and book-keep grid points in each time step. Usually, we only need to update AMR after dozens or even hundreds of time steps. Second, when we work on the same AMR for several time steps, we can input the solutions of $p, u, v$ in current step as good predictions of the solutions in next step. Our numerical experiments show that this saves around half computation time in the multigrid solver, compared to using a random guess in each time step.

After the level set is updated in each step, we do the mass correction as in Section 3.2.2 inside the level set updating tube $|\varphi(x, y, t)| \leq \lambda_1 h$. After that, we do the level set re-initialization within a wider tube $|\varphi(x, y, t)| \leq \lambda_3 h$ ($\lambda_3 > \lambda_2$) to cover both the level set updating tube and the refined mesh, so that the AMR generated later can track the new interface accurately. The re-initialization can also avoid the pollution error for the cut-off boundary of the computational tube.

In most of our work, we set up above three tubes as following: $|\varphi(x, y, t)| \leq 6h$ for the level set function updating, $|\varphi(x, y, t)| \leq 8h$ for the AMR generation, and $|\varphi(x, y, t)| \leq 12h$ for the level set re-initialization. Figure 3.4 shows the interface and the three tubes.

We outline our AMR-IIM for the moving interface iteration in below Algorithm 2 and the flow chart in Figure 3.5.

We will use similar procedure in Chapter 4 to solve Navier-Stokes equations for moving interface problems, except that we solve $p, u, v$ by projection-IIM method in Step 4.
Adaptive refinement tube $|\phi| < 8h$

Level set re-initialization tube $|\phi| < 12h$

Level set updating tube $|\phi| < 6h$

Interface

Figure 3.4: The interface (orange dashed), the level set update tube (red), the adaptive refinement tube (blue) and the level set re-initialization tube (green) for moving interface problems. $h$ is the finest the mesh resolution.

Figure 3.5: The flow chart of solving Stokes equations on AMR for moving interface problems.
Algorithm 2 Solve Stokes equations by AMR-IIM for moving interface problems

1: Start with an initial level set \( \varphi(x, y, 0) \), and compute the enclosed area \( A_0 \).

2: Generate the first AMR from the tube \( |\varphi(x, y, 0)| \leq \lambda_2 h \).

3: \textbf{while} \( t < t_{\text{stop}} \) \textbf{do}

4: \hspace{1em} Obtain the jump conditions from the most updated interface \( \varphi(x, y, t) = 0 \), and then solve the three Poisson equations for \( p, u, v \) on the most updated AMR.

5: \hspace{1em} Use the velocity \( (u, v) \) to update the level set \( \varphi(x, y, t) \) by solving the Hamilton-Jacobi equation \( \varphi_t + u \cdot \nabla \varphi = 0 \) within a narrow band \( |\varphi(x, y, t)| \leq \lambda_1 h \), where \( \lambda_1 < \lambda_2 \) to guarantee that the interface moves within the finest level mesh.

6: \hspace{1em} Make the area correction on \( \varphi(x, y, t) \) within the level set updating band \( |\varphi(x, y, t)| \leq \lambda_1 h \) against the initial enclosed area \( A_0 \).

7: \hspace{1em} Re-initialize the level set function within a wide tube \( |\varphi(x, y, t)| \leq \lambda_3 h \), where \( \lambda_3 > \lambda_2 \) to cover all refinement regions.

8: \hspace{1em} \textbf{if} the level set updating tube \( |\varphi(x, y, t)| \leq \lambda_1 h \) tends to move beyond the finest level mesh \textbf{then}

9: \hspace{2em} Generate a new AMR based on the most updated level set \( |\varphi(x, y, t)| \leq \lambda_2 h \)

10: \hspace{1em} \textbf{end if}

11: \hspace{1em} \( t = t + \Delta t \)

12: \textbf{end while}
3.4 Numerical experiments and analysis

In this section, we first present an example to test the AMR-IIM applied on Stokes equations with a fixed interface. And then we use two Stokes flow problems to show how the AMR-IIM works in moving interface problems.

3.4.1 The Stokes equations with a fixed interface

We consider the example with a fixed circular interface \( x^2 + y^2 = 1 \) in the domain \([-2, 2] \times [-2, 2]\). This example is used in [25, 40] to illustrate the jump relations of Stokes equations. The example was created from the below exact solution:

\[
\begin{align*}
 u(x, y) &= \begin{cases} 
 y/4 & \text{if } x^2 + y^2 \leq 1 \\
 y/4(x^2 + y^2) & \text{otherwise}
\end{cases} \\
 v(x, y) &= \begin{cases} 
 -x/4(1 - x^2) & \text{if } x^2 + y^2 \leq 1 \\
 -xy^2/4 & \text{otherwise}
\end{cases} \\
 p(x, y) &= \begin{cases} 
 (-3/4)x^3 + 3/8y & \text{if } x^2 + y^2 \leq 1 \\
 0 & \text{otherwise}
\end{cases}
\end{align*}
\]
Hence, the external force should be:

\[
g_1(x, y) = \begin{cases} 
(-\frac{9}{4} x^2 + \frac{3}{8})y & \text{if } x^2 + y^2 \leq 1 \\
-2\mu y & \text{otherwise}
\end{cases}
\] (3.27)

\[
g_2(x, y) = \begin{cases} 
-\frac{3}{4} x^3 + \frac{3}{8} x - \frac{3\mu}{2} x & \text{if } x^2 + y^2 \leq 1 \\
\frac{\mu}{2} x & \text{otherwise}
\end{cases}
\] (3.28)

And, the singular force density in the normal and tangential directions are:

\[
\hat{f}_1 = \left(\frac{3}{4} \cos^3 \theta - \frac{3}{8} \cos \theta\right) \sin \theta
\] (3.29)

\[
\hat{f}_2 = \frac{\mu}{2}
\] (3.30)

where \((\cos \theta, \sin \theta)\) is the normal direction of the interface \(x^2 + y^2 = 1\), and \(\mu\) is the fluid viscosity on both sides of the interface. Here we take \(\mu = 1\) for simplicity.

According to (3.4)-(3.6), the Stokes equations can be reduced to following three Poisson equations with jump conditions:

\[
p_{xx} + p_{yy} = (g_1)_x + (g_2)_y = \begin{cases} 
-\frac{9}{2} xy & \text{if } x^2 + y^2 \leq 1 \\
0 & \text{otherwise}
\end{cases}
\]

\[
[p] = \hat{f}_1 = \left(\frac{3}{4} x^3 - \frac{3}{8} x\right)y
\]

\[
[p_n] = \frac{\partial \hat{f}_2}{\partial \eta} + [g \cdot n] = 3x^3y - \frac{3}{4} xy
\]
\[ \begin{cases} 
\mu(u_{xx} + u_{yy}) = p_x - g_1 = \\
\begin{cases} 
0 & \text{if } x^2 + y^2 \leq 1 \\
2\mu y & \text{otherwise} 
\end{cases} 
\end{cases} \]

\[ [u] = 0 \]

\[ [\mu u_n] = \hat{f}_2 \sin \theta = \frac{1}{2} \mu \sin \theta \]

\[ \begin{cases} 
\mu(v_{xx} + v_{yy}) = \\
\begin{cases} 
\frac{3}{2} \mu x & \text{if } x^2 + y^2 \leq 1 \\
-\frac{\mu}{2} x & \text{otherwise} 
\end{cases} 
\end{cases} \]

\[ [v] = 0 \]

\[ [\mu v_n] = -\hat{f}_2 \cos \theta = -\frac{\mu}{2} \cos \theta \]

We solve this problem by the IIM on both uniform \( m \times m \) mesh and AMR \( m(+1)[2] \) mesh. In Table 3.1 and 3.2, we show the errors \( \| E_u \|_\infty, \| E_v \|_\infty \) and \( \| E_p \|_\infty \) against exact solutions through a grid refinement analysis, where the order of accuracy defined by

\[ \text{order} = \frac{\log(E(m)/E(2m))}{\log 2} \]

The columns "Size" show the numbers of grid points to be solved for the each of the three Poisson equations.

Table 3.1: A grid refinement analysis by uniform mesh for Stokes equations with a fixed interface

<table>
<thead>
<tr>
<th>Uniform ( m \times m )</th>
<th>Size</th>
<th>( | E_p |_\infty )</th>
<th>Order</th>
<th>( | E_u |_\infty )</th>
<th>Order</th>
<th>( | E_v |_\infty )</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m = 80 )</td>
<td>6241</td>
<td>1.68e-4</td>
<td>2.55e-4</td>
<td>1.47e-4</td>
<td>2.00e-4</td>
<td>1.37e-4</td>
<td>2.00e-4</td>
</tr>
<tr>
<td>( m = 160 )</td>
<td>25281</td>
<td>4.06e-5</td>
<td>2.05</td>
<td>6.94e-5</td>
<td>1.88</td>
<td>3.58e-5</td>
<td>2.04</td>
</tr>
<tr>
<td>( m = 320 )</td>
<td>101761</td>
<td>6.38e-6</td>
<td>2.67</td>
<td>1.72e-5</td>
<td>2.01</td>
<td>8.29e-6</td>
<td>2.11</td>
</tr>
<tr>
<td>( m = 640 )</td>
<td>408321</td>
<td>1.59e-6</td>
<td>2.00</td>
<td>4.50e-6</td>
<td>1.93</td>
<td>2.12e-6</td>
<td>1.97</td>
</tr>
</tbody>
</table>
From Table 3.2, we confirm that the AMR-IIM maintains the second order accuracy for \( p, u, v \) in Stokes equations. Compare to the results in Table 3.1 with the same finest resolution, the AMR obtains the similar accuracy but saves more than 50\% grid points to be solved. This is consistent with the results in Chapter 2 for elliptic equations.

Table 3.2: A grid refinement analysis by adaptive mesh for Stokes equations with a fixed interface

<table>
<thead>
<tr>
<th>AMR ( m(+1)[2] )</th>
<th>Size</th>
<th>( |E_p|_\infty )</th>
<th>Order</th>
<th>( |E_u|_\infty )</th>
<th>Order</th>
<th>( |E_v|_\infty )</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>40(+1)[2], ( \lambda = 2 )</td>
<td>2745</td>
<td>1.68e-4</td>
<td>1.68e-4</td>
<td>2.95e-4</td>
<td>1.43e-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80(+1)[2], ( \lambda = 4 )</td>
<td>10201</td>
<td>4.07e-5</td>
<td>2.05</td>
<td>8.06e-5</td>
<td>1.87</td>
<td>3.52e-5</td>
<td>2.02</td>
</tr>
<tr>
<td>160(+1)[2], ( \lambda = 8 )</td>
<td>39225</td>
<td>6.38e-6</td>
<td>2.67</td>
<td>2.14e-5</td>
<td>1.91</td>
<td>8.22e-6</td>
<td>2.10</td>
</tr>
<tr>
<td>320(+1)[2], ( \lambda = 16 )</td>
<td>153805</td>
<td>1.59e-6</td>
<td>2.00</td>
<td>5.32e-6</td>
<td>2.00</td>
<td>2.11e-6</td>
<td>1.96</td>
</tr>
</tbody>
</table>

3.4.2 The surface tension problem

In this example, we use the Stokes equations to simulate the motion of a distorted interface immersed in a liquid. The interface is initially distorted as a star shape. Due to the surface tension, the interface will relax to its equilibrium state, which should be a circle with the same area enclosed by the initial interface. When the fluid viscosity is large enough, this motion can be modeled by Stokes equations. When the fluid viscosity is small, we use Navier-Stokes equations to model this motion in Chapter 4.

There is no external force in this example, so that \( g_1 = g_2 = 0 \). The only driven force on the fluid is from the interface surface tension. According to Young-Laplace equation in fluid dynamics, the jump in pressure is proportional to the surface tension coefficient \( \gamma \) and the interface curvature \( \kappa \): \( [p] = \gamma \kappa \). Hence we obtain the jump conditions for the
Stokes equations:

\[
\begin{align*}
\hat{f}_1 &= \gamma \kappa, \quad \hat{f}_2 = 0
\end{align*}
\] (3.34)

where \( \gamma \) is the surface tension coefficient and \( \kappa \) is the interface curvature.

We set up the initial shape of interface to be

\[
r(\theta) = 0.6 + 0.2 \sin(5\theta), \quad 0 \leq \theta \leq 2\pi
\] (3.35)

in the domain \([-1, 1] \times [-1, 1]\). The surface tension coefficient \( \gamma = 0.1 \), and fluid viscosity \( \mu = 1 \). The Dirichlet boundary conditions are simply \( p|_{\partial \Omega} = u|_{\partial \Omega} = v|_{\partial \Omega} = 0 \), see Figure 3.6 and 3.7 for the initial pressure color profile and velocity field.

We first solve this problem on a AMR 80(+1)[2] mesh. We take the time step to be \( \Delta t = \frac{1}{8} \min\{h, h/\|u\|_\infty\} \) to satisfy the CFL condition. In Figure 3.8, we show some
Figure 3.7: Velocity field

snap-shots of the interface motion (red solid) and the underlying adaptive meshes. It can be seen that the adaptive refinement region can successfully track the moving interface. The computation records show that the AMR is updated only 6 times to the end. The interface moves for hundreds of time steps within each generated AMR until the level set updating tube $|\varphi(x, y, t)| \leq 6h$ tends to touch the border of the refinement region.

In Table 3.3, we present the results in detail in the four moments corresponding to the four snap-shots in Figure 3.8. As we expected, the magnitude of curvature $|\kappa|$ decreases to the curvature of a stable circle, and the velocity decreases to zero. The area enclosed by the interface is conserved very well with less than 0.01% area loss.

The advantage of AMR can be shown from the last column in Table 3.3 about the numbers of grid points in the four snap-shots. By AMR 80(+1)[2], we compute with 11000-12000 grid points to obtain the finest resolution of $2/160$ around the interface. If we use the uniform mesh for the same finest resolution, we need at least $160^2 = 25600$
grid points. The AMR usually helps us to reduce at least half of the computation load without loss of accuracy.

Figure 3.8: Snap-shots of a deformable interface (red-solid in the middle of the fine mesh) relaxing to its equilibrium state computed by the AMR mesh 80(+1)[2].

Then, we do the same problem by adding two levels of refinement based on a 40 × 40 uniform mesh. Figure 3.9 shows that the AMR 40(+2)[2] can also track the moving interface successfully and save even more grid points. In that case, we still obtain the highest
Table 3.3: Critical physical quantities solved by AMR 80(+1)[2] at the four moments corresponding to Figure 3.8 (a)-(d)

<table>
<thead>
<tr>
<th>Figure</th>
<th>$t$</th>
<th>$\kappa_{max}$</th>
<th>$p_{max}$</th>
<th>$|u|_{max}$</th>
<th>Area</th>
<th>Grid points</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.001</td>
<td>28.40</td>
<td>1.6010</td>
<td>0.2455e-1</td>
<td>1.194</td>
<td>11595</td>
</tr>
<tr>
<td>b</td>
<td>6.250</td>
<td>5.864</td>
<td>0.4282</td>
<td>0.1611e-1</td>
<td>1.194</td>
<td>11687</td>
</tr>
<tr>
<td>c</td>
<td>12.50</td>
<td>3.276</td>
<td>0.2509</td>
<td>0.6240e-2</td>
<td>1.194</td>
<td>11252</td>
</tr>
<tr>
<td>d</td>
<td>31.25</td>
<td>1.669</td>
<td>0.1645</td>
<td>0.1563e-3</td>
<td>1.194</td>
<td>11088</td>
</tr>
</tbody>
</table>

We also do some error refinement analysis to test if the AMR can preserve the second order accuracy of the IIM for Stokes equations. Since the exact solution is not available, we compare the solutions when $m = 40, 80, 160$ with the solution when $m = 320$. We can get the order of accuracy from the ratios of successive errors. As in [33], for a $q$th-order accurate method, the following ratio

$$\frac{\|p_{40} - p_{320}\|}{\|p_{80} - p_{320}\|} \approx \frac{C(1/40)^q - C(1/320)^q}{C(1/80)^q - C(1/320)^q} = \frac{8^q - 1}{4^q - 1}$$

should be 2.3 for a first order method and 4.2 for a second order method. Similarly, the ratio

$$\frac{\|p_{80} - p_{320}\|}{\|p_{160} - p_{320}\|} \approx \frac{C(1/80)^q - C(1/320)^q}{C(1/160)^q - C(1/320)^q} = \frac{4^q - 1}{2^q - 1}$$

should be 3 for a first order method and 5 for a second order method. We computed the solution by AMR with the base coarse mesh $m = 40, 80, 160$ and 320 and then got some ratios of successive errors. Table 3.4 shows that our AMR-IIM method maintains the second order accuracy of the IIM in this example.
Figure 3.9: Snap-shots of a deformable interface (red-solid in the middle of the fine mesh) relaxing to its equilibrium state computed by the AMR mesh 40(+2)[2].

Table 3.4: The errors of $p$, $u$, $v$ computed by AMR $m(+1)[2]$ compared to the benchmarks computed by AMR 320(+1)[2] at $t = 0$

<table>
<thead>
<tr>
<th>$m$</th>
<th>$|p_m - p_{320}|_\infty$</th>
<th>$|u_m - u_{320}|_\infty$</th>
<th>$|v_m - v_{320}|_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>3.23e-2</td>
<td>5.66e-3</td>
<td>2.30e-3</td>
</tr>
<tr>
<td>80</td>
<td>7.25e-3</td>
<td>4.45</td>
<td>1.32e-3</td>
</tr>
<tr>
<td>160</td>
<td>9.51e-4</td>
<td>7.62</td>
<td>2.94e-4</td>
</tr>
</tbody>
</table>
3.4.3 The bubble creeping problem

In this example, we use the Stokes equations to simulate a bubble creeping through a fluid. The motion is due to the difference in the fluid density inside the bubble $\rho_{in}$, and the fluid density outside the bubble $\rho_{out}$. We study the case $\rho_{in} < \rho_{out}$ in which the bubble rises and meanwhile its shape is distorted by the surface tension and the gravity. We assume the same viscosity $\mu$ for both fluids.

The gravity serves as the vertical external force in this example: $g_2 = -\rho_{in}g$ inside and $g_2 = -\rho_{out}g$ outside the bubble, where $g = 9.8 \text{ m/s}^2$. There is no horizontal external force: $g_1 = 0$. The singular driven forces are the same as the last example: $\hat{f}_1 = \gamma \kappa$ and $\hat{f}_2 = 0$, where $\gamma$ is surface tension coefficient and $\kappa$ is the interface curvature. From the jump conditions of Stokes equations in (3.4), here we have:

$$p_{xx} + p_{yy} = -\rho_y g = 0 \quad (3.38)$$

on $\Omega \setminus \Gamma$, and the jump conditions:

$$[p] = \gamma \kappa, \quad [p_n] = [\rho]g \sin \theta \quad (3.39)$$

We set up parameters as follows. The fluid densities are $\rho_{in} = 1$ inside bubble, and $\rho_{out} = 2$ outside bubble. The viscosity $\mu = 1$ on both sides of the bubble. We work on the domain $[-4, 4] \times [-4, 4]$ where the initial shape of bubble is set up to be an ellipse

$$\frac{x^2}{0.6^2} + \frac{(y + 3.2)^2}{0.25^2} = 1 \quad (3.40)$$

The Dirichlet boundary conditions are set to be $u|_{\partial \Omega} = v|_{\partial \Omega} = 0$, and $p|_{\partial \Omega} = \rho_{out}g(4 - y)$.
because the hydrostatic pressure is linear to the fluid depth.

It is known that the deformation of the bubble depends on its surface tension. See for examples in [4, 13, 61]. For a large surface tension $\gamma$, the bubble tends to restore to a circle as it rises. For a small surface tension $\gamma$, the bubble tends to be distorted. If $\gamma$ is sufficiently small, the bubble will eventually split into two pieces.

We test the AMR-IIM in four cases with various surface tension coefficients and obtain the expected bubble deformations, see Figure 3.10. The mass conservation techniques in Section 3.2.2 is also shown to be accurate. In all cases, the area loss is less than 0.01% till the end of the simulation.

Among the four cases above, we are more interested in the last case when the bubble splits, because it shows the advantage of the level set method in dealing with topological changes. Figure 3.11 shows several snap-shots of the bubble and the underline meshes AMR 80(+1)[2] as the bubble rises. The refinement region can automatically follow both the moving of the bubble position and changing of the bubble shape. We still take the time step $\Delta t = \frac{1}{8} \min\{h, h/\|u\|_{\infty}\}$. The AMR is updated every 30 steps on average. When the bubble splits into two pieces, the AMR also tend to split. The advantage of the AMR for this problem is more significant than last surface tension example. To obtain a finest resolution $4/160$ around the interface, the adaptive mesh requires around 7000 grid points, whereas the uniform mesh requires $160^2 = 25600$ grid points.

Sometimes the level set method is not accurate enough when the interface curvature is very large, for example, in the region where the bubble just split. A finer mesh resolution is usually needed to catch the interface evolution more accurately. For example, we can add one more layer of finer mesh locally based on Figure 3.11, instead of refining the whole domain uniformly, see Figure 3.12 where the finest resolution becomes $4/320$.

We believe that the AMR can be more advantageous for problems in which the sim-
Figure 3.10: Creeping bubbles with various surface tension $\gamma$ at time 0s, 0.60s, 1.1s, 1.6s, 2.2s, 2.8s. The loss of mass is less than 0.01% in all four cases.
Figure 3.11: A rising bubble (the red solid curve in the middle of the fine mesh) at time $t = 0s$, 0.720s, 1.30s, 2.28s, 3.64s, computed by AMR 80(+1)[2]
Figure 3.12: A rising bubble (the red solid curve in the middle of the fine mesh) at time $t = 0s, 0.720s, 1.30s, 2.28s, 3.64s$, computed by AMR 80(+2)\cite{2}
ulated object is relatively small but could move throughout a large domain. In those cases, it might be impossible to refine the whole domain uniformly due to the restriction of computer storage. The AMR can overcome this difficulty by utilizing the limited computational resource more efficiently.

We are also interested in whether the creeping bubble simulation computed by the AMR matches that by a uniform mesh. The bubble creeping problem computed by the IIM on a uniform mesh was studied for more than ten years, such as in [33] and [43], and the results are believed to be reliable. We use the AMR 80(+1)[2] mesh and then compare the computed bubble interface with that computed by the uniform 160 × 160 mesh with the same finest resolution. The initial bubble is set up to be ellipse \( \frac{x^2}{0.6^2} + \frac{(y + 3)^2}{0.2^2} = 1 \) and the surface tension \( \gamma = 0.01 \), see Figure 3.13 for a comparison of the bubble interfaces computed by the uniform mesh and by the AMR at \( t = 3s \). We find that both the bubble shape and position computed by AMR match those by uniform mesh very well.

![Figure 3.13: The bubble interface at \( t = 3s \) computed by the uniform mesh 160 × 160 and the AMR 80(+1)[2]](image-url)
Chapter 4

The AMR-IIM for the Navier-Stokes equations with interfaces

In this chapter, we consider the full Navier-Stokes equations in 2D for an incompressible flow including an immersed interface $\Gamma$:

$$
\begin{align*}
\rho \left( \frac{\partial u}{\partial t} + (u \cdot \nabla) u \right) + \nabla p &= \mu \Delta u + g + \int_{\Gamma} f(X(s,t)) \delta(x - X(s,t)) \, ds \\
\nabla \cdot u &= 0
\end{align*}
$$

in a rectangular domain $R$ with a known boundary condition on $\partial R$. $u$ is the fluid velocity, $\rho$ is the fluid density, $p$ is the fluid pressure, $\mu$ is the fluid viscosity, $g$ is an external force, and $f(X(s,t))$ is a source strength of the force defined only on the interface. For simplicity we assume $\rho = 1$ and $\mu$ is constant in the domain.

For moving interface problems, we also have

$$
\frac{d\Gamma}{dt} = u(X(s,t)),
$$

(4.2)
The Navier-Stokes equations can be solved by the well-known projection method. See [6,10,11] for examples. In the projection method, the Helmholtz-Hodge Decomposition is applied to decouple the velocity and pressure, and then the Navier-Stokes equations can be reduced to two Helmholtz equations and one Poisson equation. When an interface is involved, Peskin used his Immersed Boundary(IB) method to smooth the singular term in (4.1). See [55]. In [41], Li and Lai used the Immersed Interface Method(IIM) to solve the Navier-Stokes equations by adding correction terms to incorporate jump conditions. The IIM is also used to solve Navier-Stokes equations for the case of discontinuous viscosity [68], and the case of irregular domain [23]. Besides, Lee and LeVeque [29] developed a method to use the IIM for Navier-Stokes equations by adding correction terms only for the pressure.

Similar to Chapter 3, we use the AMR-IIM to solve Navier-Stokes equations by calling the AMR-IIM elliptic solver in Chapter 2 three times, two for Helmholtz equations to compute an intermediate velocity, and one for Poisson equation to update the pressure. Compare to Chapter 3, we need more efforts to compute discrete gradients and discrete Laplacians in the projection method among different mesh levels. For moving interface problems, we still use the level set method and exactly the same interface evolution techniques.

In this chapter, we first review how to apply the IIM to add correction terms upon the regular projection method for interface problems. After that, we describe how to implement the projection-IIM method on an adaptive mesh. Finally, we use a fixed interface example to show the method accuracy, and then use the the surface tension example again to see how the results are different from those solved by Stokes equations in Chapter 3. It is well known that Stokes equations is a special case of the full Navier-Stokes equations when the fluid Reynolds number is low enough, or in other word, the inertial
term \( \rho \left( \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) \) in (4.1) can be ignored compared to a much larger viscosity term \( \mu \Delta \mathbf{u} \). In this chapter, we will take low viscosity in the Navier-Stokes equations so that the inertia effect and some different patterns of fluid motion can be observed.

### 4.1 The projection-IIM for the Navier-Stokes equations with interfaces

#### 4.1.1 The projection method for the Navier-Stokes equations

Since the pioneering work of Chorin [11], there are several versions of projection method to solve the incompressible Navier-Stokes equations. In this dissertation, we use the one developed in [10] which is based on the pressure increment formulation in [6] with an additional accurate pressure correction.

Assume a Dirichlet boundary condition for the velocity along \( \partial R \) is given. From time \( t^n \) to \( t^{n+1} \), there are two steps. In the prediction step, we solve two Helmholtz equations for an intermediate velocity field \( \mathbf{u}^* \)

\[
\begin{align*}
\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + (\mathbf{u} \cdot \nabla_h \mathbf{u})^{n+1/2} + \nabla_h p^{n-1/2} &= \frac{\mu}{2} (\Delta_h \mathbf{u}^* + \Delta_h \mathbf{u}^n) + \mathbf{g}^{n+1/2}, \\
\mathbf{u}^* |_{\partial \Omega} &= \mathbf{u}_{b}^{n+1}
\end{align*}
\]

(4.3)

where \((\mathbf{u} \cdot \nabla_h \mathbf{u})^{n+1/2}\) is approximated by,

\[
(\mathbf{u} \cdot \nabla_h \mathbf{u})^{n+1/2} = \frac{3}{2}(\mathbf{u}^n \cdot \nabla_h)\mathbf{u}^n - \frac{1}{2}(\mathbf{u}^{n-1} \cdot \nabla_h)\mathbf{u}^{n-1}.
\]

(4.4)

The next is the projection step, which includes a Poisson equation with zero Neumann
boundary condition,

\[
\begin{aligned}
\nabla_h \phi^{n+1} &= \frac{\nabla_h \cdot u^*}{\Delta t}, \\
\frac{\partial \phi^{n+1}}{\partial n} \big|_{\partial \Omega} &= 0
\end{aligned}
\]  

(4.5)

And then the velocity and the pressure are updated by

\[
\begin{aligned}
\nu^{n+1} &= u^* - \Delta t \nabla_h \phi^{n+1} \\
\nabla_h p^{n+1/2} &= \nabla_h p^{n-1/2} + \nabla_h \phi^{n+1} \\
p^{n+1/2} &= p^{n-1/2} + \phi^{n+1} - \frac{\mu}{2} (\nabla_h \cdot u^*)
\end{aligned}
\]  

(4.6-4.8)

Above projection method is designed for smooth flow problems. To apply it to interface problems with discontinuities, we need to use the jump information across the interface to add correction terms for every discrete gradient or discrete Laplacian, such as \( \nabla_h u, \nabla_h p \) and \( \Delta_h u \).

4.1.2 Jump relations and correction terms using the IIM

In this section, we review some theorems from [41] about how to use jump conditions \( \hat{f}_1, \hat{f}_2, [g_1], [g_2] \) to derive correction terms added on \( \nabla_h u, \nabla_h p \) and \( \Delta_h u \). Note that \( \hat{f}_1, \hat{f}_2, [g_1], [g_2] \) are in the same notations as in Chapter 3.

The first theorem tells the basic jump conditions for incompressible Navier-Stokes equations. The jump conditions are are exactly the same as those for incompressible Stokes equations because the material derivative is continuous.

THEOREM 1. Let \((X, Y)\) be a point on the interface. Let \( \Gamma \) be a smooth closed
interface within the domain. Let the unit outward normal direction be \( n = (\cos \theta, \sin \theta) \), where \( \theta \) is the angle between the outward normal direction and the \( x \)-axis. Then we have the jump conditions

\[
[u] = 0, \quad [\mu u_n] = -\hat{f}_2 \tau, \quad (4.9)
\]

\[
[p] = \hat{f}_1, \quad [p_n] = \frac{\partial \hat{f}_2}{\partial s} + [G] \cdot n, \quad (4.10)
\]

where \( \tau = (-\sin \theta, \cos \theta) \) is the unit tangent direction, and \( \hat{f}_1, \hat{f}_2 \) are the force strengths in the normal and tangential directions respectively

\[
\hat{f}_1 = f_1 \cos \theta + f_2 \sin \theta, \quad (4.11)
\]

\[
\hat{f}_2 = -f_1 \sin \theta + f_2 \cos \theta \quad (4.12)
\]

The proof can be found in [24,25,33,40].

The second theorem tells how to use above jump conditions to derive more jump conditions of the first and second derivatives of \( u \) and \( p \) in the local \( \xi - \eta \) coordinate on the interface, see Figure 4.1.

THEOREM 2. We define the local coordinates at \((X,Y)\), a point on the interface, as

\[
\xi = (x - X) \cos \theta + (y - Y) \sin \theta, \quad (4.13)
\]

\[
\eta = -(x - X) \sin \theta + (y - Y) \cos \theta \quad (4.14)
\]

Then the interface \( \Gamma \) can be represented by \( \xi = \chi(\eta) \) in the neighborhood of \((\xi, \eta) = (0,0)\), which satisfies \( \chi(0) = 0, \chi'(0) = 0, \) and \( \chi''(0) = \kappa, \) the curvature of the interface at \((0,0)\).
The following interface relations are true at \((X,Y)\)

\[
[p] = \hat{f}_1, \quad [p_n] = \frac{\partial \hat{f}_2}{\partial s} + [G] \cdot n,
\]

\((4.15)\)

\[
[u] = 0, \quad [\mu u_n] = -\hat{f}_2 \tau, \quad [u_\eta] = 0,
\]

\((4.16)\)

\[
[\mu u_\eta] = \kappa \hat{f}_2 \tau,
\]

\((4.17)\)

\[
[\mu u_\xi] = -\frac{\partial \hat{f}_2}{\partial \eta} - \kappa \hat{f}_2 n,
\]

\((4.18)\)

\[
[\mu u_\xi\eta] = -[\mu u_\eta] + [p_\xi] n + [p_\eta] \tau + [u_\xi] u \cdot n - [G].
\]

\((4.19)\)

And then, we can transform the jumps in \(\xi - \eta\) local coordinate to the jumps in \(x - y\) Cartesian coordinate to be used in correction terms. For example,

\[
[u_x] = [u_\xi] \cos \theta - [u_\eta] \sin \theta,
\]

\((4.20)\)

\[
[u_y] = [u_\xi] \sin \theta + [u_\eta] \cos \theta,
\]

\((4.21)\)

\[
[u_{xx}] = [u_{\xi\xi}] \cos^2 \theta - 2[u_{\xi\eta}] \cos \theta \sin \theta + [u_{\eta\eta}] \sin^2 \theta
\]

\((4.22)\)

\[
[u_{yy}] = [u_{\xi\xi}] \sin^2 \theta - 2[u_{\xi\eta}] \cos \theta \sin \theta + [u_{\eta\eta}] \cos^2 \theta
\]

\((4.23)\)

The third theorem tells how to build correction terms for first and second order finite difference schemes when the interface cuts somewhere in the finite difference stencil, see Figure 4.1. The purpose is to obtain at least \(O(h)\) accurate finite difference schemes.

**THEOREM 3.** Let \(u(x)\) be a piecewise twice differentiable function. Assume that \(u(x)\) and its derivatives have finite jumps \([u]\), \([u_x]\) and \([u_{xx}]\), at \(x^* = x + \alpha h, -1 \leq \alpha \leq 1\),
then the following relations hold,

\[
\frac{u(x + h) - u(x - h)}{2h} = \begin{cases} 
  u'(x) + \frac{C(x, \alpha)}{2h} + O(h^2), & \text{if } 0 \leq \alpha \leq 1 \\
  u'(x) - \frac{C(x, \alpha)}{2h} + O(h^2), & \text{if } -1 \leq \alpha < 0
\end{cases}
\]

(4.24)

\[
\frac{u(x + h) - 2u(x) + u(x - h)}{h^2} = u''(x) + \frac{C(x, \alpha)}{h^2} + O(h),
\]

(4.25)

where

\[
C(x, \alpha) = [u] + [u_x](1 - |\alpha|)h + [u_{xx}](1 - |\alpha|)^2h^2/2.
\]

(4.26)

Figure 4.1: The local coordinate \(\xi-\eta\) defined at \(x^*\) on the interface (red). The correction term for grid point \(x\) can be derived from the jump conditions at \(x^*\)

The proof can be easily obtained by taking Taylor expansion at \(u^*\). Similar results can be obtained for vertical \(y\) direction. By adding the correction terms, we can obtain \(O(h^2)\) accuracy for the gradients of \(p, u, v\). This numerical treatment is more accurate than the discussion in Section 3.1 where we sometimes use one-side FD scheme for the
gradient of $p$.

Back to the projection method (4.3) to (4.6), we can add the derived correction terms for every central FD operators in the two Helmholtz equations:

\[
\frac{\mu}{2} \Delta_h u^* - \frac{1}{\Delta t} u^* = -\frac{u^n}{\Delta t} - \frac{\mu}{2} \Delta_h u^n + \frac{3}{2} (u^n \cdot \nabla_h) u^n - \frac{1}{2} (u^{n-1} \cdot \nabla_h) u^{n-1} + \nabla_h p^n \frac{t}{2} - G^{n+\frac{1}{2}} \tag{4.27}
\]

and the one Poisson equation:

\[
\Delta_h \phi^{n+1} = \frac{\nabla_h \cdot u^*}{\Delta t} \tag{4.28}
\]

Above method is referred to be projection-IIM method in [41], where it is shown to be second order accurate in $p$, $u$ and $v$. Our goal is to use this method on an adaptive mesh but also obtain the second order accuracy.

### 4.2 Implement the projection-IIM method on the AMR mesh

Above projection-IIM method can be applied to an AMR mesh easily. First, we label all irregular grid points where the interface cuts their central FD stencils in $x$ or $y$ direction. We add correction terms computed by (4.24) to (4.26) for the discrete gradients or Laplacians $\nabla_h u$, $\nabla_h p$ and $\Delta_h u$ in the Helmholtz equations (4.27) and the Poisson equation (4.28). We require that the finest mesh level should cover all the irregular grid points and their FD stencils.

Next, we build the right hand sides of (4.27) and (4.28) by regular central FD scheme
with proper mesh resolutions for coarse and fine mesh levels. For hanging nodes on border of two levels, we use the method described in Section 2.3 to interpolate ghost points with $O(h^3)$ or $O(h^4)$ accuracy. In Figure 4.2, for example, we need the ghost point $g_2$ to obtain the FD scheme of the hanging node 12. By interpolation, we have:

$$u_{g2} = \frac{1}{4}u_{23} - u_{18} + \frac{3}{2}u_{12} + \frac{1}{4}u_{g1} + O(h^4)$$  \hspace{1cm} (4.29)

and $g_1$ can also be interpolated by:

$$u_{g1} = -\frac{1}{16}u_1 + \frac{9}{16}u_3 + \frac{9}{16}u_4 - \frac{1}{16}u_5 + O(h^4)$$  \hspace{1cm} (4.30)

So that the ghost point 12 can be interpolated with $O(h^4)$ accuracy:

$$u_{g2} = \frac{1}{4}u_{23} - u_{18} + \frac{3}{2}u_{12} - \frac{1}{64}u_1 + \frac{9}{64}u_3 + \frac{9}{64}u_4 - \frac{1}{64}u_5 + O(h^4)$$  \hspace{1cm} (4.31)

Then we plug above interpolated ghost point $g_2$ into the central FD scheme at hanging node 12. We can have at least $O(h^3)$ accurate gradients $\nabla_h u$, $\nabla_h p$, and $O(h^2)$ accurate Laplacian $\Delta_h u$ in (4.27).

When we solved the Helmholtz or Poisson equations, we can use the 5-point FD scheme on the finest mesh level, but use the 9-point compact FD scheme on the coarse mesh level in order to minimize the pollution of coarse mesh level on the global accuracy.

We know that the 9-point Laplacian:

$$\nabla_9^2 u_{ij} = \frac{1}{6h^2}[4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1}$$

$$+ u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{ij}]$$  \hspace{1cm} (4.32)
Figure 4.2: A part of an adaptive mesh that includes the fine level mesh around an interface (red curve) and the coarse mesh. For hanging node 12, we need to interpolate ghost points $g_1$ and $g_2$ first, and then plug in the interpolated $g_2$ to compute $\nabla_h u$, $\nabla_h p$ and $\Delta_h u$ for point 12.

can be used to approximate $\nabla^2 u$ by:

$$\nabla^2_9 u = \nabla^2 u + \frac{h^2}{12} \nabla^2(\nabla^2 u) + O(h^4)$$  \hspace{1cm} (4.33)

When the 9-point Laplacian is applied to the Helmholtz equation:

$$\nabla^2 u + \lambda u = f$$  \hspace{1cm} (4.34)

we obtain

$$\nabla^2_9 u = f - \lambda u + \frac{h^2}{12} \nabla^2(f - \lambda u) + O(h^4),$$

$$\nabla^2_9 u + \lambda (u + \frac{h^2}{12} \nabla^2 u) = f + \frac{h^2}{12} \nabla^2 f + O(h^4)$$

$$\nabla^2_9 u + \lambda (u + \frac{h^2}{12} \nabla^2 u) = f + \frac{h^2}{12} \nabla^2 f + O(h^4)$$  \hspace{1cm} (4.35)
as the 9-point FD scheme for the Helmholtz equation.

For moving interface problems, we use exact the same techniques for interface evolution as described in Chapter 3 for Stokes equations. We still use the Hamilton-Jacobi equation to update the level set locally and then take the mass conversation step as in Section 3.2. We set up the three tubes around interface (level set updating tube, adaptive refinement tube and level set re-initialization tube) in the same way as in Section 3.3.

When we use the projection-IIM from time \( t^n \) to \( t^{n+1} \), we need the jump conditions of \( u \) and \( p \) not only at \( t^{n+1} \), but also at \( t^n \) and \( t^{n-1} \) as in (4.27). The solutions \( u^{n+1} \) and \( p^{n+\frac{1}{2}} \) partially depend on the solutions from last two steps \( u^{n-1} \), \( u^n \) and \( p^{n-\frac{1}{2}} \). This is different from the Stokes equations in Chapter 3, where the solutions of \( p \) and \( u \) are computed from only the current level set \( \varphi(x, y, t) \), and the level set \( \varphi(x, y, t) \) is the only connection between two consecutive time steps.

We outline the procedure to use the projection-IIM on AMR for moving interface problems as below Algorithm 3.

### 4.3 Numerical experiments and analysis

#### 4.3.1 A circular flow with a fixed interface

We test the adaptive version of the projection-IIM method through the example from [40, 41] to verify the second order accuracy our AMR-IIM method. This example can be used to set up the simulation of an incompressible flow past a circular cylinder.

This example is to solve the Navier-Stokes equations in the domain \([-1, 1] \times [-1, 1]\) involving a fixed circular interface \( x^2 + y^2 = 1/4 \). The example is created from below
Algorithm 3 Solve Navier-Stokes equations by AMR-IIM for moving interface problems
1: Start with an initial level set $\varphi(x, y, 0)$ and zero velocity field $u^0 = 0$. Compute enclosed area $A_0$.
2: Generate the first AMR from the tube $|\varphi(x, y, t)| \leq \lambda_2 h$.
3: Solve Stokes equations on the first AMR for $p^{\frac{1}{2}}$, $u^1$.
4: while $t < t_{\text{stop}}$ do
5: Obtain the jump conditions for $u^n$, $u^{n-1}$, $p^{n-\frac{1}{2}}$ from the most updated interface $\varphi(x, y, t) = 0$, solve two Helmholtz equations for the intermediate velocity $u^*$, and one Poisson equation for $\phi^{n+1}$ on the most updated AMR, and then obtain the updated solutions $u^{n+1}$ and $p^{n+\frac{1}{2}}$.
6: Use the velocity $u^{n+1}$ to update the level set $\varphi(x, y, t)$ by solving the Hamilton-Jacobi equation $\varphi_t + u \cdot \nabla \varphi = 0$ within a narrow tube $|\varphi(x, y, t)| \leq \lambda_1 h$, where $\lambda_1 < \lambda_2$ to guarantee the interface moves within the finest level mesh.
7: Make correction on $\varphi(x, y, t)$ within the tube $|\varphi(x, y, t)| \leq \lambda_1 h$ against the initial enclosed area $A_0$ for mass conservation.
8: Re-initialize the level set $\varphi(x, y, t)$ within a wide tube $|\varphi(x, y, t)| \leq \lambda_3 h$, where $\lambda_3 > \lambda_2$ to cover all refinement region.
9: if the level set updating tube $|\varphi(x, y, t)| \leq \lambda_1 h$ tends to move beyond the finest level mesh then
10: Generate a new AMR based on most updated level set $|\varphi(x, y, t)| \leq \lambda_2 h$
11: end if
12: $t = t + \Delta t$
13: Update $u$ and $p$: $u^{n-1} = u^n$, $u^n = u^{n+1}$, $p^{n-\frac{1}{2}} = p^{n+\frac{1}{2}}$
14: end while
exact solutions:

\[
    u(x, y, t) = \begin{cases} 
    h(t)(\frac{y}{r} - 2y) & \text{if } r \geq 1/2 \\
    0 & \text{otherwise }
\end{cases} \tag{4.36}
\]

\[
    v(x, y, t) = \begin{cases} 
    h(t)(-\frac{x}{r} + 2x) & \text{if } r \geq 1/2 \\
    0 & \text{otherwise }
\end{cases} \tag{4.37}
\]

\[
    p(x, y, t) = \begin{cases} 
    p^+(x, y, t) & \text{if } r \geq 1/2 \\
    0 & \text{otherwise }
\end{cases} \tag{4.38}
\]

where \( r = \sqrt{x^2 + y^2} \). \( h(t) \) and \( p^+(x, y, t) \) are given in following three cases respectively. As in Figure 4.3, the velocity field shows a circular flow outside the interface.

Figure 4.3: The velocity field of circular flow when \( h(t) = 1 \) with circular interface
The external force $\mathbf{g} = (g_1, g_2)$ and their jumps $[g_1], [g_2]$ can be derived from the exact solutions of $u, v, p$. The singular force $\hat{f}_1, \hat{f}_2$ can be derived from jump relations (4.9).

For example, the normal singular force:

$$\hat{f}_1 = [p] = p^+(x, y, t),$$

(4.39)

and the tangential singular force

$$\hat{f}_2 = [\mu u_n]/\sin \theta = -2\mu h(t).$$

(4.40)

The Dirichlet boundary conditions for $u, v$ are also given from the exact solution.

We test the adaptive version of projection-IIM for three cases with different selections of $h(t)$ and $p^+$. We run till $t = 4s$ with the time step $\Delta t = \frac{1}{2} \min\{h, h/\|u\|_\infty\}$. Below we show the grid refinement analysis for the errors of velocity and pressure against their exact solutions. We use errors in infinity norm $\|E_u\|_\infty = \|E_u\|_\infty + \|E_v\|_\infty$ and $\|E_p\|_\infty$ for error analysis. The order of accuracy is defined by:

$$\text{order} = \frac{\log(E(m)/E(2m))}{\log 2}$$

• **Case 1** Take $h(t) = 1$, $p^+ = 0$ and $\mu = 0.02$ as in [41]. Table 4.1 shows the numerical results by AMR $m(+1)[2]$ at $t = 4s$.

<table>
<thead>
<tr>
<th>AMR $m(+1)[2]$</th>
<th>$|E_u|_\infty$</th>
<th>order</th>
<th>$|E_p|_\infty$</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1.67e-3</td>
<td></td>
<td>6.00e-4</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>4.53e-4</td>
<td>1.9</td>
<td>1.71e-4</td>
<td>1.8</td>
</tr>
<tr>
<td>128</td>
<td>1.16e-4</td>
<td>2.0</td>
<td>4.96e-5</td>
<td>1.8</td>
</tr>
</tbody>
</table>
• **Case 2** Take $h(t) = e^{-t}$, $p^+ = \sin(\pi x)\sin(\pi y)$ and $\mu = 0.02$. So the exact solutions are

\[
u(x, y, t) = \begin{cases} 
e^{-t} \left(\frac{y}{r} - 2y\right) & \text{if } r \geq 1/2 \\ 0 & \text{otherwise} \end{cases}
\]

(4.41)

\[
u(x, y, t) = \begin{cases} 
e^{-t} \left(-\frac{x}{r} + 2x\right) & \text{if } r \geq 1/2 \\ 0 & \text{otherwise} \end{cases}
\]

(4.42)

\[
u(x, y, t) = \begin{cases} \sin(\pi x)\sin(\pi y) & \text{if } r \geq 1/2 \\ 0 & \text{otherwise} \end{cases}
\]

(4.43)

Table 4.2 shows the numerical results by AMR $m(+1)[2]$ at $t = 4s$.

<table>
<thead>
<tr>
<th>AMR $m(+1)[2]$</th>
<th>$|E_u|_\infty$</th>
<th>order</th>
<th>$|E_p|_\infty$</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1.41e-2</td>
<td></td>
<td>4.00e-3</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>3.25e-3</td>
<td>2.1</td>
<td>9.83e-4</td>
<td>2.0</td>
</tr>
<tr>
<td>128</td>
<td>8.06e-4</td>
<td>2.0</td>
<td>2.40e-4</td>
<td>2.0</td>
</tr>
</tbody>
</table>

• **Case 3** Take $h(t) = \sin(t)$, $p^+ = \cos(\pi x)\cos(\pi y)$ as in [23]. So the exact solutions are

\[
u(x, y, t) = \begin{cases} \sin(t) \left(\frac{y}{r} - 2y\right) & \text{if } r \geq 1/2 \\ 0 & \text{otherwise} \end{cases}
\]

(4.44)
\[ v(x, y, t) = \begin{cases} 
\sin(t)\left(-\frac{x}{r} + 2x\right) & \text{if } r \geq 1/2 \\
0 & \text{otherwise}
\end{cases} \quad (4.45) \]

\[ p(x, y, t) = \begin{cases} 
\cos(\pi x)\cos(\pi y) & \text{if } r \geq 1/2 \\
0 & \text{otherwise}
\end{cases} \quad (4.46) \]

We further discuss this problem under two sub-cases \( \mu = 0.5 \) (small Reynolds number) and \( \mu = 0.005 \) (large Reynolds number). See Table 4.3 and 4.4 for results.

### Table 4.3: Case 3a, \( h(t) = \sin(t), \mu = 0.5, t = 4s \)

<table>
<thead>
<tr>
<th>AMR ( m(+1)[2] )</th>
<th>( | E_u |_\infty )</th>
<th>order</th>
<th>( | E_p |_\infty )</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1.12e-3</td>
<td></td>
<td>1.01e-2</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>3.07e-4</td>
<td>1.9</td>
<td>2.34e-3</td>
<td>2.1</td>
</tr>
<tr>
<td>128</td>
<td>8.40e-5</td>
<td>1.9</td>
<td>6.31e-4</td>
<td>1.9</td>
</tr>
</tbody>
</table>

### Table 4.4: Case 3b, \( h(t) = \sin(t), \mu = 0.005, t = 4s \)

<table>
<thead>
<tr>
<th>AMR ( m(+1)[2] )</th>
<th>( | E_u |_\infty )</th>
<th>order</th>
<th>( | E_p |_\infty )</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1.49e-2</td>
<td></td>
<td>9.31e-3</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>2.52e-3</td>
<td>2.5</td>
<td>2.13e-3</td>
<td>2.1</td>
</tr>
<tr>
<td>128</td>
<td>6.34e-4</td>
<td>2.0</td>
<td>5.42e-4</td>
<td>2.0</td>
</tr>
</tbody>
</table>

From all above cases, the second order accuracy is achieved for both the velocity and the pressure by the adaptive version of projection-IIM.

95
4.3.2 The surface tension problem revisit by the Navier-Stokes equations

We do the surface tension problem in Section 3.4.2 again to see what would be different if we solve it by Navier-Stokes equations. We first set up the problem from the jump conditions as in Section 4.1.2. There is no external force in this example:

\[ g_1 = g_2 = 0, \quad [g_1] = [g_2] = 0 \]  \hspace{1cm} (4.47)

The only singular force is the one in normal direction, which is proportional to the interface curvature \( \kappa \) and the surface tension coefficient \( \gamma \):

\[ \hat{f}_1 = \gamma \kappa, \quad \hat{f}_2 = 0 \]  \hspace{1cm} (4.48)

Among the jump conditions in Section 4.1.2, we need take some effort for the \([p_\eta]\) in (4.19)

\[ [p_\eta] = [p]_\eta = \frac{\partial \hat{f}_1}{\partial \eta} = \gamma \frac{\partial \kappa}{\partial \eta} \]  \hspace{1cm} (4.49)

where the tangent derivative \( \frac{\partial \kappa}{\partial \eta} \) can be approximated by the weighed least square method described in [37,42]. In addition, the zero Dirichlet boundary conditions \( u|_{\partial \Omega} = v|_{\partial \Omega} = 0 \) are used in the two Helmholtz equations to solve \( u^*, v^* \).

We first solve the example with an initial ellipse interface:

\[ \frac{x^2}{r_a^2} + \frac{y^2}{r_b^2} = 1 \]  \hspace{1cm} (4.50)
at the center of domain $[-1, 1] \times [-1, 1]$. We take the initial semi-major axes $r_a = 0.8$, and semi-minor axes $r_b = 0.2$. Due to the surface tension, the ellipse will relax to a circle to minimize its circumference. That means both $r_a$ and $r_b$ will converge to $\sqrt{0.8 \times 0.2} = 0.4$ in long run for area conservation.

We run this simulation on AMR 80(+1)[2] in various combinations of viscosity $\mu$ and surface tension coefficient $\gamma$. In Figure 4.4, we show the lengths of $r_a$ and $r_b$ changing over time. We can see the effect of viscosity by comparing the figures vertically. When $\mu$ is as large as 1, the viscosity term $\mu \Delta u$ dominates the Navier-Stokes equations and the interface converges to its equilibrium position directly. In this case, the motion of the interface is very close to that solved by Stokes equations in Chapter 3. When $\mu$ becomes smaller, the inertia term $\left(\frac{\partial u}{\partial t} + (u \cdot \nabla) u\right)$ becomes important and the oscillation of the interface around its equilibrium position can be observed very clearly. Note that the Stokes equations in Chapter 3 do not work successfully in this case with small viscosity.

We can also see the effect of surface tension by comparing the figures horizontally. The larger the surface tension is, the faster the interface relaxes to the equilibrium state. This is consistent with the expectations from physics knowledge. Moreover, it can be observed that in every case the interface converges to the circle $r_a = r_b = 0.4$, which means our approach can preserve the enclosed area very well.

We are more interested in the oscillation case when the viscosity $\mu = 0.02$ and the surface tension coefficient $\gamma = 1$. Eight snapshots are shown in Figure 4.5 for the motion of the interface. It can be observed that the shape of the interface is not always an ellipse, but sometimes like a dumbbell.

We redo the star interface example in Section 3.4.2, but under the case of small
Figure 4.4: The evolution of the semi-major axes $r_a$ (red) and the semi-minor axes $r_b$ (blue) over time in different viscosity $\mu$ and surface tension coefficient $\gamma$. 
Figure 4.5: Snapshots of the oscillating interface (red solid) around its equilibrium status (blue dashed) when the viscosity $\mu = 0.02$, and the surface tension coefficient $\gamma = 1$. 
viscosity. The initial interface is given by:

\[ r(\theta) = 0.6 + 0.25 \sin(5\theta), \quad 0 \leq \theta \leq 2\pi \]  \hspace{1cm} (4.51)

We take the viscosity \( \mu = 0.02 \), and the surface tension \( \gamma = 1 \). Figure 4.6 shows some snapshots in the simulation. The oscillation of the interface around its circular equilibrium position is again observed very clearly.
Figure 4.6: Snapshots of the oscillating interface (red solid) around its equilibrium status (blue dashed) when the viscosity $\mu = 0.02$, and the surface tension coefficient $\gamma = 1$. 

101
Chapter 5

Conclusion and future work

This dissertation presents the adaptive mesh refinement (AMR) technique for the immersed interface method (IIM). The AMR-IIM has been used to solve 2D elliptic interface problems, Stokes and Navier-Stokes equations with fixed or moving interfaces, and some application problems about two-phase incompressible flows. The AMR-IIM inherits the second order accuracy of the previous IIM on a uniform mesh. The AMR-IIM can be used to obtain a high resolution locally around the interface to improve the accuracy by adding grid points as few as possible. The AMR-IIM can also be used to reduce the computational cost spent on smooth regions, so that the limited computer memories can be utilized more efficiently to solve larger problems. The AMR-IIM is built on the level set representation of the interface, so that it can be conveniently implemented for many interface evolution problems.

We sum up the dissertation from below three aspects.

Numerical accuracy. Throughout the study, we keep all modified finite difference schemes of the original IIM for irregular grid points close to interface, such as the correction terms and modified coefficients in elliptic equations, and the correction terms added
in projection method in Navier-Stokes equations. That is why the AMR-IIM inherits the second order accuracy. In that sense, the AMR-IIM is a combination of the IIM FD schemes close to the interface, and standard FD schemes in different resolutions far away from the interface. On the finest mesh level, we compute exactly the same as the original IIM. On coarse mesh levels, we prefer a high order FD scheme to reduce the solution error as much as possible, although the second order convergency is always preserved. Unlike the immersed boundary method (IB), the IIM controls the first order truncation error around the interface, so that the global solution error could be partially polluted by some smooth regions apart from the interface.

Compatibility and stability. It is crucial to control the width of the AMR region in order to be compatible with both the IIM and the level set method in moving interface problems. The refinement region should be thick enough to cover the IIM FD stencils of irregular grid points, and to guarantee the level set is updated within the finest mesh resolution in each time step. Meanwhile, the refinement region should be included by the level set re-initialization tube, so that the generated AMR can correctly track the shape of the most updated interface. Our numerical experiments also show that the AMR-IIM is not as robust as the previous IIM in a uniform mesh, especially for moving interface problems. We sometimes need to tune the computation parameters, such as the number of iterations for the level set re-initialization, the time step-size, and the convergence criteria in the algebraic multigrid solver. These parameters are usually depends on different problems case by case.

Implementation. The FORTRAN codes built for the AMR-IIM have experienced the development process from short to long, and then back to short. We started from the simplest elliptic case (Example 1 in Chapter 2) to test the newly born AMR-IIM, and then added more features of the IIM such as the max-preserving scheme for discontinuous
coefficients, and then imbedded the AMR-IIM into the level set framework to deal with
general interfaces, and then coupled the AMR-IIM with interface evolution module for
Stokes equations, and then coupled the AMR-IIM with the projection method module
for Navier-Stokes equations. Meanwhile, the one level AMR was extended to a two-
level version and even a four-level version. During this development process, we keep re-
organizing the codes to make them easy to use. For example, we formulate the frequently
used FD schemes as subroutines with flexible mesh resolution, such as the 5-point scheme,
the 9-point scheme, the interpolation scheme and the discrete gradient on hanging nodes.
We also formulate subroutines to automatically call any grid point and its 4 or 8 neighbors
from the correct mesh levels they belong to, rather than discussing the messy cases about
which mesh levels are involved. So far, the codes for the AMR module are simplified to
be only hundreds of lines for all cases covered in this dissertation.

In the end, we provide some potential extensions of this work.

Data structure. Although the current AMR data structure described in Chapter 2
works very well, we believe there is still some room to improve for the storage efficiency.
Some current quad-tree and oct-tree based data structure and adaptive level set tech-
niques could be applied to this work.

Discontinuous viscosity. When Stokes or Navier-Stokes equations involve discontin-
uous viscosity $\mu$ across the interface, the jump conditions for pressure and velocity are
coupled together. Currently, the augmented IIM is the only approach for that case. We
hope to develop an adaptive version of the augmented IIM, so that the AMR-IIM can be
applied to more general incompressible flow problems.

3D and more applications. The AMR-IIM should be extended to the three dimensional
case naturally, since the IIM in 3D is ready and the level set method works well in 3D.
We believe that the AMR can be more advantageous in 3D problems. The AMR-IIM
should be able to tackle some large 3D problems which are currently impossible to solve on a uniform mesh due to the restriction of computer memories.
REFERENCES


