THREE-DIMENSIONAL MARKER-BASED MULTIPHASE FLOW COMPUTATION USING ADAPTIVE CARTESIAN GRID TECHNIQUES

By

RAJKESHAR SINGH

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2006
To my family and teachers
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Wei Shyy for providing me the opportunity and flexibility to perform this work. I can not thank him enough for being so patient and understanding for years and pushing us to learn more.

I would like to thank Dr. Renwei Mei, Dr. James F. Klausner and Dr. Samim Anghaie for agreeing to serve in my thesis committee. Since the surroundings dictate the quality of life in several ways, I thank my research group members and Dr. Siddharth Thakur for helping with academic aspects while providing a memorable company for the past four years.

I thank my wife who has been the best friend for more than half a decade and has been extremely patient and understanding during all these years. Finally I thank my parents who have been behind me every step of the way providing their unconditional support.
TABLE OF CONTENTS

ACKNOWLEDGMENTS .................................................................................................................. vi
LIST OF TABLES ........................................................................................................................ viii
LIST OF FIGURES ........................................................................................................................ x
LIST OF SYMBOLS ....................................................................................................................... xv
ABSTRACT ................................................................................................................................ xvi

1 INTRODUCTION ....................................................................................................................... 1
  1.1 Objective and Contributions .............................................................................................. 3
  1.2 Outline of the Dissertation ................................................................................................. 7

2 LITERATURE REVIEW ............................................................................................................ 9
  2.1 Interface Tracking .................................................................................................................. 9
   2.1.1 Lagrangian .................................................................................................................. 9
   2.1.2 Eulerian ..................................................................................................................... 10
   2.1.3 Mixed Eulerian-Lagrangian ....................................................................................... 11
   2.1.4 Challenges and Recent Advances ............................................................................. 11
   2.1.5 Present Approach: Markers with or without Connectivity ........................................ 16
  2.2 Interfacial Dynamics Modeling .......................................................................................... 17
   2.2.1 Sharp Interface .......................................................................................................... 17
   2.2.2 Continuous Interface ................................................................................................. 19
   2.2.3 Challenges and Recent Advances ............................................................................. 19
   2.2.4 Present Approach: Sharp or Continuous Interface ................................................... 21
  2.3 Adaptive Grid Computation .............................................................................................. 22
   2.3.1 Cartesian Grid Data Structures .................................................................................. 24
   2.3.2 Present Approach: Unstructured Adaptive Cartesian Grid ...................................... 26

3 MULTIDIMENSIONAL IMMERSED BOUNDARY COMPUTATION ..................................... 27
  3.1 Immersed Boundary Method ............................................................................................ 27
   3.1.1 Interfacial Conditions ............................................................................................... 29
   3.1.2 Material Property Smoothing .................................................................................... 31
   3.1.3 Momentum Source Term Computation ...................................................................... 33
6.1 Summary ........................................................................................................... 113
6.2 Future Work ..................................................................................................... 115

LIST OF REFERENCES ............................................................................................. 118

BIOGRAPHICAL SKETCH ....................................................................................... 126
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Some industrial and natural occurrences of two-phase flows.</td>
<td>2</td>
</tr>
<tr>
<td>1-2</td>
<td>Key marker based 3D tracking computations in literature.</td>
<td>5</td>
</tr>
<tr>
<td>2-1</td>
<td>Summary of some representative methods and related issues.</td>
<td>14</td>
</tr>
<tr>
<td>2-2</td>
<td>Summary of recent advances in interface tracking.</td>
<td>15</td>
</tr>
<tr>
<td>5-1</td>
<td>Error estimates using Equation (5.2) for interface in a time reversed vortex field test</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>comparing conservative and non-conservative restructuring.</td>
<td></td>
</tr>
<tr>
<td>5-2</td>
<td>Volume error for reconstruction of a spherical interface and approximate radial</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>perturbation required for correction.</td>
<td></td>
</tr>
<tr>
<td>5-3</td>
<td>Max velocity error and computation time normalized with time taken for 16x16</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>uniform grid. The adapted grids are shown in Figure 5-4.</td>
<td></td>
</tr>
<tr>
<td>5-4</td>
<td>Lid driven cavity computation using uniform and adaptive grids.</td>
<td>73</td>
</tr>
<tr>
<td>5-5</td>
<td>Natural convection computation.</td>
<td>76</td>
</tr>
<tr>
<td>5-6</td>
<td>Pressure drop for an inviscid fluids with density ratio = 1.0.</td>
<td>79</td>
</tr>
<tr>
<td>5-7</td>
<td>Effect of Laplace number on spurious velocity.</td>
<td>79</td>
</tr>
<tr>
<td>5-8</td>
<td>Effect of density ratio on pressure drop and spurious current on 50x50x50 grid</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>(viscosity ratio = 1.0, $La = 250$)</td>
<td></td>
</tr>
<tr>
<td>5-9</td>
<td>Effect of viscosity ratio on pressure drop and spurious current on 50x50x50 grid</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>(density ratio = 1000, $La = 250$)</td>
<td></td>
</tr>
<tr>
<td>5-10</td>
<td>Grid convergence test for $E_o = 1.0$, $M = 1.0 \times 10^{-3}$. The density and viscosity</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>ratios are set to 20.0.</td>
<td></td>
</tr>
<tr>
<td>5-11</td>
<td>Computation of rising bubbles with different terminal shape regimes.</td>
<td>87</td>
</tr>
<tr>
<td>5-12</td>
<td>Computational parameter for rising bubble-coalescence tests.</td>
<td>88</td>
</tr>
<tr>
<td>5-13</td>
<td>Properties of tetradecane and nitrogen.</td>
<td>93</td>
</tr>
</tbody>
</table>
5-14 Binary drop collision computation with density ratio = 666.081 and viscosity ratio 
= 179.28....................................................................................................................94

5-15 Computational parameters for one-dimensional phase change computation........98
LIST OF FIGURES

Figure | page |
--- | --- |
1-1 | Illustration of a typical multiphase flow. | 1 |
1-2 | Mixed Eulerian-Lagrangian tracking. | 5 |
2-1 | Illustration of Lagrangian interface tracking. | 10 |
2-2 | Schematic of Eulerian tracking using level set and volume of fluid. | 11 |
2-3 | Interface in a time reversed vortex field using level-set method. | 12 |
2-4 | A conceptual interface reconstruction using piecewise linear segments in the volume-of-fluid method. | 13 |
2-5 | A surgical alteration of the interface data to accomplish topology change. | 15 |
2-6 | A three-dimensional marker based film boiling computation. | 15 |
2-7 | Illustration of conceptual differences between marker-based tracking with and without connectivity information. | 17 |
2-8 | Surface tension, pressure and normal component of shear stress acting on an interface. | 18 |
2-9 | Sharp and continuous interface methods. | 21 |
2-10 | The effect of fluid property jumps on the computation time for a rising bubble. | 23 |
2-11 | A cell-by-cell and block-by-block grid adaptation. | 25 |
2-12 | A tree-based adaptive Cartesian grid-data. | 25 |
3-1 | Schematic of the immersed boundary method. | 28 |
3-2 | Probe based technique for temperature gradient computation on the interface. | 30 |
3-3 | A triangulated interface and the data-structure. | 31 |
3-4 | Indicator function and the material flags for cells completely outside or inside the interface. | 33 |
Momentum source term due to surface tension force on a circular interface. ..........35
Computation of the unit normal and tangent vectors on interface triangles. ..........35
Curvature computation for a unit circle using Equation (3.18) and cubic spline with
(a) twelve and (b) sixty points.................................................................................36
Interface restructuring approach.............................................................................38
Conservative marker deletion for two-dimensional interfaces...............................39
Conservative marker deletion for three-dimensional interfaces..............................39
Level-contour based interface reconstruction.........................................................40
Interface reconstruction criterion uses two probes from the interface along the
normal vector and bi-linearly interpolates the indicator values Ind1 and Ind2.............41
Two dimensional reconstructed interface edges (e1, e2 and e3) and corresponding
connectivity data. ........................................................................................................42
Illustration of 2D cell-by-cell interface reconstruction.............................................43
A 2D cell with four markers creating two locally disconnected edges....................44
A reconstructed interface segment in a cell...............................................................45
Some common reconstructed interface segments and integer vertex-flags..............45
Difficulties due to degeneracy of reconstructed interface data and implemented
remedy..........................................................................................................................46
Orientation setting of reconstructed interface.........................................................47
Examples of 3D interface reconstruction..................................................................47
Cartesian grid data-structure....................................................................................51
Adaptive grid refinement..........................................................................................52
Cartesian cells sharing a face and the corner cells are not allowed to differ by more
than one level of refinement......................................................................................53
Geometry based adaptation.......................................................................................54
Staggered grid arrangement.......................................................................................58
Spatial discretization of convection term..................................................................59
4-7 A one-dimensional interface between $i$ and $i+1$. .....................................................61
4-8 Linear interface temperature correction within one cell on each side of the interface. ...........................................................................................................................................62
4-9 Presently employed interface temperature correction technique solves a correction equation sharply on each side of the interface. ...........................................................63
5-1 Interface in a time reversed vortex field........................................................................67
5-2 Error estimate and grid convergence for interface in time reversed vortex field....67
5-3 Grid convergence test for volume error due to reconstruction......................................69
5-4 Decaying vortex field .................................................................................................71
5-5 Grid convergence test for decaying vortex field showing maximum velocity error computed on uniform and adaptive grids.................................................................71
5-6 Streamlines for lid-driven cavity simulation for Re = 100........................................72
5-7 Lid driven cavity. .......................................................................................................73
5-8 The u velocity profile along a vertical line through the center of lid-driven cavity using (a) uniform grids and (b) adaptive grids.............................................................74
5-9 Indicator contour used for modeling the cylindrical computational domain. ........75
5-10 Computed temperature and error for heat transfer in a concentric cylinder. ....75
5-11 Natural convection with Pr = 0.71 and Ra = 1.0x10^5. ........................................76
5-12 Static bubble computation.........................................................................................78
5-13 Grid convergence tests for static bubble. ...............................................................79
5-14 Effect of reconstruction frequency on spurious velocities.....................................81
5-15 Computational set-up for a single rising bubble. ..................................................82
5-16 Aspect ratio computation .......................................................................................83
5-17 Bubble rise with Eo = 1.0, $M = 1.0 \times 10^{-3}$, density and viscosity ratio of 20........84
5-18 Computational domain for cases in Table 5-11. ....................................................85
5-19 Computed shapes from Table 5-11. ........................................................................86
5-20 Computed case for $M = 0.971$, Eo = 97.1. .........................................................86
5-21 Computational set-up for head-on coalescence test..................................................88
5-22 Head-on coalescence ..................................................................................................89
5-23 Grid and streamlines at t = 0.125 sec for head-on coalescence................................89
5-24 Instance of interface reconstruction during head-on coalescence of two rising bubbles at t = 0.116 s.................................................................90
5-25 Experimental photographs of bubble coalescence taken at 0.03s time intervals. ....90
5-26 Off-axis coalescence. ...............................................................................................91
5-27 Binary drop collision regimes marking the location of computed parameters with solid circles.................................................................93
5-28 Impact parameter $B$ is defined as $B=h/d.$ .................................................................93
5-29 Binary drop collision Case 1 from Table 5-14............................................................100
5-30 A cross section of the computational grid and the time history of grid data-size for Case 1 from Table 5-14.................................................................101
5-31 Binary drop collision Case 2 from Table 5-14............................................................102
5-32 A cross section of the computational domain showing the interface and the velocity vectors for Case 2 from Table 5-14.................................................................102
5-33 A cross section of the computational grid and the time history of grid data-size for Case 2 from Table 5-14.................................................................103
5-34 Binary drop collision Case 3 from Table 5-14............................................................104
5-35 A cross section of the computational grid and the time history of grid data-size for Case 3 from Table 5-14.................................................................105
5-36 A cross section of the computational domain showing the interface and the velocity vectors on the left and pressure profile on the right for Case 3 from Table 5-14.................................................................106
5-37 Binary drop collision Case 4 from Table 5-14............................................................107
5-38 A cross section of the computational grid and the time history of grid data-size for Case 4 from Table 5-14.................................................................108
5-39 A cross section of the computational domain showing the interface and the velocity vectors for Case 4 from Table 5-14.................................................................109
5-40 One dimensional phase change setup.....................................................................110
5-41 Comparison of computed and theoretical solution for Case 1 in Table 5-15

5-42 One-dimensional phase change computation for Case 2 in Table 5-15 showing
    interface location on the left and temperature distribution at t = 0.1 on the right.

5-43 Stationary bubble growth rate.
LIST OF SYMBOLS

\( \rho, \mu, \sigma \)  \quad \text{Density, viscosity and surface tension}

\( C, K, \lambda \)  \quad \text{Heat capacity, thermal conductivity and latent heat of evaporation}

\( m \)  \quad \text{Interfacial mass transfer rate (-ve for evaporation)}

\( k, f, n \)  \quad \text{Interface local curvature, surface tension force, unit normal vector}

\( F_s \)  \quad \text{Momentum source term due to surface tension force}

\( I \)  \quad \text{Indicator function}

\( \delta, D \)  \quad \text{Dirac-delta and discretized Dirac-delta function}

\( d \)  \quad \text{Bubble diameter}

\( U = (u, v, w) \)  \quad \text{Velocity}

\( P \)  \quad \text{Pressure}

\( T \)  \quad \text{Temperature}

\( g \)  \quad \text{Gravity}

\( (\phi)_{\text{int}} \)  \quad \text{A property } \phi \text{ on interface markers}

\( (\phi)_1 \)  \quad \text{A property } \phi \text{ of fluid outside the interface}

\( (\phi)_2 \)  \quad \text{A property } \phi \text{ of fluid inside the interface}

\( \frac{(\phi)_1}{(\phi)_2} \)  \quad \text{Ratio of a fluid property } \phi
\( T_{sat} \)  
Saturation temperature

\( X = (x, y, z) \)  
Spatial coordinates along x, y and z coordinate axes

\( L \)  
Length scale

\( \text{Re} = \frac{\rho U L}{\mu} \)  
Reynolds number

\( \text{We} = \frac{\rho U^2 L}{\sigma} \)  
Weber number

\( \text{Fr} = \frac{U^2}{Lg} \)  
Froude number

\( \text{Ja} = \frac{\rho C \Delta T}{\rho \lambda} \)  
Jacob number

\( \text{Pe} = \frac{\rho C U L}{K} \)  
Peclet number

\( \text{Pr} = \frac{\mu C}{K} \)  
Prandtl number

\( \text{La} = \frac{\rho \sigma L}{\mu^2} \)  
Laplace number

\( \text{Ra} = \frac{\rho^2 C g \beta A T L^3}{\mu K} \)  
Rayleigh number

\( \text{Ca} = \frac{\mu |U|_{\text{max}}}{\sigma} \)  
Capillary number

\( \text{M} = \frac{g \mu^4}{\rho \sigma^3} \)  
Morton number

\( \text{Eo} = \frac{\rho g L^2}{\sigma} \)  
Eötvos number
Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

THREE-DIMENSIONAL MARKER-BASED MULTIPHASE FLOW COMPUTATION USING ADAPTIVE CARTESIAN GRID TECHNIQUES

By
Rajkeshar Singh

August, 2006

Chair: Wei Shyy
Major Department: Mechanical and Aerospace Engineering

Multiphase flows associated with interfacial dynamics, steep jump in fluid properties and moving boundaries between different phases pose substantial computational challenges in terms of both modeling as well as computational cost. The present work uses an immersed boundary technique to model the interfacial dynamics, Lagrangian markers to track the moving phase boundaries, and a stationary Cartesian grid to solve all the flow governing equations. Time dependent triangulated surface meshes are employed to represent the time dependent interfaces shape and location. Based on the solution characteristics, multi-level, three-dimensional adaptive grid techniques are incorporated into the computational framework to help meet the resolution requirements. Furthermore, a conservative marker redistribution technique is developed to maintain a desired marker-spacing, and a connectivity preserving level contour-based reconstruction technique is devised to handle topological changes associated with the interfacial dynamics. The flow equations were solved using the projection method with a finite...
volume staggered grid formulation on adaptive grids. For phase change problems, accuracy of the mass transfer computation critically affects the overall computational outcome. Efforts have been made to address this via a sharp interface-based mass transfer mode combined with the immersed boundary method. The capabilities and accuracy of the individual components and overall computational system are tested with a range of computations including demonstration of improvements with conservative interface restructuring, reconstruction and its effect on immersed boundary solution accuracy, estimation of computational cost saving with adaptive grids, rising bubble coalescence, binary drop collision, and stationary bubble growth in a superheated liquid pool. The method has demonstrated its capability for handling high density ratio, \( O(1000) \), multiphase fluid dynamics.
CHAPTER 1
INTRODUCTION

Multiphase flows are characterized by two or more fluids in relative motion. The constituent fluids usually have different physical properties separated by an interface that evolves with the flow. Figure 1-1 depicts some of the defining elements of a typical multiphase flow— a moving/deforming surface that separates different fluids/phases; discontinuous fluid properties (density, viscosity, conductivity, etc.) across the interface; interfacial effects such as surface tension, phase change, etc.

Figure 1-1. Illustration of a typical multiphase flow.

Presence of such flows in every day life can be recognized by simple examples of ocean waves, boiling water, blood flow in the body. Some specific practical applications may be cited as boiling heat transfer in power industries, biochemical and metallurgical industries, cavitation and ultrasound technology for medical and industrial applications. The constituents in these flows may be liquid-liquid, gas-liquid, liquid or gas with solid. Table 1-1 by Kleinstreuer (2003) gives a summary of some of the industrial or natural occurrences of such flows. Since consideration of all the combinations in Table 1-1 is a
prohibitively difficult task, the scope of the present work is controlled by focusing only on incompressible liquid-liquid or gas-liquid flows for bubble and drop dynamics problems.

**Table 1-1. Some industrial and natural occurrences of two-phase flows.**

<table>
<thead>
<tr>
<th>Phase combination</th>
<th>Phase configuration</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gas-Solid</td>
<td>Dispersion of solid particles in gas</td>
<td>Aerosol, particulate matter pollution; filters and particle-collection devices; Pneumatic transport of particles; gas–fluidized beds</td>
</tr>
<tr>
<td>Liquid-Solid</td>
<td>Dispersion of particles in liquid</td>
<td>Cells and particles in blood; hydraulic transport of particles; liquid-fluidized beds</td>
</tr>
<tr>
<td>Gas-Liquid or Liquid Gas</td>
<td>Dispersion of liquid drops in gas</td>
<td>Mist, fog, cloud; coalescence of drops and formation of rain; droplet removal devices; fuel droplet injection; spray coating and cooling</td>
</tr>
<tr>
<td></td>
<td>Dispersion of gas bubbles in liquid</td>
<td>Transport of oil and gas; foam formation; evaporators, boiling flow in pipe</td>
</tr>
<tr>
<td>Liquid-liquid</td>
<td>Dispersion of liquid drops in liquid</td>
<td>Emulsion and creams; break-up of immiscible drops; coalescence of drops and phase separation</td>
</tr>
<tr>
<td></td>
<td>Both bodies of liquids connected</td>
<td>Transport of two liquids in horizontal pipe; movements of oil-water interface in porous rock.</td>
</tr>
</tbody>
</table>

Note: Adapted from Kleinstreuer (2003)

Presence of multiple time and length scales makes numerical simulation of such flows a difficult and expensive affair. Time dependent interfaces with topology changes (bubble coalescence or breakup) and interfacial mass/heat transfer add further to the computational complexity and cost. Recent years spanning more than a decade have witnessed an ever increasing computational activity with the research spanning various aspects ranging from enhancement of computational efficiency to interface tracking and flow modeling. The computational expense and difficulties in handling interfaces are in part responsible for relatively fewer computations in 3D as compared to a vast number of 2D or axisymmetric cases. These limitations are even more visible with phase change
computations. Besides the high computational cost, phase change computations present several other difficulties with the mathematical modeling and computational algorithms. Some of the modern day phase change computations are - Dhir (2001), Son (2001), Son et al. (2002) and Luo et al. (2005) using level-set; Welch and Wilson (2000) using volume-of-fluid; Esmaeeli and Tryggvason (2004), Tryggvason et al. (2005), Shin et al. (2005) and their earlier work using marker based tracking.

Although continuing developments in computational techniques and resources are making the numerical simulation more and more tractable, several algorithmic and modeling challenges remain unresolved. With further discussion, it will be highlighted in the following chapter that some of the computational aspects often have conflicting requirements and demand several compromises to be made based on the problem of interest.

1.1 Objective and Contributions

The primary goal of the present work is to develop an effective multidimensional interfacial flow computation-system. The process involves assembling key ideas from different approaches and techniques into a unified framework along with suitable improvements and modifications wherever necessary. Development of a parallel computing platform is an essential requirement for large scale computing. However, the current efforts are part of a larger research group activity and thus do not cover parallel computing aspects in this document. The computational components are divided into three generic categories: interface tracking; flow modeling; flow computation. The general emphasis and objectives in these categories are:

- Multi-fluid interface tracking
  - Time dependent triangulated surface grids for interface representation.
  - Maintain resolution by locally adding/deleting nodes on the surface grid.
- Capability to handle complex topology changes.
- Interfacial dynamics and flow modeling
  - Single fluid formulation for the mass and momentum conservation equations.
  - Surface tension forces modeled as smoothed volumetric source term in the momentum equation.
  - Accurate computation of the interfacial mass transfer rate for phase change problems.
- Adaptive grid computation
  - Grid generation and dynamic adaptation based on interface location and flow features.
  - Adaptive grid base incompressible Navier-Stokes computation.

The interface is tracked on a stationary Cartesian grid using time dependent triangulated surfaces (Figure 1-2). This approach is also referred to as a marker based tracking method. Since all the flow computations are performed on the background stationary grid and the interface is tracked using Lagrangian markers, the overall approach is of a mixed Eulerian-Lagrangian type. Although this approach has several advantages due to explicit knowledge of the interface shape/location for computing the geometric properties, the difficulties in data-handling have deterred a wider usage.

Table 1-2 presents some of the key usages and techniques demonstrated to handle complex multidimensional interfacial flows with marker based tracking. The present method is similar to that of Tryggvason et al. (2001) with primary contributions made in interface handling. The triangulated-interface is restructured by locally adding or deleting markers to avoid undesirably long or short edges. The commonly used restructuring techniques in literature (Tryggvason et al., 2001, 2005; Sousa et al. 2004) introduce some amounts of error in the interface volume for which a conservative algorithm is proposed. Topology changes were handled using the level-contour based interface reconstruction of Shin and Juric (2002) with appropriate algorithmic considerations to establish a valid interface connectivity data.
Figure 1-2. Mixed Eulerian-Lagrangian tracking. (a) A two dimensional interface using connected markers on a background grid and (b) a triangulated interface.

Table 1-2. Key marker based 3D tracking computations in literature.

<table>
<thead>
<tr>
<th>3D Explicit Tracking</th>
<th>Author</th>
<th>Summary of technique</th>
</tr>
</thead>
</table>
| A set unconnected triangles | Shin and Juric (2002) | • Easy algorithms for computing geometric properties and dealing with complex topology changes.  
• Reduced flexibility in controlling interface resolution and dealing with multiple interfaces. |
| A set of unconnected points | Torres and Brackbill (2000) | • Easy to handle complex topology changes but interface geometry computation more involved that Shin and Juric (2002)  
• Reduced flexibility in controlling interface resolution and dealing with multiple interfaces. |
| Triangulated surface grid. | Tryggvason et al. and group (2001, 2005) | • Flexible control over interface resolution and dealing with multiple interfaces.  
• Uses reconstruction technique of Shin and Juric (2002) to handle topology changes. |

The interface moves on the background Cartesian grid that is dynamically adapted based on the interface location and flow solution to optimize the computational cost. All the flow governing equations are solved on background grid. The data-structure and Cartesian grid generation is based primarily on the work of Aftosmis (1997). The Navier-Stokes solution is performed using projection method with a staggered grid based finite-volume formulation. The underlying non-uniform nature of the grid makes staggered grid relatively more difficult to implement than the corresponding collocated grid (Singh &
Shyy, 2006), but provides a compact pressure-velocity coupling. As detailed in subsequent chapters, several simplifying approximations in the numerical discretization were made for a convenient flow solver implementation on the spatially adaptive grid. These approximations are present only on the coarse-fine grid-interfaces which are kept away from the physical location of the interface by suitable refinement criteria.

Most of the interfacial dynamics modeling and development uses the already available concepts from literature. The underlying single fluid formulation for mass and momentum equation with marker based tracking is commonly recognized as the immersed boundary method (Peskin, 1977; Tryggvason et al., 2001). This method smears the fluid properties across the interface and models the surface tension forces as smoothed volumetric source terms in the momentum equation. Although the mass and momentum equations use immersed boundary method, it is possible to avoid such approximations for the temperature equation. Since interfacial mass transfer depends on the jump in temperature gradients across the interface, an attempt has been made to treat the temperature equations sharply in order to accurately compute the mass transfer rate. The immersed boundary method of Juric and Tryggvason (1998) uses an iterative procedure to maintain the interface temperature while the present method uses a simple linear correction similar to the level-set based computations of Morgan (2005). At this point, the current approach deviates from a conventional immersed boundary method and borrows the philosophy from recent developments in level-set and sharp interface ghost fluid method of Luo et al. (2005) and Gibou et al. (2003). The presented method solves the temperature equation on both sides of the interface and does not require assuming fixed temperature inside the interface as in the work of Son (2001).
The specific contributions of the present work may be listed as:

1. Marker based interface tracking
   a. Conservative interface restructuring technique to maintain resolution without introducing errors in the interface volume.
   b. Development of interface reconstruction algorithm for topology change. Several challenges/difficulties are highlighted along with simple remedies.
   c. Conservative restructuring along with reconstruction technique offers an enhanced flexibility in handling multiple interfaces as compared to a connectivity-free tracking method of Shin and Juric (2002).

2. Interfacial dynamics modeling
   b. A sharp interface treatment for temperature equation for accurate interfacial mass-transfer computation during phase change. Strategies to incorporate level-set based sharp interface techniques within the present marker based tracking are presented.

3. Adaptive grid computation
   a. Implementation of an adaptive Cartesian grid generator with dynamic adaptation based on the interface location and flow solution.
   b. Development of a staggered grid, finite volume based Navier-Stokes computation algorithm for velocity and pressure computation.

The capabilities of the developed approach and algorithms have been demonstrated via a set of complex multiple rising bubbles and binary drop collision computations as one of the few known cases in literature using marker based tracking. For example, the only known 3D binary drop collision with marker based tracking are by Nobari and Tryggvason (1996); Shin and Juric (2002) performed for moderate density ratios of $O(10)$ while present computations were performed for flows with density ratios of $O(1000)$.

1.2 Outline of the Dissertation

The document is organized into six chapters. Chapter 2 presents brief review of alternative approaches in literature. The basics of immersed boundary modeling and related key components dealing with interface management are presented in Chapter 3.
Chapter 4 presents the adaptive Cartesian grid generation technique and discusses the Navier-Stokes solution procedure highlighting various simplifying assumptions in the numerical discretization.

The key results are presented in Chapter 5. Tests characterizing and demonstrating interface restructuring, reconstruction, single and multiple rising bubbles and binary drop collision are presented along with preliminary assessment of the accuracy of phase change computation.

Chapter 6 provides a summary of the presented work and highlights various issues, approximations following a brief comment on future extensions and refinements.
CHAPTER 2
LITERATURE REVIEW

This chapter provides a brief review of some of the popular techniques based on the categories outlined in the previous chapter: interface tracking; interfacial dynamics modeling; adaptive grid based computation. Various pros and cons of these techniques are visited in an attempt to place the current choices and highlight some of the relevant issues and difficulties to be addressed in following chapters.

2.1 Interface Tracking

As reviewed by Shyy et al. (1996), the three main categories are Lagrangian, Eulerian and mixed Eulerian-Lagrangian. Due to the extensive amount of available literature, the scope of this section is limited to providing a compact background and some of the key issues with these techniques along with the state-of-the-art developments.

2.1.1 Lagrangian

Lagrangian method discretizes the computational domain with the interface as a boundary for creating a body fitted grid (Figure 2-1). This approach requires adjustment or regeneration of the grid in order to accommodate changes in the interface geometry. Some usages of this approach can be found in the works of Ryskin and Leal (1984), Raymond and Rosant (2000), Wasekar and Manglik (2003) using structured curvilinear grids and work of Perot and Nallapati (2003) using triangulated (tetrahedrons in 3D) grids.
Although the body fitted nature of the grid makes it a desirable choice, handling of body fitted grids require sophisticated grid generation capabilities especially when the interfaces undergoes topology change. Observing the frequency of the usage, this is not among the most popular choices and hence it will not be considered further.

Figure 2-1. Illustration of Lagrangian interface tracking.

2.1.2 Eulerian

Eulerian methods use a scalar function to represent the interface implicitly. The level-set (Osher & Fedkiw, 2001, 2003) and volume-of-fluid (Youngs, 1982; Rider & Kothe, 1998; Scardovelli & Zaleski, 2002) are examples of Eulerian tracking. This method ‘captures’ the interface using the information contained in a scalar function. This scalar function is a signed distance function in the level-set method and volume fraction in the volume-of-fluid method (Figure 2-2).

Unlike the Lagrangian methods, Eulerian methods track the interface on a stationary grid using a transport Equation (2.1) for the scalar function evolution that contains the interface information. Due to the Eulerian nature of the interface, this method does not necessitate any changes in the computational grid as required by Lagrangian methods.

\[ \frac{\partial \phi}{\partial t} + V \cdot \nabla \phi = 0 \]  (2.1)
2.1.3 Mixed Eulerian-Lagrangian

This class of method derives its name from the fact that it tracks the interface using Lagrangian markers moving on an Eulerian computational grid (Figure 1-2). These markers move on the computational grid using a simple advection Equation (2.2) where the marker velocities are interpolated from the Eulerian grid. Thus, (similar to Lagrangian) any geometric information about interface is explicitly available and (similar to Eulerian methods) the interface evolution does not require any modification to the computational grid (Tryggvason et al., 2001; Glimm et al., 1998, 2000; Singh & Shyy, 2006).

\[
\frac{\partial X_{\text{int}}}{\partial t} = U_{\text{int}}
\]  

(2.2)

2.1.4 Challenges and Recent Advances

The above introduced methods for interface tracking have some associated advantages and disadvantages that make general preference of any particular method difficult. Table 2-1 gives a summary of the key features of these methods. The Eulerian methods are among the easiest to implement and some of their most cited advantages lie
in their natural ability to handle interfaces undergoing topology changes. On the flip side, level-set method (LS) suffers from erroneous mass loss that may become unacceptably large and detrimental. Such a mass-loss problem is highlighted by the errors shown in the tests conducted by Enright et al. (2002) where a spherical interface undergoes severe deformation and it is expected to return to original shape when placed in a time-reversed vortex field. Figure 2-3(a) shows the severe loss in volume of a spherical interface in such a field using level-set method for interface tracking. This drawback was ameliorated by Enright et al. (2002) using a particle level-set method that uses Lagrangian markers around the interface to avoid erroneous breakup/deletion of the level-set characteristics (Figure 2-3(b)).

![Figure 2-3](image)

Figure 2-3. Interface in a time reversed vortex field using level-set method. (a) Severe mass loss with level set method; (b) improvements produced by particle level-set method (Enright et al., 2002). [Reprinted with permission]

Unlike the level-set method, the volume-of-fluid tracks the volume (mass) explicitly and hence does not suffer from mass-loss problems. However, volume-of-fluid methods have been observed to produce spurious flotsam (‘floating wreckage’) and jetsam (‘jettisoned goods’) in the regions of high interface-curvature under-resolved by the computational grid (Rider & Kothe, 1998). The most serious difficulty with the VOF is due to errors in computation of interface curvature required for surface tension
computation. These difficulties arise due to the sharply changing nature of the volume-fraction and difficulties in reconstructing a continuous interface using the volume-fraction information (Figure 2-4). A more comprehensive detail of various VOF based interface reconstruction and subsequent developments could be found in Rider and Kothe (1998), Scardovelli and Zaleski (2002), Renardy and Renardy (2002), Meier et al. (2002).

![Figure 2-4. A conceptual interface reconstruction using piecewise linear segments in the volume-of-fluid method.](image)

The mixed Eulerian-Lagrangian methods possess several desirable features when compared with Lagrangian or Eulerian methods. Due to explicit definition of interface using markers, the interface does not inaccurately diffuse in time and loose mass as level set methods. It has also been noticed that explicit tracking does not require as high grid resolution as Eulerian methods (Glimm et al., 1998). Explicit definition of the interface also avoids inaccuracies caused by the errors in interface reconstruction for VOF methods.

The most serious concern against the Eulerian-Lagrangian methods has been raised due to the complexity of interface data that becomes difficult to handle especially for topology changes. These difficulties arise due to the algorithmic complexity involved in ‘surgical’ altering of the interface data to accommodate topology changes. The source of difficulty lies in establishing valid connectivity information. Although such a procedure
is easily accomplishable for two-dimensional interfaces (Figure 2-5), extension to three dimensions is a much more involved procedure. The potential complexities in marker-based tracking may be highlighted from the interface shapes seen in some of the 3D film boiling computations of Shin and Juric (2002) (Figure 2-6).

Several attempts have been made by researchers to simplify the marker-based algorithm by resorting to elimination of the need to maintain the connectivity information. One of the first such attempts is due to Torres and Brackbill (2000) who used a set of unconnected markers to describe the interface. Lack of the need to maintain explicit connectivity information, this approach makes it easier to handle topological changes. The subsequent development of connectivity free tracking using unconnected triangular elements by Shin and Juric (2002) simplified the algorithms further. Table 2-2 presents some key developments by combining various individual techniques to exploiting their attractive features.

Table 2-1. Summary of some representative methods and related issues.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Advantages</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lagrangian</td>
<td>• Interface explicitly known and part of the computational grid.</td>
<td>• Frequent grid regeneration.</td>
</tr>
<tr>
<td></td>
<td>• Geometric property computed directly on interface.</td>
<td>• Difficult and expensive for complex 3D computations.</td>
</tr>
<tr>
<td>Eulerian (Level-set)</td>
<td>• Easy geometry-computation.</td>
<td>Severe mass loss/gain in under-resolved areas of the interface.</td>
</tr>
<tr>
<td></td>
<td>• All computations on Eulerian grid.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Easy to handle topology changes.</td>
<td></td>
</tr>
<tr>
<td>Eulerian (Volume of Fluid)</td>
<td>• No mass loss/gain.</td>
<td>• Poor performance in under-resolved areas of the interface.</td>
</tr>
<tr>
<td></td>
<td>• All computations on Eulerian grid.</td>
<td>• Difficult geometry computation.</td>
</tr>
<tr>
<td></td>
<td>• Easy to handle topology changes.</td>
<td></td>
</tr>
<tr>
<td>Mixed Eulerian-</td>
<td>• Interface explicitly known even at sub-grid levels.</td>
<td>• Complex data-structure and book-keeping for 3D interfaces.</td>
</tr>
<tr>
<td>Lagrangian</td>
<td>• Flow computations on Eulerian grid.</td>
<td>• Difficult for topological change.</td>
</tr>
</tbody>
</table>
Table 2-2. Summary of recent advances in interface tracking.

<table>
<thead>
<tr>
<th>Base method</th>
<th>Improved method</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level-set (LS)</td>
<td>Particle level-set (Enright et al., 2002)</td>
<td>Ameliorates the excessive/erroneous mass loss using a cloud of markers around level-set interface.</td>
</tr>
</tbody>
</table>
| Volume of fluid (VOF)| Hybrid marker-VOF (Aulisa et al., 2003, 2004) | • Employs marker based interface to avoid interface reconstruction for geometric-property computation.  
  • Improved performance in under-resolved areas. |
| LS and VOF           | Coupled LS -VOF (Sussman, 2003)            | • Excellent mass-conservation due to VOF.                                |
| Marker tracking      | The point-set tracking (Torres & Brackbill, 2000) | • A set of unconnected.                                                |
|                      | Connectivity free tracking with level-contour reconstruction (Shin & Juric, 2002) | • Lack of connectivity-information makes it easier to handle topological changes.  
  • Geometry-computations require spline-fitting.  
  • A set of unconnected triangles. 
  • Possible to compute geometry (curvature, normal) without any surface fitting or interpolation. 
  • Level-contour reconstruction for topology change. |

Figure 2-5. A surgical alteration of the interface data to accomplish topology change.

Figure 2-6. A three-dimensional marker based film boiling computation. (a) A snapshot of the computational domain and (b) a triangulated interface before and after pinch-off from Shin and Juric (2002). [Reprinted with permission]
2.1.5 Present Approach: Markers with or without Connectivity

Since it is evident that the Eulerian type approaches are relatively easier, it makes the choice of the method difficult based on any single consideration. Since the most serious difficulty with marker based tracking is due to data-handling, the explicit knowledge of the interface shape/location and recent algorithmic advances in literature have been the criteria for choosing explicit marker based method.

The method of level-contour based reconstruction with connectivity-free marker-based tracking of Shin and Juric (2002) offers an attractive choice. However, the lack of connectivity information on the interface makes it difficult to locally alter the marker-spacing by marker addition and deletion to maintain a desired marker-spacing on the interface. As seen in Figure 2-7(a), deletion of an edge $e2$ requires collapsing of vertices of the neighboring edges (point $p2$ and $p5$) but the lack of connectivity makes the identification of edges $e1$ and $e3$ difficult i.e., no explicit information is available to suggest, for example, that nodes $p2$ and $p3$ have the same physical location. For computations such as bubbly flows by Esmaeeli and Tryggvason (1998, 1999), several bubbles (~100 or more) move in close proximity without merger. In these simulations, the bubbles are not allowed to merge with each other but local restructuring of the interface is still required to maintain desirable resolution. The connectivity-free method resorts to global reconstruction of the interface even to maintain the marker spacing and may inadvertently produce mergers. For such reasons, the current work maintains the connectivity information (Figure 2-7(b)) to allow greater flexibility in handling multiple interfaces. This decouples the procedure for handling interface resolution and interface reconstruction by using local restructuring when needed and performing reconstruction only to achieve topology changes.
2.2 Interfacial Dynamics Modeling

A moving/deforming interface in the computational domain presents some obvious difficulties in numerical computation of the flow field. The difficulties arise primarily due the nonlinear coupling of fluids and steep fluid-property jumps across the interface. In the event of phase change, effects such as mass and heat transfer across the interface also need to be considered. For simplicity, the basic introduction is provided here with respect to isothermal flows. Issues and techniques related to phase change modeling will be considered in subsequent sections and chapters.

In general, the pressure and viscous stresses show discontinuities across the interface that is related to the surface tension force and fluid property jumps. Figure 2-8 shows the pressure ($P$) and the normal shear stress components ($n \cdot \tau \cdot n$) across the interface and Equation (2.3) relates the jump in flow properties with the surface tension force. Based on the level of detail and accuracy in treatment of the discontinuities, the existing methods can be classified into two categories: sharp interface method (SIM); continuous interface method (CIM). The sharp interface method (SIM) is a class of techniques that attempt to satisfy the jump condition (Equation(2.3)) explicitly while
continuous interface method (CIM) represents techniques that smooth the jump over few cells across the interface.

\[
(P_2 - P_1) - n \cdot (\tau_2 - \tau_1) \cdot n = \sigma \kappa
\]  

(2.3)

Figure 2-8. Surface tension, pressure and normal component of shear stress acting on an interface.

2.2.1 Sharp Interface

There are three prominent techniques within this category: sharp interface of Ryskin and Leal (1984), Ye et al. (1999, 2001); immersed interface (LeVeque & Li, 1994; Li & Lai, 2001); ghost fluid (Liu et al., 2000; Kang et al., 2000). The technique of Ryskin and Leal (1984) and Ye et al. (1999, 2001) are the most uncompromising in terms of satisfying the jump conditions across the interface. The basic approach can be explained with the cut-cell method of Ye et al. (1999, 2001). This method cuts the background Cartesian grid with the interface and creates sub-domains (Figure 2-9(a)) that act like body fitted grids with arbitrary shaped cells on the interface. For stability reasons, the cut-cells are merged with suitable neighbors to remove arbitrarily small cells. The governing equations are solved separately in each sub-domain with the corresponding fluid properties. At every time step, an iterative procedure, by perturbing the interface in local normal direction, is followed to find the shape/location of the interface such that Equation (2.3) is satisfied along with the mass and momentum conservation equations.
The immersed interface and the ghost-fluid techniques use the jump conditions to construct boundary conditions for flow variables on the interface. As an example, the ghost-fluid technique solves the governing equations in each phase separately by creating ghost-cell values using the analytical jump condition. Unlike the sharp interface method of Ye et al. (1999, 2001), these techniques do not solve for the interface shape and flow field simultaneously or create cut-cells.

2.2.2 Continuous Interface

In contrast to the sharp interface method, the continuous interface method smoothes the fluid property and models the surface tension forces as a smoothed momentum source term (Peskin, 1977). This facilitates a single fluid formulation for the entire domain with the flow/fluid properties varying smoothly across the interface over a thin zone. This smearing/smoothing region across the interface shown in Figure 2-9(b) is usually three to four cells in thickness. The continuous interface method has been widely used with mixed Eulerian-Lagrangian tracking (i.e., the immersed boundary method) (Peskin, 1977; Tryggvason et al. 2001; Francois & Shyy, 2003) as well as Eulerian tracking such as level-set and volume-of-fluid.

2.2.3 Challenges and Recent Advances

The iterative procedure of coupling the phases across the interface and the complex procedure of cell-cutting makes the extension of the sharp interface technique of Ye et al. (1999, 2001) to three dimensions a tedious and challenging task. Although the ghost fluid is among the simplest sharp interface method to use, it has traditionally been used with level-set based interface representation. Although successful demonstration of ghost fluid type approach with Eulerian-Lagrangian tracking were made by Hao and Prosperetti
(2004) for some bubble dynamics problems, further research is required to establish the overall stability and performance.

Accuracy studies by Ye et al. (2004) and Shyy (2004) comparing the sharp interface and continuous interface methods for a spherical drop in static equilibrium sheds some light on their characteristic differences. With a spherical drop in static equilibrium, the difference in pressure is balanced by surface tension force according to the Young-Laplace Equation (2.4). The sharp interface method attempts to capture the pressure jump as a discontinuity whereas the continuous interface method smoothes the pressure jump over few cells (Figure 5-12(b)).

\[ \Delta P = \sigma \kappa \]  \hspace{1cm} (2.4)

For a static bubble computation, the theoretical solution produces uniformly zero-velocity in the domain. However, due to certain amount of net numerical stress-imbalance across the interface, computations tend to produce a non-zero velocity field termed as parasitic currents or spurious velocities (Figure 5-12(a)). These spurious velocities can destabilize the interface especially if viscous forces are not strong enough (Rider & Kothe, 1995). As an observation, the magnitude of the spurious velocity can be kept orders of magnitude smaller with the sharp interface treatment. The second order accurate sharp interface technique of Ye et al. (2004) could keep this velocity up to machine level as compared to continuous interface methods producing velocities of the order $10^{-4}$ (in terms on Capillary number $Ca$) (Tryggvason, 2001). The first order accurate ghost fluid technique could also keep these velocities to the order of $10^{-6}$ or lower (Kang et al., 2000). Several attempts using improved curvature/surface-tension
computation have also been attempted in literature to keep such erroneous velocities as low as possible (Renardy & Renardy 2002; Shin et al., 2005).

![Diagram of fluid interfaces](image)

Figure 2-9. Sharp and continuous interface methods. (a) Sharp interface cut-cell method and (b) a continuous interface using the immersed boundary technique.

2.2.4 Present Approach: Sharp or Continuous Interface

The continuous interface method provides a numerically stable system due to smoothing of discontinuities but leads to smearing of the interfacial effects. This aspect of the continuous interface method poses accuracy concerns for certain types of problems like solidification dynamics at morphological scales that require higher precision in interface treatment (Udaykumar et al., 1999).

For the reasons of computational efficiency and the relative difficulties in using the sharp interface method with marker based tracking, the present work employs the continuous interface method via immersed boundary technique. Despite the accuracy
issues related to smearing of discontinuities, it is still a reasonably accurate method for a large number of drop dynamics problems. Over the years, the immersed boundary technique has been used extensively to study a wide range of physical processes: Peskin (1977) for studying blood flow in heart-valves; Juric and Tryggvason (1996) for dendritic solidification; Agresal et al. (1998) for deformation and adhesion of circulating cells; Esmaeeli and Tryggvason (1998, 1999) for simulation of bubbly flows; Francois and Shyy (2002, 2003) for micro-scale drop dynamics.

The overall interfacial dynamics treatment in the presence of phase change is of a hybrid nature i.e., the mass and momentum equations are solved using the immersed boundary method, but the temperature equation is solved in a sharp fashion. The primary intention, as pointed out in previous chapter, is to accomplish higher accuracy in interfacial mass transfer computation. Due to the absence of a surface tension type source term in the temperature equation, sharp treatment of the temperature equation can be performed with relative ease as compared to a sharp treatment of the surface tension forces, pressure, etc. Since the interface has been assumed to be at saturation temperature at all times, it simplifies the process by allowing the interface temperature to be used as a boundary condition in the numerical discretization of the thermal diffusion term in temperature equation. This discretization approach is similar to the recent developments by Gibou et al. (2003) and Morgan (2005) using ghost fluid based sharp interface for phase change computations.

### 2.3 Adaptive Grid Computation

Most of the computational time is devoted to performing computation/operations on the interface and solution of the flow-governing equations. Usually the latter part is responsible for majority of the computational burden. Since the governing equations of
the flow are incompressible, the procedure involves solution of an elliptic equation to compute the pressure field making it one of the primary reasons for the high computational cost. Since the system has multiple fluid components, the jump in the fluid properties, in general, has an adverse bearing on the overall computation-time. Figure 2-10 from the work of Francois et al. (2004) is a representative case of degrading performance with increasing fluid property jump. For brief information, these computations were performed for a buoyancy-driven rising bubble using the immersed boundary method. The Reynolds number, Weber number and Froude number were set to 100, 4 and 1.0. A W-cycle multigrid technique was used to solve the pressure Poisson equation. The horizontal axis in Figure 2-10 shows the level of multigrid used for computation where \( level = 1 \) represents computation without employing any multigrid acceleration.

![Graph showing the effect of fluid property jumps on the computation time for a rising bubble.](image)

Figure 2-10. The effect of fluid property jumps on the computation time for a rising bubble. (a) Effect of density ratio; (b) effect of viscosity ratio (Francois et al., 2004). [Reprinted with permission]
The performance seems to be affected the most by the density ratio factor: without multigrid, the computation time for density ratio of 1000 is nearly ten times higher in Figure 2-10 than for a density ratio of 10. Although these observations come from a specific method and problem, they still represent the general trend (Francois et al., 2004; Lörstad & Fuchs, 2004) and recommend appropriate measures to reduce the computational cost as much as possible. The presented work uses adaptive Cartesian grids to resolve the flow features locally and thus attempts to reduce the overall computational data-size.

2.3.1 Cartesian Grid Data Structures

The approach for generating adaptive Cartesian grids is divided into two categories: a cell-by-cell adaptation (Figure 2-11(a)); a block-by-block adaptation (Figure 2-11(b)). Adaptive mesh refinement known as AMR is an example of a block-by-block adaptation (Berger & Oliger, 1984; Sussman, 1999, 2005) that interprets the grid as a union of uniformly refined grid-blocks (rectangles in 2D and hexagons in 3D). Due to the approach of collecting cells in blocks and refining them, AMR tends to produce a larger grid data-size than necessary. As observed by Aftosmis (1997), typically only 70% of the cells in a rectangular patch may actually need refinement. In the sense of reducing the computational data-size, cell-by-cell adaptation is a more flexible approach.

A cell-by-cell adaptation uses either a tree-based data (Wang, 1998; Popinet, 2003; Losasso et al., 2004) or an unstructured data format (Aftosmis, 1997; Ham et al., 2002; Singh & Shyy, 2005, 2006) to store the grid. A tree-based data-structure (Figure 2-12) offers a compact storage system. All the connectivity information is implicitly contained in the location of cells in the tree-structure. One of the most common queries in a
Numerical simulation is the cell-cell connectivity information needed, for example, to compute convection/viscous fluxes. Since no explicit information about the identities of the neighbor cells is stored, it needs to be extracted from the information contained in the tree. As reported by Aftosmis (1997), approximately 10-20% of the CPU time for CFD solvers is dedicated to tree traversals answering grid-connectivity queries.

Figure 2-11. A cell-by-cell and block-by-block grid adaptation. (a) A cell-by-cell adaptation in the present work; (b) schematic of a block-by-block adaptation.

Figure 2-12. A tree-based adaptive Cartesian grid-data. (a) A cell undergoing two levels of successive refinement; (b) data structure showing the place of cells based on the refinement level.
2.3.2 Present Approach: Unstructured Adaptive Cartesian Grid

An anisotropically adaptive grid (Ham et al., 2002; Singh et al., 2005) offers an optimized grid data by using a direction-based refinement strategy. However, the resulting data-structure and solution algorithms become difficult and tedious to implement. For simplicity and convenience, an isotropic adaptation is employed where a cell marked for refinement is simultaneously split in all coordinate directions. The flow-governing equations and numerical discretization aspects have been discussed in the related chapters.

The grid is stored using unstructured data with cell-by-cell refinement approach. It is believed that the extra memory requirements due to explicit storage of connectivity information will be offset in part by compact storage system exploiting the Cartesian nature of the grid and easier accessibility of the data for the flow solver.
CHAPTER 3
MULTIDIMENSIONAL IMMERSED BOUNDARY COMPUTATION

The immersed boundary method originally proposed by Peskin (1997) has been used to model the surface tension force as a smoothed volume source term in the momentum equation. The jump in density and viscosity are smoothed within few cells across the interface. The isothermal immersed boundary computation is similar to the work of Tryggvason et al. (2001). This chapter presents the key components of the immersed boundary modeling and interface tracking including algorithms for maintaining interface resolution and technique for handling topology changes. The numerical solution technique including adaptive grid generation and equation discretization are presented in the following chapter.

3.1 Immersed Boundary Method

The incompressible Navier-Stokes equations for mass and momentum and energy conservation represented by Equations (3.1), (3.2) and (3.3) are solved on the stationary background grid. The source terms $F_s$ models the surface tension as a smoothed momentum source term using a smoothed Dirac delta function (Figure 3-1(a)). The fluid properties are smoothed using an indicator function $I$ that varies smoothly from a value of unity outside the interface to zero inside (Figure 3-1(b)). The density and viscosity are smoothed using Equation (3.4) (Prosperetti, 2002) where subscripts 1 and 2 refer to the fluid outside and inside the interface, respectively.

The source term $Q_s$ in the energy equation accounts for the heat source due to phase change. Since the interface is considered at saturation temperature, the job of this term in
the immersed boundary method of Juric and Tryggvason (1998) is essentially to maintain the interface temperature. Despite smeared treatment of the hydrodynamics equations, the temperature equation in the presented work is solved in a sharp fashion to compute the mass-transfer accurately. Since the temperature equation treatment is different than the traditional immersed boundary method, role of the source term $Q_s$ and corresponding numerical solution procedure will be visited in the following chapter.

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{U} = 0 \tag{3.1}
\]

\[
\rho \left( \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{U} \mathbf{U} \right) = -\nabla P + \nabla \cdot \mu \left( \nabla \mathbf{U} + \nabla^T \mathbf{U} \right) + \mathbf{F}_s + \rho \mathbf{g} \tag{3.2}
\]

\[
\rho C \left( \frac{\partial T}{\partial t} + \nabla \cdot (T \mathbf{U}) \right) = \nabla \cdot (K \nabla T) + Q_s \tag{3.3}
\]

\[
\rho = \rho_2 + (\rho_1 - \rho_2) I
\]

\[
\frac{\rho}{\mu} = \frac{\rho}{\mu_2} + \left( \frac{\rho}{\mu_1} - \frac{\rho}{\mu_2} \right) I \tag{3.4}
\]

Figure 3-1. Schematic of the immersed boundary method. (a) Interfacial effects smoothed over a thin zone across interface; (b) indicator function for fluid property smoothing.
3.1.1 Interfacial Conditions

The mass conservation equation across the interface is written as Equation (3.5). The jump in pressure across the interface is related to the surface tension force, jump in shear stress and fluid velocities using Equation (3.6). Using the mass continuity equation, the jump in the normal component of the velocity can be written using Equation (3.7). The tangential component of the velocity is continuous across the interface and the normal component is also continuous if there is no mass transfer.

\[
\rho_1 (U_1 - U_{int}) \cdot n = \dot{m}
\]
\[
\rho_2 (U_2 - U_{int}) \cdot n = \dot{m}
\]

(3.5)

\[
\rho_2 \left( (U_n)_2 - (U_n)_{int} \right) \cdot U_2 + P_2 \cdot n - \tau_2 \cdot n = \rho_1 \left( (U_n)_1 - (U_n)_{int} \right) \cdot U_1 + P_1 \cdot n - \tau_1 \cdot n + \sigma kn
\]

(3.6)

\[
(U_1 - U_2) \cdot n = \dot{m} \left( \frac{1}{\rho_1} - \frac{1}{\rho_2} \right)
\]

(3.7)

The mass transfer rate across the interface is computed using Equation (3.8) for which a probe based technique is used to compute the required temperature gradients on the interface. These gradients are computed using Equation (3.9) and Figure 3-2 showing two probes of length $1.2\Delta$ along the interface normal direction. The term $\Delta$ is the background grid size and $T_{int}$ is interface temperature (set to saturation temperature $T_{sat}$). The temperature $T_1$ and $T_2$ on the probes are computed using bilinear interpolation from the background.

\[
\dot{Q} = (-KVT \cdot n)_1 - (-KVT \cdot n)_2 = \lambda \dot{m}
\]

(3.8)

\[
(KV T \cdot n)_1 = \frac{T_1 - T_{int}}{1.2\Delta}
\]

(3.9)

\[
(KV T \cdot n)_2 = \frac{T_{int} - T_2}{1.2\Delta}
\]
Using the mass continuity across the interface, the interface velocity can be written as:

\[ U_{\text{int}} \cdot n = \left( \frac{U_1 + U_2}{2} \right) \cdot n - \dot{m} \left( \frac{\rho_1 + \rho_2}{2} \right) \]  

(3.10)

The interface velocity due to mass-transfer component is computed with the help of Equation (3.8) and unsmoothed fluid properties. The interface velocity component due to fluid velocity is simply interpolated from the background grid using a Dirac delta function described later.

The velocities in individual phases are considered incompressible and the jump in velocity is smoothed using Equation (3.11) (Juric & Tryggvason, 1998). The effect of mass transfer is modeled using locally non-divergence-free velocities near the interface. The Equation (3.12) relates the divergence of the velocity field with the interfacial mass-transfer after taking divergence of Equation (3.11) and applying incompressibility condition for velocities in the individual phases where \( U_1 \) and \( U_2 \) are fluid velocities outside and inside the interface.

\[ U = U_2 + (U_1 - U_2) I \]  

(3.11)

\[ \nabla \cdot U = (U_1 - U_2) \cdot \nabla I = \left( \frac{1}{\rho_1} - \frac{1}{\rho_2} \right) \dot{m} \cdot \nabla I \]  

(3.12)

Figure 3-2. Probe based technique for temperature gradient computation on the interface.
3.1.2 Material Property Smoothing

Figure 3-3 shows a common finite-element type data-structure used to store the triangulated interface. It assigns a unique identification (integer number) to each node (marker) and stores its coordinates along with the triangle-nodes. All the triangles are assigned a unique orientation such that the right hand thumb rule using the ordering of nodes of a triangle gives the local normal-vector pointing outside the interface.

<table>
<thead>
<tr>
<th>Node ID</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>2</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>nnode</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Triangle ID</th>
<th>node1</th>
<th>node2</th>
<th>node3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>2</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>ntria</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

Figure 3-3. A triangulated interface and the data-structure.

The indicator function is computed at Cartesian cell centers and linearly interpolated to cell faces whenever required. It is computed by solving the Poisson Equation (3.13) where $\delta_{(X-X_{\text{int}})}$ is a Dirac-delta function non-zero only at $X=X_{\text{int}}$ and $D$ is its discretized form for which Equation (3.14) (Peskin & McQueen, 1996; Griffith & Peskin, 2005) is used.

Since the fluid properties are constant in most part of the domain, the indicator function is solved only within few cells (4~5) across the interface. For rest of the cells, the indicator value is set manually. The cells completely outside the interface (Fluid 1) are assigned a material flag MatFlag(cell) = 1 and cells completely inside the interface (Fluid 2) are set to MatFlag(cell) = 2 (Figure 3-4). This knowledge is used to manually set the indicator function to zero or one and use it as a Dirichlet boundary condition for Equation(3.13).
\[ \nabla^2 I = \nabla \cdot \left( \int_{\text{interface}} n\delta_{(s,x)} dA \right) \approx \nabla \cdot \left( \sum_{\text{triangle}} nD(r)dA \right) \]  

(3.13)

\[ D(r) = \frac{\delta(r_x/\Delta_x)\delta(r_y/\Delta_y)\delta(r_z/\Delta_z)}{\Delta_x\Delta_y\Delta_z} \]

\[ \delta \left( h = \left| \frac{r}{\Delta} \right| \right) = \begin{cases} 
\frac{1}{8} \left( 3 - 2h + \sqrt{1 + 4h(1-h)} \right) &: \text{if } h \in [0,1) \\
\frac{1}{8} \left( 5 - 2h - \sqrt{-7 + 4h(3-h)} \right) &: \text{if } h \in (1,2] \\
0 &: \text{else} 
\end{cases} \]  

(3.14)

A ray-tracing algorithm popular in computer-graphics and computational-geometry community could be employed for accurate evaluation of the state of a cell i.e., inside or outside the interface. However, since such accurate information is required only outside the smearing zone, a simple and efficient method based on the painter’s algorithm frequently used in computer-graphics rendering is used. Unlike the ray-tracing algorithm, the painter’s algorithm does not require expensive computation of three-dimensional line-surface intersection and it is sufficient for the current purposes to deal with simply connected closed topologies. Open surfaces can also be dealt with ease following suitable algorithmic considerations. This algorithm stores the state of a cell in an array named MatFlag(cell). This approach exploits the fact that a cell outside the interface has all its neighbors either outside or on the interface. The outline of the implemented algorithm is presented below:

- Set MatFlag(cell) = 2 for all the cells contains a marker else set it to zero.
- Select any cell outside all interfaces, call it SeedCell, and put it as the first entry in a temporary array named SeedCellList() .
- Set TotalSeedCell = 1; SeedCellList(TotalSeedCell) = SeedCell .
- Set MatFlag(Seedcell) = 1
- Use a temporary variable NextCheck to march through the array SeedCellList() .
  Set NextCheck = 1
• Do while (\( \text{NextCheck} \leq \text{TotalSeedCell} \) )
  o \( \text{SeedCell} = \text{SeedCellList}(\text{NextCheck}) \)
  o Go through the neighbor cells which share a face with \( \text{SeedCell} \)
    - If (\( \text{MatFlag}(\text{NeighborCell}) > 0 \)) Go to another neighbor cell
    - \( \text{TotalSeedCell} = \text{TotalSeedCell} + 1 \)
    - \( \text{SeedCellList}(\text{TotalSeedCell}) = \text{NeighborCell} \)
    - \( \text{MatFlag}(\text{NeighborCell}) = \text{MatFlag}(\text{SeedCell}) (= 1) \)
  o \( \text{NextCheck} = \text{NextCheck} + 1 \)
• End Do While Loop
• Set \( \text{MatFlag}(\text{Cell}) = 2 \) for all the cells which still have \( \text{MatFlag}(\text{Cell}) = 0 \)

![Diagram of fluid interfaces with material flags](image)

Figure 3-4. Indicator function and the material flags for cells completely outside or inside the interface.

3.1.3 Momentum Source Term Computation

The surface tension force is computed on the interface triangles and distributed to the background grid as smoothed source term \( F_s \) in Equation (3.2). Figure 3-5 shows the smoothed nature of the momentum source for a two-dimensional interface. The surface tension force on a discretized interface element (curves in 2D and triangles in 3D) can be evaluated in several ways: computation with Equation (3.15) where unit normal vector and curvature can be computed using curve fitting for two-dimensional interfaces (Francois & Shyy, 2002, 2003; Ye et al., 2001) and surface fitting for three-dimensional interfaces (Sousa et al., 2004); computation using a line integral form shown in Equation (3.16) and fitting curves/surfaces to obtain normal and tangent vectors (Al-Rawahi & Tryggvason, 2002; Tryggvason et al., 2001).
\[
\delta f = \int_{\partial A} \sigma k n dA \quad (3.15)
\]
\[
\delta f = \int_{\partial A} \sigma k n dA = \int_{\partial A} \sigma (n \times \nabla) \times \nabla dA = \int_{s} \sigma t \times nds \quad (3.16)
\]

There are two important observations to be made here: the net surface tension force on a closed surface should be zero (conservation); curvature computation using interpolation based methods are numerically sensitive and often requires some form of data smoothing (Francois & Shyy, 2002, 2003; Ye et al., 2001; Sousa et al., 2004). The use of Equation (3.15) does not enforce conservation whereas the line-integral form does not require explicit curvature computation and maintains the conservation. The present work uses the line integral form and computes the local normal and tangent vectors along the triangle edges using the simple approach of Al-Rawahi and Tryggvason (2004) shown in Figure 3-6. To ensure conservation (i.e., the net surface tension force on a closed interface is zero), the cross-product of normal and tangent on a triangle edge is obtained by averaging its value between the triangles sharing this edge. The surface forces computed on triangle edges are distributed directly to the background grid using Equation (3.19). If required, the curvature can be computed using Equation (3.18). Such a simple technique is seen to produce sufficient accuracy demonstrated by Figure 3-7 comparing curvature of a unit circle using present method and a cubic-spline interpolation. The overall accuracy of the present surface tension force computation and modeling has already been demonstrated by Shin and Juric (2002) for boiling flows and Al-Rawahi and Tryggvason (2004) for dendritic solidification.

\[
\delta f = \sum_{\text{edge}=1,2,3} \sigma (t \otimes n)_{\text{edge}} ds \quad (3.17)
\]
\( \kappa = \frac{\delta f \cdot n}{\sigma dA} \)  
(3.18)

\[ F_s = \sum_{\text{edge}} f_{\text{edge}} D(r) \]  
(3.19)

Figure 3-5. Momentum source term due to surface tension force on a circular interface.  
(a) The x-component, (b) y-component and (c) the x-component along a horizontal line though the center of circle.

Figure 3-6. Computation of the unit normal and tangent vectors on interface triangles.
3.1.3 Interface Advection

The interface velocity in Equation (3.10) has a component due to the fluid motion and another due to phase change. The component due to phase change is obtained by computing the mass transfer rate on the interface markers, as described earlier. The component due to fluid velocity is interpolated from the background grid and the final marker velocity expression is:

\[
U_{\text{int}} = \sum_{\text{cell}} U_{(\text{cell})} D(r) dx dy dz - m \left( \frac{\rho_1 + \rho_2}{2 \rho_1 \rho_2} \right)_n
\]  
(3.20)

For simplicity, the marker movement due to background fluid velocity is solved using 2\textsuperscript{nd} Order Runge-Kutta method but the velocity due to phase change is treated using forward Euler method as shown in Equation (3.21).

\[
\begin{align*}
X' &= X^n + \Delta t U'_{(x')}, \\
X'' &= X' + \Delta t U'_{(x')}, \\
X^{n+1} &= \frac{X^n + X''}{2} - n\Delta t \left( \frac{\rho_1 + \rho_2}{2 \rho_1 \rho_2} \right)_m
\end{align*}
\]  
(3.21)
Error associated with interpolated marker velocity and integration of the advection equation produces some error in the interface volume. These errors are usually small but can accumulate over time and become unacceptably large. To correct such defects, an explicit phase-volume correction step is performed by perturbing the markers in local normal direction and applying bisection method to find a location that keeps the volume error within some prescribed level ($\sim 10^{-7}\%$). This correction may be performed either at every time-step or periodically. All the presented computations performed this operation at every time step and usually one or two bisection iterations were found to be sufficient. For flows without phase change, the interface volume does not change and hence the amount of volume error and hence the correction is always known. For phase change computations, the amount of volume error and thus the correction was estimated based on the mass-transfer rate.

3.2 Conservative Restructuring for Interface Resolution

Due to the limited support of the Dirac delta function used for distribution of surface forces and other interpolation operations, excessively long edges (e.g. more than two Cartesian-cell-size) will appear as a ‘hole’ on the interface. On the other hand, excessively small edges can produce unphysical sub-grid undulations. To ameliorate such effects, markers are continuously added and deleted based on the following criteria:

\[
\begin{align*}
\text{Edge}_{-} \text{length} > \text{Cell}_{-} \text{size} & \quad \Rightarrow \text{Break edge (marker addition)} \\
\text{Edge}_{-} \text{length} < \frac{\text{Cell}_{-} \text{size}}{3} & \quad \Rightarrow \text{Delete edge (marker deletion)}
\end{align*}
\]

(3.22)

It is required that the marker addition and deletion causes minimal disturbances on the interface and avoids introducing volume errors. However, the commonly employed techniques in literature (e.g. Figure 3-8) do not preserve the interface volume. More
precisely, it is the marker deletion that is non-conservative. Such non-conservative behaviors are demonstrated by Sousa et al. (2004) who encountered up 5% volume errors due to interface restructuring.

Figure 3-8. Interface restructuring approach. (a) Long edges are split at the mid-point and (b) a small edge $p_ip_2$ is collapsed to its midpoint $p_3$.

The presently employed restructuring approach is same as in Figure 3-8 with an added correction step to the marker deletion procedure in order to preserve the interface volume. This correction-algorithm is an extension of the volume preserving marker relocation of Sousa et al. (2004) who used such an approach for interface smoothing and removal of sub-grid undulations on the interface. The underlying procedure is explained using a 2D example shown in Figure 3-9 that shows a marker removal by collapsing the edge formed by point $p_2$ and $p_4$ to a point $p_3$ resulting in a net phase-area loss. To correct this error, first a reference point $p_{ref}$ at $p_3$-$n$ is selected (Figure 3-9(b)) where $n$ is the local unit normal vector at $p_3$. A reference phase area of the polygon $p_1p_2p_4p_5p_{ref}$ is computed and point $p_3$ is relocated after deleting the edge to recover preserve the reference area (Figure 3-9(c)). The conservative marker/edge deletion on 3D surfaces is similar to its 2D counterpart and it is summarized below:
1. Select a point \( p \) at the midpoint of the edge to be deleted. A local normal vector \( n \) at this point is taken as the average of the normal vector on the triangles connected to this edge.

2. Define a reference point \( p_{\text{ref}} \) as \( p-n \) and compute a reference volume \( \Omega_0 \) by adding the tetrahedral-volumes made by the reference point and the triangles in Figure 3-10(a).

3. Collapse the edge to its midpoint and compute a volume \( \Omega \) by adding the volumes of tetrahedrons made by point \( p \) with the triangles in Figure 3-10(b).

4. Relocate the point \( p \) to \( p + n \left( \Omega_0 / \Omega \right) \) for preserving the local phase volume \( \Omega_0 \).

---

Figure 3-9. Conservative marker deletion for two-dimensional interfaces. (a) Old interface \( p_1p_2p_4p_5 \) and new interface \( p_1p_3p_5 \) with non-conservative method; (b) define a reference point \( (p_{\text{ref}}) \) and compute a reference area; (c) relocate \( p_3 \) in local normal direction to preserve the reference area.

Figure 3-10. Conservative marker deletion for three-dimensional interfaces. (a) A reference point \( p_{\text{ref}} \) is defined at \( p-n \) (\( p \) is edge mid-point) and compute a reference, (b) collapse edge to point \( p \) and move \( p \) along normal vector to preserve the reference volume.
3.3 Interface Reconstruction for Topology Change

When two interfaces come within ‘close proximity’ they are merged together and similarly when sections of the same interface are in ‘close proximity’ the interface is broken. The term ‘close proximity’ at this instance is intended to be around one grid-cell size and will be addressed further in the document. Since the indicator function reflects the interface shape, it is used to redefine the interface to achieve topology changes.

Figure 3-11 shows a 2D interface and indicator contours along with indicator = 0.5 contour exhibiting interface merger. The work of Shin and Juric (2002) exploited this idea to reconstruct the interface by deleting the interface data completely and reconstructing a new interface at indicator = 0.5 contour. Their work tracked the interface using a connectivity free method and needed to perform reconstruction to control interface resolution as well as facilitate topology changes. Such a situation does not arise in the current work as the resolution is controlled via conservative restructuring while reconstruction is performed only to achieve topology change.
3.3.1 Reconstruction Criteria

The need for reconstruction is checked at prescribed intervals and interfaces are reconstructed based on some indication of a possible topological change. A simple criterion shown in Figure 3-12 sends two probes along the local normal vector from the marker points. The length of each probe is set to one and half times the background grid-cell-size. The indicator function value on the probes is interpolated from the background grid using bilinear interpolation. If both the probes have indicator function value greater than half or less than half, representing merger or breakup for the given grid resolution, reconstruction is carried out. The need for reconstruction is checked at frequent intervals depending on the problem. When uncertain, it was checked at every few (e.g. 50) time steps in the presented computations.

![Figure 3-12](image)

Figure 3-12. Interface reconstruction criterion uses two probes from the interface along the normal vector and bi-linearly interpolates the indicator values \( \text{Ind1} \) and \( \text{Ind2} \).

3.3.2 Reconstruction Procedure and Difficulties

The approach for reconstruction is to first create a set of globally numbered markers, one on each edge with indicator = 0.5 contour intersecting it. Cells with new markers are visited one at a time and markers on the cell-edges are connected to define interface segments within the Cartesian cells (Figure 3-13).
The following sections present the overall reconstruction procedure using a two-dimensional interface and the ideas are extended to three-dimensional interfaces. Some critical difficulties in maintaining the integrity of the connectivity data are highlighted and simple remedies are provided.

Figure 3-13. Two dimensional reconstructed interface edges (e1, e2 and e3) and corresponding connectivity data.

3.3.2.1 Two-dimensional interface

The indicator function values computed at cell centers are linearly interpolated to Cartesian cell-vertices. The vertices are flagged and a rule for marker creation on an edge is set as:

- vertex-flag = \[
\begin{cases} 
1 & : \text{indicator} > 0.5 \text{ (outside interface)} \\
0 & : \text{otherwise (inside/on the interface)} 
\end{cases}
\]

- Marker is created on an edge only if its two vertices have different flag values.

The above rules avoid ambiguities and simplify the algorithm by eliminating the need to look into markers in neighboring cells during reconstruction in a given cell. As an example, consider Figure 3-14(a) where an edge e2 is constructed only while visiting cell1 and not while in cell2 even though the corresponding markers physically lie on the edges of both these cells but are considered only on the vertical edges of cell1 due to their opposite vertex flag values. This procedure, however, allows creation of edges with zero lengths in order to create a valid connectivity data. Such a situation arises when, for
example, the reconstructed markers $p_1$, $p_2$ and $p_3$ in Figure 3-14(b) all same physical location i.e., indicator = 0.5 contour is at the Cartesian-cell vertex. No special attention is paid to such cases as a valid connectivity data is still produced while such edges/elements are later deleted by the marker/edge deletion procedure presented earlier.

![Diagram](image)

Figure 3-14. Illustration of 2D cell-by-cell interface reconstruction. (a) Markers on edges with opposite vertex-flag values (0 and 1) are connected to form edges $e_1$, $e_2$, and $e_3$ in cells $cell_3$, $cell_1$ and $cell_4$, respectively; (b) markers $p_1$, $p_2$ and $p_3$ are connected in $cell_1$ and $cell_2$ irrespective of their physical coordinates.

A 2D cell can have two, three or four markers. Cells with two markers produces single edge; cell with three markers produce two edges connected within the cell; cell with four markers are used to produce two edges non-connected inside the cell. It is last case that causes some difficulty in deciding the orientation of the two disconnected edges. These four markers are not used to create a closed interface within the cell as it is not required since the interface will be closed in some other cell easily deduced by using the vertex flag information.

It can be seen that if the interface was to be reconstructed in triangulated cells, such ambiguities with four markers do not arise as a triangular cell can have only two or no markers producing at the most single edge within the triangle. Although, the markers are
created only on Cartesian cell edges, a set of locally triangulated cells are used to
decipher the connectivity/orientation of the edges whenever needed as shown in Figure 3-
15 where the one vertex of the triangles in the cell is at the Cartesian cell-center. The
indicator function is known at the cell center and hence a vertex flag value can be
assigned.

![Figure 3-15](image)

Figure 3-15. A 2D cell with four markers creating two locally disconnected edges. The
connectivity of markers is decided based on the vertex flags of the temporarily
triangulated cell.

3.3.2.2 Three-dimensional interface

The basic approach for three-dimensional interfaces is similar to the two-
dimensional counterpart i.e., create globally numbered markers on cell edges and connect
within one cell at a time. Figure 3-16 shows a schematic reconstruction by creating
markers on edges that have opposite vertex flags and connecting them to create polygons.
The reconstructed polygons are broken into triangles by inserting a marker at the
geometric center of the polygon. Figure 3-17 shows some typical reconstructed polygons
with corresponding vertex-flags. The last two cases in Figure 3-17 contain six markers
each but one has a single six-sided polygon while the other has two triangles. These two
cases with six markers are distinguished from each other by inspecting the relative
location of edges containing the markers.
Figure 3-16. A reconstructed interface segment in a cell. (a) Reconstructed markers are connected to produce a polygon \((p_1 \ p_2 \ p_3 \ p_4 \ p_5)\); (b) polygon is broken into triangles with a new marker at the geometric center.

Figure 3-17. Some common reconstructed interface segments and integer vertex-flags.

Since the indicator function itself is smooth in nature, polygons shown in Figure 3-17 are observed the most often. Similar to 2D reconstruction, occasionally two or more disconnected polygons within a cell are observed. Presence of more than six markers in a 3D cell presents an obvious algorithmic difficulty in terms of how to define the surface segments using the cloud of markers. As shown in Figure 3-18(a) eight markers could be connected in at least two ways to define two differently oriented and connected polygons.
If the reconstruction were to be conducted in tetrahedrons, only three or four markers and hence just one polygon is present (Figure 3-18(b)). Similar to 2D case, whenever required the cells are locally broken into tetrahedrons (Figure 3-18(c)) by connecting the cell-center to cell-vertices in order to decide the connectivity. Although only the markers created on Cartesian edges are used to construct the interface, possible presence of markers on tetrahedron edges (based on vertex flags for tetrahedrons) are used simply as a means to decide how to connect marker on Cartesian cell edges in case there are more than six of them in a cell.

Figure 3-18. Difficulties due to degeneracy of reconstructed interface data and implemented remedy. (a) A cell containing eight markers and one of several ways to define reconstructed polygons; (b) a tetrahedron can have only one polygon; (c) some edges of tetrahedrons created in a Cartesian cell.

The newly created triangles are oriented using the gradient of the indicator function (Figure 3-19(a)) to point the normal vector in the direction of increasing indicator value. Triangles with zero or arbitrarily small area can not be reliably oriented and may compromise the integrity of the interface data. The orientation of all such triangles is recursively corrected by using orientation of the neighboring triangle (Figure 3-19(b)). The triangles on the entire interface can be correctly oriented starting with proper
orientation of any single triangle. Figure 3-20 shows examples of reconstruction creating a rupture and merger of two interfaces.

Figure 3-19. Orientation setting of reconstructed interface. (a) Orientation of triangles is set in the direction of increasing indicator function value; (b) orientation of a triangle (tria) is used to set the orientation of its neighbor (nghtria).

Figure 3-20. Examples of 3D interface reconstruction. (a) A surface undergoing rupture, (b) two spheres merging.
3.4 Summary

This chapter presented the immersed boundary formulation and other computational components including the interface tracking and data-management. The key aspects of the presented method are summarized below:

- **Immersed boundary method**
  - Immersed boundary flow equations and interfacial conditions.
  - Fluid property smoothing using an indicator function.
  - Surface force computation and modeling.

- **Interface tracking**
  - A conservative algorithm for interface restructuring.
  - A level contour based reconstruction algorithm to handle topology changes highlighting several difficulties and strategies employed to address them.

The interface restructuring and reconstruction form the key contributions of the present work while using the well-established immersed boundary method to model the interfacial phenomenon except the phase change. The temperature equation is treated sharply and the solution algorithms are presented in the following chapter.

The interface reconstruction introduces some amount of error in the interface-volume which is corrected explicitly by perturbing the markers in the local normal direction. The magnitudes of volume errors and required correction are estimated in chapter 5 along with the effect of reconstruction on the accuracy of flow computations via measurement of spurious velocities for static bubble computations. The effect of conservative restructuring is also demonstrated in chapter 5 via a spherical interface placed in a time-reversed vortex field.

The marker tracking by itself is not restricted to any particular choice of interfacial dynamics modeling; it may be used for sharp interface modeling such as ghost fluid method. Such a sharp interface modeling avoids smearing of fluid properties and reduces
the spurious currents. So far, the only known work using marker based tracking with sharp interface ghost fluid method is by Hao and Prosperetti (2004). They used indicator function to compute the geometric properties of the interface for relatively simple computations not exhibiting topology changes. The indicator function was used to assign the fluid properties sharply as done for the temperature equation described in the following chapter. For such a sharp interface method, several stability and accuracy related practical issues caused by sub-grid undulations, interface reconstruction and errors in curvature computation need to be examined closely.
CHAPTER 4
ADAPTIVE CARTESIAN GRID BASED COMPUTATION

The dynamic grid adaptation is based on the interface location and some feedback from the flow solution. The grid is stored using an unstructured data format and flow equations are solved using projection method. This chapter presents the grid generation/adaptation method along with the algorithms for Navier-Stokes computation.

4.1 Data Structure

Figure 4-1 shows a face-based data-structure used for storage of an isotropically refined grid. Since detailed information may be located in the work of Aftosmis (1997), Ham et al. (2002), Singh et al. (2005), Singh and Shyy (2006), only a brief description of the data-structure is represented here. The variable level in Figure 4-1 stores the information about how many times a cell has been split from an initial state. The integers $i$, $j$ and $k$, together with the level information provide the size as well as the physical location of a cell. Also, all the faces forming a cell are stored to provide cell-to-face connectivity information. For every face, a single byte integer named orientation gives the orientation of the face: faces normal to x-axis have $orientation = 1$, faces normal to y-axis have $orientation = 2$ and faces normal to z-axis have $orientation = 3$. The variable sideCell1 is identity of a cell sharing this face in the direction of orientation and sideCell2 is the cell on the other side of this face.

The size $(dx, dy, dz)$ and center $(xc, yc, zc)$ of a 3D cell with coordinates $(level, i, j, k)$ are computed using Equation(4.1). Other relevant information such as cell volume and face area can be computed and stored in a very compact way. Since the volume of a cell
is depends on its level, it can be pre-computed and stored for up to the maximum allowed level of refinement. Similarly the area of a face is a function of only its orientation and the smallest of level of two side cells and hence it can be pre-computed and stored as well.

\[
(dx, dy, dz) = \left( \frac{L_x}{Nx} \frac{1}{2^{level}}, \frac{L_y}{Ny} \frac{1}{2^{level}}, \frac{L_z}{Nz} \frac{1}{2^{level}} \right)
\]

\[
(xc, yc, zc) = ((i - 0.5)dx, (j - 0.5)dy, (j - 0.5)dz)
\]

Figure 4-1. Cartesian grid data-structure. (a) Cell and face data; (b) the convention for two side cells for a face with orientation along x, y or z coordinates.

The grid generation process starts by uniformly dividing the entire domain into prescribed number of cells in each coordinate direction. All the cells in this uniform grid are referred to as base cells upon which adaptation is carried out. These base cells are the largest cells possible in the computational domain and have their level coordinate set to zero. To briefly explain the process, consider a 2-D domain in Figure 4-2(a) uniformly split into four parts in x-direction and three parts in y-direction. Let the number of uniform cells in x and y directions be Nx and Ny, respectively. In two-dimensions, each cell has three integer coordinates (level, i, j). After the initial uniform splitting of the computational domain, all the cells are marked as ground level cells i.e., level = 0. The adaptation is carried out by splitting cells wherever desired. The final grid is obtained by
recursively splitting the cells to desired level of resolution. When a cell with coordinates 
(level, i, j) in Figure 4-2 (b) is isotropically split, coordinates of the new cells shown in 
Figure 4-2(c) are computed as:

\[ level' = level + 1 \]
\[ (i', j') = (2i - 1, 2j - 1) \]
\[ (i'', j'') = (2i, 2j) \] \hspace{1cm} (4.2)

Figure 4-2. Adaptive grid refinement. (a) A 4x4 base grid (level = 0), (b) coordinates of a 
two-dimensional cell before splitting and (c) after splitting.

4.2 Grid Adaptation

Interface location and shape provides the geometry based adaptation criteria. In 
addition to geometry, the flow-solution is also used as an indicator for adaptation. The 
refinement biased algorithm for isotropic adaptation is similar to the work of Ham and et 
al. (2002) i.e., when a cell flagged for refinement is to be broken; any cell blocking the
refinement is also broken. This makes the adaptation a refinement biased recursive process. The coarsening however is not a recursive process; a cell is coarsened only if it is permissible by the constraints due quality considerations and the data-structure. Coarsening is achieved by combining four cells in 2D or eight cells in 3D only if all these candidate cells are marked for coarsening. Following three restrictions for quality control were imposed on the adaptation process:

1. Up to a prescribed number of cell layers (~10) across the interface are always at maximum resolution to push errors due to non-uniform cells away from the interface. Uniform grid around the interface also allows simple finite difference solution of the Poisson equation for indicator function in chapter 3.

2. Cells sharing a face are not allowed to differ by more than one level of refinement for smooth variation in grid size (Figure 4-3)

3. Corner cells are not allowed to differ by more than one level of refinement (Figure 4-3). This criterion is not a requirement when using collocated grid based computation (Singh & Shyy, 2006). For the implemented staggered grid algorithm, this is an essential requirement to facilitate a convenient programming implementation.

4.2.1 Interface Geometry based Adaptation

All the cells cut by the interface are flagged for refinement. To facilitate a smooth transition in the cell size, up to five layers of cells around a flagged cell are also refined. This process of flagging and splitting cells is carried out recursively until the prescribed
grid resolution is obtained. The Figure 4-4 shows the process of geometry based adaptation starting from a base grid and refining up to six levels. For demonstration purposes, only two additional layers of cells around an interface-cell were refined.

![Figure 4-4. Geometry based adaptation. (a) Initial 5x5 base grid and grid with maximum refinement level of (b) one, (c) three and (d) six.](image)

4.2.2 Flow Solution based Adaptation

Cells away from interface (outside ~10 cells across the interface) are adapted based on the flow solution. The present implementation uses a curl based adaptation criteria (Wang, 1998) that computes a parameter $\zeta$ for each cell as shown in Equation (4.3). The length scale $l$ is estimated as the cubic root of cell-volume. The decision to refine or
coarsen a cell is made by comparing $\zeta_{cell}$ to the standard deviation (Equation(4.3)) using the criteria in Equation(4.4).

\[
\zeta_{cell} = \| \nabla \otimes U \|^ {3/2}
\]

\[
\sigma^i = \frac{1}{N_{cell}} \sum_{i=1,N_{cell}} \zeta_i^2
\]  

(4.3)

$\zeta_{cell} > \sigma^i \Rightarrow$ Refine cell
$\zeta_{cell} < 0.1\sigma^i \Rightarrow$ Coarsen cell  

(4.4)

During the adaptation procedure, the Cartesian cell center values such as pressure, temperature and face normal velocities need to be reconstructed for the newly created cells and faces. The cell-centered values are reconstructed linearly while the face-normal velocities are reconstructed using the divergence free reconstruction due to Balsara (2001). Flow variable reconstruction during cell and face coarsening is performed simply by averaging of the corresponding cell-centered or face-centered values.

### 4.3 Immersed Boundary Solution Procedure

The following seven steps present the overall structure of the computational algorithm:

1. Interpolate marker velocities from background grid and due to phase change.
   - Compute the mass transfer rate $\dot{m}$ on the markers.
   - Advect markers using 2nd Order Runge-Kutta for interpolated velocity and forward Euler for the velocity due to mass transfer.

\[
X^* = X^n + \Delta t U_{(x')}
\]

\[
X^{**} = X^* + \Delta t U_{(x')}
\]

\[
X^n + 1 = \frac{X^n + X^{**}}{2} - n\Delta t \left( \frac{\rho_1 + \rho_2}{2\rho_1 \rho_2} \right) \dot{m}
\]

- Estimate interface volume change due to mass transfer and explicitly correct any volume error.

2. Check and restructure interface to maintain resolution.
3. Compute indicator function and smoothed fluid properties.
4. Compute surface tension forces and distribute as momentum source term.
5. Solve Navier-Stokes equation to advance flow field to next time level.
6. Check for geometry and solution based grid adaptation at prescribed intervals.
7. Check for interface reconstruction at prescribed intervals:
   - Perform reconstruction when required.
   - Explicitly correct any global phase volume error caused by reconstruction. The magnitude of volume correction in a reconstructed interface is inversely proportional to its volume so that lesser correction is applied to smaller entities.
   - If multiple interfaces are present, the reconstruction is selectively performed only for the interfaces that request it based on the probe based criteria presented in the previous section.

The first four steps and the step 7 have been described in the previous chapter. Of the remaining two steps, the geometry and solution based adaptation criteria have already been presented in the earlier sections. The following sections present the numerical discretization and Navier-Stokes solution procedure.

The incompressible mass and momentum equations are solved using a projection (Chorin, 1968; Kim & Moin, 1985) with staggered grid finite volume formulation. The temperature equation is solved using collocated grid method. The pressure, temperature and fluid properties are stored at the cell center and face-normal velocity is stored on Cartesian cell faces (Figure 4-5(a)). The flow computation follows the following sequence of steps:

Step 1: Solve for temperature field
The following temperature equation is solved using sharp interface method. The spatial discretization of the terms is presented later. The fluid properties and diffusion term are treated sharply. The convection and diffusion terms are discretized using 2nd order Runge-Kutta and fully implicit method, respectively.

\[
(pC)_k^{n+1} \frac{T_k^{n+1} - T_k^n}{\Delta t} = -\nabla \cdot \left( u_T \right)_k^{n+1/2} + \nabla \cdot \left( K \nabla T \right)_k^{n+1}
\]

Step 2: Velocity computation using projection method.
   - Predictor-step
     Solve the momentum for an intermediate velocity field \(\vec{U}^*\) using Equation(4.5) where all the know values such as surface tension source, gravitation, convection
and old time-step viscous term due to Crank-Nicholson method are lumped into \( S^n \). The temporal discretization of the convection term uses 2nd order Runge-Kutta method. The pressure term is approximated using the old time pressure field. Subsequently, remove the effect of pressure term by shifting the velocity field back to obtain another intermediate velocity field \( U^{**} \) using Equation(4.6).

\[
\left( \frac{\Delta V \rho}{\Delta t} - a_v \right) U^* = -\int \nabla P^n \cdot dA + f_{\text{visc}}^* + S^n 
\]

\[ U^{**} = U^* + \Delta t \frac{\nabla P^n}{\rho^{n+1}} \]  

- **Corrector-step**

Correct the predicted velocity field \( U^{**} \) using Equation(4.7). The pressure field for this correction is computed by enforcing velocity-divergence condition and solving the Poisson Equation(4.8) with conjugate gradient method. The divergence of the new velocity field \( U^{n+1} \) is zero if no phase change occurs else it is computed using Equation(4.9) to account for the interfacial mass transfer described in the previous chapter.

\[
U^{n+1} = U^{**} - \Delta t \frac{\nabla \rho^{n+1}}{\rho^{n+1}} \]  

\[
\sum_{\text{cell face}} \left( \frac{\nabla \rho^{n+1}}{\rho^{n+1}} \right) \cdot ndA = \frac{1}{\Delta t} \left( \sum_{\text{cell face}} U^{**} \cdot ndA - \int \nabla \cdot U^{n+1} dV \right) \]  

\[
\nabla \cdot U = \left( \frac{1}{\rho_1} - \frac{1}{\rho_2} \right) \dot{m} \cdot \nabla I \]  

### 4.3.1 Staggered Grid Arrangement and Approximations

A velocity constraint suggested by Losasso et al. (2004, 2005) is used to simplify the spatial discretization. With this approach, the velocities and pressure gradients on fine faces, \( \text{face1} \) and \( \text{face2} \) of a coarse cell shown in Figure 4-5(b) are constrained as \( U_1 = U_2 \) and \( P_{x1} = P_{x2} \) i.e., \( \text{face1} \) and \( \text{face2} \) have the same control volume. The face normal gradients on these fine faces are approximated using Equation(4.10) so that same amount of velocity correction is applied when using Equation(4.7). It was also shown by Losasso et al. (2004, 2005) that applying the same pressure gradients on finer faces can produce
second order spatial accuracy for the computed pressure field. Although assigning the same velocity to fine faces reduces the spatial accuracy near non-uniform cells, it is accepted as a compromise for algorithmic simplicity especially when non-uniform cells can be pushed away from the critical regions (e.g. interface) via suitable refinement criteria.

\[ P_{x1} = P_{x2} = \frac{(P_1 + P_2) - 2P_3}{2d} \]  

\[ (4.10) \]

Figure 4-5. Staggered grid arrangement. (a) Spatial location of flow/fluid properties; (b) single control volume used for fine faces of a coarse cell.

4.3.2 Spatial Discretization of Velocity Equation

The convection term of type \( \nabla \cdot (\phi U) \) is represented in finite volume form integrated over the control-volume faces as:

\[ \int_{cv} \nabla \cdot (\phi U) dV = \sum \phi U \cdot n dA = \sum \phi U_{cf} dA \]  

\[ (4.11) \]

The normal velocities \( U_{cf} \) are computed appropriately to maintain the divergence free condition within each control volume i.e., \( \sum U_{cf} dA = 0 \). Using Figure 4-6(a) and the velocity constraints, the control volume face-normal velocities are computed using face-area weighted averaging that produces:
The momentum flux term $\phi$ on control volume faces are computed using linear interpolation of face-normal velocities. To conserve the momentum flux across non-uniform grid cells, Figure 4-6(b) makes another approximation that computes the flux on top control-volume face for velocity $U_1$ and equally distributes to the corresponding control volumes of faces for $U_2$ and $U_3$ at a T-node junction. The flux term $\phi$ in Equation (4.11) is computed using 2nd or 3rd order essentially non-oscillatory scheme (ENO) where the grid is locally of uniform size. Since few layers of cells around interface are always at uniform resolution, 3rd order ENO scheme is employed there. For non-uniform cells (away from the interface) where uniform grid cells were not present to construct even a 2nd order ENO scheme, a simple central differencing was used for simplicity.

$$U_{cf1} = 0.5(U_1 + U_2)$$
$$U_{cf2} = 0.5(U_1 + U_3)$$
$$U_{cf3} = 0.5(U_1 + U_4)$$
$$U_{cf} = \frac{(U_5 Area_1 + U_6 Area_2)}{Area_1 + Area_2}$$  \hspace{1cm} (4.12)
The momentum equation, with the known source terms lumped into \((S_u,S_v,S_w)\), can be rewritten using Equation (4.13). The last term in Equation (4.13) comes from \(\nabla \cdot \mu \left( \nabla^T u \right)\) for incompressible flows. The discretization of \(\nabla \cdot \mu \left( \nabla u \right)\) requires computation of control-volume face-normal velocity gradients and the terms due to \(\nabla \cdot \mu \left( \nabla^T u \right)\) require computation of mixed derivatives. Discretization of \(\nabla \cdot \mu \left( \nabla u \right)\) follows the strategy outlined by Figure 4-6 while the terms due to \(\nabla \cdot \mu \left( \nabla^T u \right)\) are added only around the interface where the viscosity is spatially varying and it is easily discretized there since the grid around the interface is uniform. Owing to divergence free nature of the velocity and constant viscosity in most of the domain, the contribution due to terms of \(\nabla \cdot \mu \left( \nabla^T u \right)\) are neglected in cells away from the interface where the grid may be non-uniform.

\[
\begin{align*}
\rho \frac{Du}{Dt} &= -\frac{\partial P}{\partial x} + S_u + \nabla \cdot (\mu \nabla u) + \nabla \cdot \mu \frac{\partial (u,v,w)}{\partial x} \\
\rho \frac{Dv}{Dt} &= -\frac{\partial P}{\partial y} + S_v + \nabla \cdot (\mu \nabla v) + \nabla \cdot \mu \frac{\partial (u,v,w)}{\partial y} \\
\rho \frac{Dw}{Dt} &= -\frac{\partial P}{\partial z} + S_w + \nabla \cdot (\mu \nabla w) + \nabla \cdot \mu \frac{\partial (u,v,w)}{\partial z}
\end{align*}
\]

(4.13)

### 4.3.3 Spatial Discretization of Temperature Equation

Since the temperature is located at the cell-center, a collocated grid discretization with auxiliary variable creation for non-uniform cells is used (Ham et al., 2002; Singh & Shyy, 2006). Convection terms are discretized using 3\(^{rd}\) order ENO (essentially non-oscillatory) scheme around the interface. A 2\(^{nd}\) order ENO scheme is used for cells away from the interface if the required uniform grid stencil can be easily be found, else a
simple central differencing was used and was not found to produce any instabilities in the computed results.

4.3.3.1 Sharp interface treatment

The fluid properties \((\rho C, K)\) are used sharply based on the interface location. The interface location, for this purpose, is approximated by indicator = 0.5 contour as was done for interface reconstruction described in the previous chapter. The term \(\nabla \cdot (KT)\) is computed using central differencing for cells away from the interface. For interfacial cells, the discretization is based on the sharp interface treatment of Gibou et al. (2002) and Luo et al. (2005) using level-set method. The discretization in the interfacial cells can be explained using Figure 4-7 where the diffusion term for cell \(i\) is computed using the interface temperature as a boundary condition (Equation (4.14)).

\[
\frac{\partial}{\partial x} \left( K \frac{\partial T}{\partial x} \right) = \left( \frac{K_{i+1/2} T_{sat} - T_i}{\delta} \right) - \left( \frac{K_{i-1/2} T_i - T_{i+1}}{dx} \right)
\]

Figure 4-7. A one-dimensional interface between \(i\) and \(i+1\).

4.3.3.2 Interface temperature treatment

Coming back to the temperature equation presented in Chapter 3, the role of the source term \(Q_s\) in Equation (3.3) is essentially to maintain the interface at saturation temperature. The approach of Juric and Tryggvason (1998) treated this source term implicitly using a Newton iteration technique. The work of Son et al. (1999) and Son
(2001) eliminated this source term by fixing the temperature inside the interface at saturation temperature and solving only outside the interface. Later on, Morgan (2005) suggested another approach by linearly correcting the temperature within one cell on each side of interface. This linear correction approach is summarized as:

**Step 1.**

Solve temperature equation:

\[
\frac{\partial T}{\partial t} + \nabla \cdot \mathbf{u} T = \frac{1}{\rho C_T} \left( \nabla \cdot (K \nabla T) \right)
\]  

(4.15)

**Step 2.**

Correct temperature in cells immediately across the interface. Consider the interface lying between cell ‘\(i\)’ and ‘\(i+1\)’ in the following figure. After solving the temperature equation in step 1, the temperature at cell centers ‘\(i\)’ and ‘\(i+1\)’ are modified by linear interpolation of interface temperature (saturation temperature \(T_{\text{sat}}\)) and neighboring cell-center temperatures as shown in Figure 4-8.

![Figure 4-8. Linear interface temperature correction within one cell on each side of the interface.](image)

The present approach defines the interface at indicator = 0.5 and defines the correction as a solution of \( \nabla \cdot (K \nabla T) = 0 \). This approach does not require explicit knowledge of the minimum distance of a cell from the interface as required in the method of Morgan (2005) using level-set tracking. This correction technique is convenient with the immersed boundary method and can easily correct the temperature within any prescribed thickness across the interface. This equation is referred to as the interface
temperature correction equation and it is solved sharply within a thickness of 1.2 grid-cells on each side of the interface. Using Figure 4-9, this method will correct the temperature of $T_i$ and $T_{i+1}$ while leaving $T_{i-1}$ and $T_{i+2}$ untouched. The interface saturation temperature ($T_{sat}$) and temperatures of cells outside the correction zone are taken as Dirichlet boundary condition for the correction equation. Since $K$ is constant within each domain, this equation reduces to $\nabla \cdot (K \nabla T) = 0$ and it is enforced while solving the temperature equation.

![Figure 4-9](image)

Figure 4-9. Presently employed interface temperature correction technique solves a correction equation sharply on each side of the interface.

### 4.4 Summary

This chapter presented the adaptive Cartesian grid computation technique employed in the current work. The discretization of various terms in the flow equation and corresponding constraints on the grid were presented. The hydrodynamics was solved using immersed boundary method while the temperature equation was treated sharply for accurate interfacial mass-transfer computation.

Several improvements in the grid adaptation are possible. For example, a solution based adaptation requires some measure of local solution error and a more general
criterion suitable for both low and high Reynolds number flows based on the truncation or discretization error (Ham et al., 2002; Ferm & Lotstedt, 2003; Muzafjerija & Gosman, 1997) can be considered. Flow solution procedure and numerical discretization also need further research for in order to improve the spatial accuracy by eliminating restrictions placed on the velocity and pressure field.
The tests are conducted in three categories: basic tests to establish the working of adaptive grid flow solver and characterization of the interface handling techniques; bubble/drop dynamics tests; phase change computations. The conservative interface restructuring technique and reconstruction are tested for various aspects such as accuracy, mass conservation. A spherical surface in a time reversed vortex field is used to test the effectiveness of conservative restructuring. Interface reconstruction is tested for its accuracy in terms of mass-conservation and solution disturbances it causes. The spurious velocity currents and effect of reconstruction along with a set of multidimensional single and multiple rising bubbles and binary drop collision are computed to establish the accuracy of computational procedure. Computations of diffusion heat transfer between concentric cylinders, natural convection, one dimensional phase change and a stationary bubble growth are conducted to gauge the performance of the presently developed phase change computation procedure.

5.1 Interface Restructuring: Interface in a Vortex Field

A spherical interface placed in a time-reversed vortex field (Figure 5-1) deforms and should return back to the original spherical shape (Aulisa et al., 2004). The vortex field with time period $T = 4$ was imposed on the Cartesian grid using Equation (5.1) and
the markers were advected by interpolating this velocity field from the Cartesian grid.

\[
u(x, y, z) = \cos(\pi t / T) \sin^2(\pi x) \left( \sin(2\pi z) - \sin(2\pi y) \right) \\
v(x, y, z) = \cos(\pi t / T) \sin^2(\pi y) \left( \sin(2\pi x) - \sin(2\pi z) \right) \\
w(x, y, z) = \cos(\pi t / T) \sin^2(\pi z) \left( \sin(2\pi y) - \sin(2\pi x) \right)
\] (5.1)

During the simulation period, markers are continuously added and deleted from the interface to maintain the resolution. The interface at the end of one time period should return back to the spherical shape with some errors due to restructuring and marker advection using the interpolated velocity field from the Cartesian grid. The errors at the end of one time period were estimated for the final surface area, volume and radius of markers (distance from the expected sphere center) as shown in Equation (5.2). Table 5-1 presents the computed errors showing a better performance by the conservative restructuring as compared to the non-conservative restructuring presented in Chapter 3. Figure 5-1 shows the time history of the volume-error in the interface along with the grid convergence studies. The conservative method performs considerably well showing much smaller volume errors with better than quadratic convergence. Note that the volume errors with conservative restructuring are caused solely by the integration of the marker advection equation using the interpolated velocity field.

\[
\text{RMS Radius Error} = \sqrt{\frac{\sum_{i=1}^{N} (r_{i,\text{final}} - r_{i,\text{initial}})^2}{N}}
\]

\[
\text{Area Error} = 100 \left( 1 - \frac{A_{\text{final}}}{A_{\text{initial}}} \right)
\] (5.2)

\[
\text{Volume Error} = 100 \left( 1 - \frac{V_{\text{final}}}{V_{\text{initial}}} \right)
\]
Table 5-1. Error estimates using Equation (5.2) for interface in a time reversed vortex field test comparing conservative and non-conservative restructuring.

<table>
<thead>
<tr>
<th>Grid</th>
<th>RMS_Radius error (conservative, non-conservative)</th>
<th>% Area_error (conservative, non-conservative)</th>
<th>% Volume_error (conservative, non-conservative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30x30x30</td>
<td>(0.2435, 1.0254)x10^{-2}</td>
<td>(4.2898, -7.9391)</td>
<td>(-0.5591, -14.0950)</td>
</tr>
<tr>
<td>60x60x60</td>
<td>(0.0091, 0.4212)x10^{-2}</td>
<td>(1.5306, -2.4060)</td>
<td>(-0.1178, -4.9882)</td>
</tr>
<tr>
<td>120x120x120</td>
<td>(0.0079, 0.1155)x10^{-2}</td>
<td>(0.4169, -0.5978)</td>
<td>(-0.0216, -1.0455)</td>
</tr>
</tbody>
</table>

Figure 5-1. Interface in a time reversed vortex field. (a) Initial position of the interface with diameter 0.15 placed at (0.5, 0.75, 0.5) in a unit cube with its center at (0.5, 0.5, 0.5); computed shape history for vortex field in Equation (5.1) with period $T = 4$.

Figure 5-2. Error estimate and grid convergence for interface in time reversed vortex field. (a) Interface volume error on 30x30x30 grid; (b) final volume error on 30x30x30, 60x60x60 and 120x120x120 grids with dotted quadratic reference rate.
5.2 Interface Reconstruction: Effect on Mass Conservation

The interface marker spacing and accuracy of the indicator-function computation and hence the level-contour based reconstruction depends on the background grid resolution. The aim is to measure interface volume-error caused by reconstruction i.e., the difference between the original and reconstructed interface volumes. The tests were performed using a spherical interface of radius \( r = 0.2 \) at the center of a domain with dimensions \((0.4, 0.4, 0.4)\). The interface was reconstructed with different background grid resolution to confirm a quadratic convergence of the volume error (Figure 5-3) observed by Shin and Juric (2002).

The volume errors produced by reconstruction are corrected explicitly by perturbing the markers in local normal direction. It is important to assess the approximate amount of perturbation a reconstruction may produce. Denoting the initial surface area and volume of the interface as \( A \) and \( V \), an approximate radial perturbation to correct a volume defect \( \Delta V \) may be estimated using Equation (5.3). This shows a second order convergence behavior for the radial perturbation. To get further insight, this perturbation should be measured in terms of the grid resolution. If the diameter of the interface is resolved with \( N \) grid cells, the radial perturbation can be rewritten in terms of the grid cell size \( \Delta \) as shown in Equation (5.4). The last column of Table 5-2 shows the behavior of this quantity and reveals that this perturbation is approximately one tenth of the grid cell even for the coarsest grid used. This error too exhibits a quadratic convergence with grid refinement. Considering the fact that fluid properties are smeared over nearly four to five cells across interface making the immersed boundary method a first order accurate method, the amount of radial disturbance caused by even moderately resolved grids \((N > 20)\) is reasonably small.
\[ |Δr| \approx \frac{|ΔV|}{A} = \frac{|ΔV|}{4πr^2} = \frac{ΔV}{V} \frac{r}{3} \quad (5.3) \]

\[ |Δr| \approx \frac{|ΔV|}{V} \frac{r}{3} = \frac{|ΔV|}{V} \frac{NΔ}{6} \quad (5.4) \]

---

**Figure 5-3.** Grid convergence test for volume error due to reconstruction.

**Table 5-2.** Volume error for reconstruction of a spherical interface and approximate radial perturbation required for correction.

<table>
<thead>
<tr>
<th>Grid cell per-diameter (d/Δ)</th>
<th>% Volume error</th>
<th>Radial perturbation (Equation (5.4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>-7.5995</td>
<td>0.1266 Δ</td>
</tr>
<tr>
<td>20</td>
<td>-1.9887</td>
<td>0.0663 Δ</td>
</tr>
<tr>
<td>40</td>
<td>-0.4881</td>
<td>0.0325 Δ</td>
</tr>
<tr>
<td>80</td>
<td>-0.1233</td>
<td>0.0164 Δ</td>
</tr>
</tbody>
</table>

---

**5.3 Single Phase Navier-Stokes Solver**

A decaying vortex field and lid driven cavity tests are presented to establish the accuracy of the hydrodynamic solver. The temperature equation is tested for the discretization of the diffusion term using the diffusive heat transfer between concentric cylinders maintained at constant temperatures. The specified temperature on the cylinder geometries mimics the specified interface temperature and measures the error in the diffusion term discretization.
5.3.1 Decaying Vortex Field

A vortex field (Kim & Choi, 2000) shown in Equation (5.5) was introduced in a square domain of 1x1 with free-slip condition on all the walls. The center of the computational domain is located at the origin of the coordinate system. The computed decay of the velocity field at Reynolds number $Re = 10$ with (reference velocity and length scales taken as 1.0) was compared with the theoretical value at time $t = 0.3$. Figure 5-4(a) shows the streamlines.

$$u(x, y, t) = -\cos(\pi x)\sin(\pi y)e^{-2\pi^2 t/Re}$$
$$v(x, y, t) = \sin(\pi x)\cos(\pi y)e^{-2\pi^2 t/Re}$$

Figure 5-4(b) shows the streamlines.

The simulations were carried out on a series of adaptive Cartesian grids and compared with uniform-grid computation with resolutions corresponding to the maximum allowed resolution on the adaptive grid. The adaptive grids were generated by recursively refining a 16x16 base grid and the computations were started with prescribed maximum allowed refinement in the entire domain. The solution based adaptation was performed at every 20 time step with the computational time step set at 0.001. An adapted grid with two refinement levels at time $t = 0.3$ is shown in Figure 5-4(b).

The maximum error in numerically computed u-component of the velocity field and the corresponding computation-time is presented in Table 5-3 with the time normalized by the time taken for 16x16 uniform-grid computation. As seen in Figure 5-5, uniform grid computations show quadratic convergence but adaptive grid shows an average convergence rate of $\sim 1.5$. Since it was observed that the maximum error occurs along in the boundary cells which have non-uniform neighbor cells (Figure 5-4), the computed error is a reasonable reflection of the accuracy of spatial discretization over
non-uniform Cartesian grids. Although the accuracy from uniform grid has been observed to be superior, it is the computational cost that gives adaptive grid computations an edge. Since most of the cells shown in adapted grids are of uniform sizes, these computations may not offer the most attractive picture of computation-time savings that are expected to be more pronounced for 3D computations.

![Figure 5-4. Decaying vortex field. (a) Streamlines; (b) two refinement levels on a 16x16 base grid.](image)

![Figure 5-5. Grid convergence test for decaying vortex field showing maximum velocity error computed on uniform and adaptive grids.](image)
Table 5-3. Max velocity error and computation time normalized with time taken for 16x16 uniform grid. The adapted grids are shown in Figure 5-4.

<table>
<thead>
<tr>
<th>Max. grid resolution</th>
<th>Uniform grid computation</th>
<th>Adaptive grid base computation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max. U error</td>
<td>Normalized time</td>
</tr>
<tr>
<td>16x16</td>
<td>2.966x10^-3</td>
<td>1.000</td>
</tr>
<tr>
<td>32x32</td>
<td>7.184x10^-4</td>
<td>4.440</td>
</tr>
<tr>
<td>64x64</td>
<td>1.752x10^-4</td>
<td>21.602</td>
</tr>
<tr>
<td>128x128</td>
<td>4.344x10^-5</td>
<td>104.810</td>
</tr>
</tbody>
</table>

5.3.2 Lid-driven Cavity

A lid driven cavity flow with $Re = 100$ was computed where the effect of grid adaptation is more visible than the decaying vortex field in previous section. Figure 5-6 shows the streamlines of the flow computed on a 120x120 grid. Since the adaptation criterion is based on curl of the velocity field, computed vorticity contours are presented in Figure 5-7(a) and they are in accordance with the computations of Ghia et al. (1982).

![Figure 5-6. Streamlines for lid-driven cavity simulation for $Re = 100$](image)

The computations were carried out on three grids with maximum resolution of 30x30, 60x60 and 120x120 obtained by one, two and three refinement levels as shown in Figure 5-7. Figure 5-8 shows the u-velocity profile along a vertical line at the center of the domain for the uniform and adaptive grid computations showing no significant difference among the computed results on adaptive and uniform grids. The exact values of minimum u-velocity in Figure 5-8 are presented in Table 5-4. This table also provides
a comparison of the total number of grid cells and computational time for uniform and adaptive grids. The second-last column of this table shows ratio of the total number of cells from adaptive and uniform grid computations. The last column is the ratio of computational time between adaptive and uniform grids indicating up to \(\sim 70\%\) computational time savings on large grids.

Table 5-4. Lid driven cavity computation using uniform and adaptive grids. Comparison of the number of cells, computation time and minimum u-velocity on the vertical line through center.

| Max. grid resolution | Uniform grid | | Adaptive grid | |
|----------------------|--------------|------------------|---------------|------------------|-----------------|-----------------|-----------------|------------------|-----------------|-----------------|
|                      | #Cells | Time | Min u | #Cells | Time | Min u | Cell ratio | Time ratio | |
| 30x30                | 900    | 1.000 | -0.203 | 447    | 0.587 | -0.204 | 0.497 | 0.587 | |
| 60x60                | 3600   | 2.933 | -0.211 | 1263   | 1.157 | -0.210 | 0.351 | 0.394 | |
| 120x120              | 14400  | 10.910 | -0.212 | 4407   | 3.264 | -0.212 | 0.306 | 0.299 | |

Figure 5-7. Lid driven cavity. (a) Vorticity contour and final adaptive grids with maximum refinement level of (b) one, (c) two and (d) three.
Figure 5-8. The \( u \) velocity profile along a vertical line through the center of lid-driven cavity using (a) uniform grids and (b) adaptive grids.

5.3.3 Diffusive Heat Transfer between Concentric Cylinders

The diffusion Equation (5.6) is solved for steady state solution with Dirichlet boundary condition of constant temperature on the cylinder walls. The density, heat capacity and conductivity are set to 1.0. The inner and outer surfaces of the cylinder are maintained at temperature \( T_i = 2.0 \) and \( T_o = 1.0 \), respectively. With the inner and outer radii denoted by \( R_i \) and \( R_o \), the theoretical steady state temperature distribution \( T(R) \) in circular coordinate system is shown in Equation (5.7).

The computations were performed on a 2D Cartesian grid. The geometry of concentric cylinders (concentric circles in 2D) was modeled using an indicator function as shown in Figure 5-9. The computations are performed where indicator >= 0.5 (space between the concentric circles). Figure 5-10 shows the computed normalized temperature distribution and grid convergence studies showing convergence rate between linear and quadratic as expected based on the discretization of the diffusion terms near the boundary.

\[
\rho C \frac{\partial T}{\partial t} = \nabla \cdot (K \nabla T) \tag{5.6}
\]
\[
\frac{T(R) - T_i}{T_o - T_i} = \frac{\log(R/R_i)}{\log(R_o/R_i)}
\]

(5.7)

Figure 5-9. Indicator contour used for modeling the cylindrical computational domain. (a) Computations are performed only within the domain with Indicator \( \geq 0.5 \); (b) cross-section of (a) showing outer and inner radii as \( R_o = 0.3 \) and \( R_i = 0.1 \), respectively.

Figure 5-10. Computed temperature and error for heat transfer in a concentric cylinder. (a) temperature distribution \((T-T_i)/(T_o-T_i)\); (b) grid convergence study with resolution 30 (30x30 grid), 60 (60x60 grid) and 120 (120x120 grid) showing \( L_2 \) and \( L_{\infty} \) error norms with reference quadratic and linear lines.
5.3.4 Natural Convection in a Square Cavity

The temperature equation solver was tested for a natural convection case with Prandtl number \((Pr) = 0.71\) and Rayleigh number \((Ra) = 1.0 \times 10^5\) using Boussinesq approximation. The computations were performed in a 2D unit square \((L = 1)\) with top left and right walls at hot and cold temperatures (difference = \(\Delta T\)), respectively. The top and bottom walls were assigned adiabatic boundary conditions. Results in Table 5-5 show good agreement with the computations by Davis and Jones (1983) for the minimum and maximum Nusselt numbers on the left wall \((Nu_{x=0} = -\partial T/\partial x\) with temperature and lengths non-dimensionalized with \(\Delta T\) and \(L)\) and \(v_{\text{max}}\) as the maximum vertical velocity on a horizontal line through the center of the domain. Figure 5-11 shows the computed streamlines and temperature isotherms.

![Streamlines and temperature isotherms](image)

Figure 5-11. Natural convection with \(Pr = 0.71\) and \(Ra = 1.0 \times 10^5\). (a) Streamlines and (b) the temperature isotherms.

<table>
<thead>
<tr>
<th>Measured parameter</th>
<th>Grid resolution</th>
<th>Davis and Jones (1983)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_{\text{max}})</td>
<td>60x60</td>
<td>68.29</td>
</tr>
<tr>
<td>(Nu_{x=0, \text{min.}})</td>
<td>120x120</td>
<td>0.721</td>
</tr>
<tr>
<td>(Nu_{x=0, \text{max.}})</td>
<td>7.952</td>
<td>7.781</td>
</tr>
</tbody>
</table>
5.4 Spurious Velocity Currents

Spurious velocities or parasitic currents (Figure 5-12(a)) are unphysical velocity fields created due to numerical errors causing imbalance of interfacial stresses. These currents are best seen for static bubble simulations where the theoretical velocity is identically zero everywhere and the pressure drop across interface balances the surface tension forces dictated by the Young-Laplace Equation (5.8) (Figure 5-12(b)). There are several reasons ranging from the accuracy of surface tension computation to errors in computation of pressure and grid anisotropy effects. A further detailed insight into the origins and nature of these currents can be found in the work of Torres and Brackbill (2000) and Harvie et al. (2005). Since the magnitude of these currents for continuous interface methods are found to be directly proportional to the surface tension and inversely proportional to the viscosity (Lafaurie, 1994), they have the potential to destabilize a computation and restrict the accuracy and the range of computational flow-parameters (fluid properties, bubble sizes) that can be successfully simulated.

The expected magnitude of non-dimensional spurious velocities (Capillary number) based on the observations of Shin and Juric (2002) and Sousa et al. (2004) is of the order of $10^{-4}$ and it has been hypothesized by Lafaurie (1994) that this magnitude remains largely unaffected by the Laplace number. To offer a perspective, the volume of fluid computations of Lafaurie (1994) observed spurious current of the order of $10^{-2}$ that limited their simulations to Laplace number (La) $< 10$ while the current code was tested up to Laplace number 12000 and found to be stable. The properties of the fluid outside the interface and bubble diameter were taken as relevant reference scales.

A spherical bubble of diameter $d = 0.2$ was placed in a cubic domain of size $(3d, 3d, 3d)$ with slip-wall boundary condition. The surface tension $\sigma$ was set to 0.1 producing
a theoretical pressure jump of 2.0. The computations were performed on \(25 \times 25 \times 25\), \(50 \times 50 \times 50\) and a \(100 \times 100 \times 100\) grid. All the time axes in the plots for the spurious velocity computations are non-dimensionalized with the capillary time scale of \(d \mu / \sigma\).

\[
\Delta p = \sigma \kappa
\]  

(5.8)

Figure 5-12. Static bubble computation. (a) Spurious currents of magnitude \(Ca \approx 10^{-4}\), (b) computed pressure for theoretical jump of 2.0.

5.4.1 Effect of Fluid Property Jump and Grid Resolution

A Laplace number = 250 case was performed on \(25 \times 25 \times 25\) and \(50 \times 50 \times 50\) grids. The magnitude of spurious current reduced by an order of magnitude on finer grid as shown in Figure 5-13(a) but was found to exhibit no concrete improvements on finer grids. Since the pressure drop across the interface is captured over a few cells, an error in computed pressure drop was measured with an error norm in Equation\( (5.10)\) used by Brackbill et al. (1992) where \(N_{in}\) is the number of cells inside the interface. The pressure drop \(\Delta p_i\) in Equation (5.10) is the difference in the pressure computed at the cell center and average pressure outside the interface (\(\bar{p}_{out}\)) as shown in Equation (5.9) for computing the average pressure drop \(\Delta p_{num}\). The observed convergence rate of ~0.3 by Brackbill et al. (1992) is supported by the computed data in Table 5-6 and Figure 5-13(b).
The simulated Laplace numbers in the range of 250 to 12000 were found to have no appreciable effect on the Capillary number as shown in Table 5-7. High density and viscosity ratios (1000 and 100 respectively) were also tested and found not to cause very serious degradation as shown Table 5-8 and Table 5-9.

\[ \Delta p_{num} = \bar{p}_{in} - \bar{p}_{out} = \frac{1}{N_{in}} \sum_{i=1}^{N_{in}} P_i - \frac{1}{N_{out}} \sum_{j=1}^{N_{out}} P_j \]  \hspace{1cm} (5.9)

\[ \text{Pressure drop error} = \sqrt{\frac{1}{N_{in}} \sum_{i=1}^{N_{in}} \left( \Delta p_i - \Delta p_{exact} \right)^2 / \Delta p_{exact}} \]  \hspace{1cm} (5.10)

![Figure 5-13. Grid convergence tests for static bubble. (a) Capillary number evolution, (b) error in pressure drop (Equation(5.10)) using data in Table 5-6.](image)

Table 5-6. Pressure drop for an inviscid fluids with density ratio =1.0. Data was taken after three non-dimensional time step of 5.0x10^{-3}.

<table>
<thead>
<tr>
<th>Grid resolution</th>
<th>Pressure drop error (Equation (5.10))</th>
</tr>
</thead>
<tbody>
<tr>
<td>25x25x25</td>
<td>0.197</td>
</tr>
<tr>
<td>50x50x50</td>
<td>0.151</td>
</tr>
<tr>
<td>100x100x100</td>
<td>0.117</td>
</tr>
</tbody>
</table>

Table 5-7. Effect of Laplace number on spurious velocity.

<table>
<thead>
<tr>
<th>Laplace number (La = \sigma pd / \mu^2 )</th>
<th>Capillary number (Ca = U_{max} H / \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>5.32x10^{-4}</td>
</tr>
<tr>
<td>12000</td>
<td>5.35x10^{-4}</td>
</tr>
</tbody>
</table>
Table 5-8. Effect of density ratio on pressure drop and spurious current on 50x50x50 grid (viscosity ratio = 1.0, La = 250)

<table>
<thead>
<tr>
<th>Density ratio = ( \rho_1/\rho_2 )</th>
<th>Pressure drop = ( \frac{\Delta p_{\text{sum}}}{\Delta p_{\text{exact}}} )</th>
<th>Capillary number = ( \frac{U_{\text{max}} \mu}{\sigma} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.90</td>
<td>5.32x10^{-4}</td>
</tr>
<tr>
<td>10</td>
<td>0.89</td>
<td>5.27x10^{-4}</td>
</tr>
<tr>
<td>100</td>
<td>0.87</td>
<td>5.15x10^{-4}</td>
</tr>
<tr>
<td>1000</td>
<td>0.87</td>
<td>6.35x10^{-4}</td>
</tr>
</tbody>
</table>

Table 5-9. Effect of viscosity ratio on pressure drop and spurious current on 50x50x50 grid (density ratio = 1000, La = 250)

<table>
<thead>
<tr>
<th>Viscosity ratio = ( \mu_1/\mu_2 )</th>
<th>Pressure drop = ( \frac{\Delta p_{\text{sum}}}{\Delta p_{\text{exact}}} )</th>
<th>Capillary number = ( \frac{U_{\text{max}} \mu}{\sigma} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.87</td>
<td>6.52x10^{-4}</td>
</tr>
<tr>
<td>100</td>
<td>0.86</td>
<td>6.61x10^{-4}</td>
</tr>
</tbody>
</table>

5.4.2 Effect of Interface Reconstruction

The current level-contour based interface reconstruction is expected to have the same characteristics as shown in the tests of Shin and Juric (2002). The effect of reconstruction on mass conservation showing quadratic convergence with grid refinement has already been established. Here the impact of reconstruction on spurious velocities is assessed.

The tests were performed for Laplace number 2500 on a 50x50x50 grid. Temporary spikes in spurious velocities were observed with every reconstruction (Figure 5-14(a)) as also seen by Shin and Juric (2002). These spikes tend die down in time unless the reconstruction is performed too frequently depending on the time scale of the problem. Figure 5-14(a) shows that if reconstruction is performed every 10 Capillary time scale, the temporary increase in spurious velocity tends to die down to the profile exhibited if no reconstruction were performed. However, if the reconstruction is performed at every Capillary time scale, the spurious currents are an order of magnitude larger and continue to grow. Further observation can be made from Figure 5-14 (b) which shows the interface...
normal velocity at a selected marker point. It shows that the oscillatory behavior caused by the surface tension effects quickly die down if no reconstruction is performed during the simulation. With reconstruction every 10 Capillary time scale, the amplitude of velocity does not decay but remains stable with mean value of nearly zero. This is to be expected based on the Laplace number of 2500 representing viscous time scales 2500 times slower than the Capillary time scales. But the reconstruction every Capillary time scale clearly shows a completely different mean value that is several orders of magnitude higher. These observations further strengthen the belief that reconstruction should be performed as sparingly as possible.

![Graphs showing capillary number and velocity evolution](image)

Figure 5-14. Effect of reconstruction frequency on spurious velocities. (a) Capillary number evolution, (b) normal velocity at a selected marker point.

### 5.5 Buoyancy Driven Single Rising Bubble

A set of rising bubbles are presented next to establish the accuracy of the present multidimensional immersed boundary procedure and implementation for handling complex interfacial flows.
5.5.1 Grid Convergence Test

A single bubble of diameter $d$ is placed in a computational domain of size and orientation indicated in Figure 5-15(a). The bubble rises in the positive Y-coordinate direction under the influence of gravity. The non-dimensional parameters used in the simulations are Reynolds number ($Re$), Morton number ($M$), Eötvös number ($Eo$), density and viscosity ratios. The properties of fluid outside the interface and bubble diameter are taken as the required scales.

Figure 5-15. Computational set-up for a single rising bubble. (a) Computational domain, (b) initial adaptive grid with 3 levels of refinement over 7x21x7 base grid.

Simulations of $Eo = 1.0$ and $M = 1.0 \times 10^{-3}$ representing a high surface tension and low Reynolds number case were conducted. To test the grid convergence, simulations were conducted on two grids with three levels of refinement over a 7x21x7 base grid and a 10x30x10 base grid. The 7x21x7 base grid produces maximum resolution equivalent to
nearly 19 cells per bubble-diameter and the 10x30x10 base grid produces maximum resolution of nearly 27 cells per diameter. The density and viscosity ratios are both set to 20.0. The aspect ratio is computed as shown in Figure 5-16 and the average bubble rise velocity is computed using Equation (5.11) (Esmaeeli & Tryggvason, 1998). With these parameters, the bubble is expected to remain almost spherical with an estimated rise Reynolds number of 1.6 using the shape diagram of Clift et al. (1978) (Esmaeeli & Tryggvason, 1998). The computed results shown in Figure 5-17 and Table 5-10 confirm a spherical terminal shape for the rising bubble. The aspect ratio of the bubble is large than 0.99 for the entire duration of the computation. Similar computations by Sousa et al. (2004) using uniform Cartesian grids of size 16x32x16 and 32x64x32 for a domain of size 2d x 4d x 2d produced an average rise Reynolds number of 1.4.

\[ \bar{v}_c = \frac{1}{Vol} \int_{\text{interface}} \bar{x} (\bar{v} \cdot \hat{n}) ds \]  \hspace{1cm} (5.11)

Table 5-10. Grid convergence test for \( E_o = 1.0, M = 1.0 \times 10^{-3} \). The density and viscosity ratios are set to 20.0.

<table>
<thead>
<tr>
<th>Maximum adaptive grid resolution</th>
<th>Rise Reynolds number</th>
<th>Aspect ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>10x30x10</td>
<td>1.53</td>
<td>0.992</td>
</tr>
<tr>
<td>56x168x56</td>
<td>1.51</td>
<td>0.996</td>
</tr>
</tbody>
</table>

\[
\text{Aspect ratio} = \frac{b}{a}
\]

Figure 5-16. Aspect ratio computation
Figure 5-17. Bubble rise with $Eo = 1.0$, $M = 1.0 \times 10^{-3}$, density and viscosity ratio of 20.
(a) streamlines, (b) aspect ratio, (c) rise Reynolds number with time ($t \sqrt{g/d}$).

5.5.2 Rising Bubbles in Different Terminal Shape Regimes

A large body of experimental data on shapes and rise velocities was condensed by Clift et al. (1978) in the form of a single diagram using Eötvos number, Morton number and Reynolds number as the non-dimensional parameters. This diagram has been used for validation by several researchers (Haario et al., 2004; Annaland et al., 2005) and serves as a general guideline for assessing the shape and rise velocities.
A series of Morton number ($M$) and Eötvos number ($Eo$) cases from Table 5-11 were computed and compared with the expected terminal bubble-shape and rise velocity based on the shape diagram of Clift et al. (1978) along with comparison with the Volume of fluid computations of Annaland et al. (2005). The side walls and bottom surface of the domain (Figure 5-18(a)) were assigned free-slip boundary condition and outlet condition (specified pressure and zero face normal velocity gradients) was used on the top surface. The initial computational grid was created in a domain of size = $(5d, 10d, 5d)$ with three levels of refinement over $12\times24\times12$ base grid (Figure 5-18(b)). The computed shapes shown in Figure 5-19 are in good agreement with the expected shape from Table 5-11 and the computed rise Reynolds numbers agree well with computations of Annaland et al. (2005). The skirted bubble surface and streamlines showing several vortices are presented in Figure 5-20 (b) and compared with the computation of Haario et al. (2004) for $M=1.3$ and $Eo=125$.

![Figure 5-18](image.png)

**Figure 5-18.** Computational domain for cases in Table 5-11. (a) Computational domain, (b) A $12\times24\times12$ base grid with refinement levels.
Figure 5-19. Computed shapes from Table 5-11. (a) Spherical, (b) ellipsoidal, (c) dimpled ellipsoidal and (d) skirted regimes; (e) shape diagram of Clift et al. (1978) with the corresponding locations of computed shapes indicated.

Figure 5-20. Computed case for $M = 0.971$, $Eo = 97.1$. (a) Interface and its cross-section, (b) streamlines, (c) streamlines for $M=1.3$, $Eo=125$ by Haario et al. (2005).
Table 5-11. Computation of rising bubbles with different terminal shape regimes.

<table>
<thead>
<tr>
<th>Computational parameter</th>
<th>Terminal Shape (Clift et al., 1978)</th>
<th>Rise Reynolds number</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>$Eo$</td>
<td>Clift et al. (1978)</td>
</tr>
<tr>
<td>1.26x10^{-3}</td>
<td>0.971</td>
<td>1.7</td>
</tr>
<tr>
<td>1.0x10^{-1}</td>
<td>9.71</td>
<td>4.6</td>
</tr>
<tr>
<td>1.0x10^{-3}</td>
<td>97.1</td>
<td>1.5</td>
</tr>
<tr>
<td>9.71x10^{-1}</td>
<td>97.1</td>
<td>20</td>
</tr>
</tbody>
</table>

5.6 Coalescence of Two Rising Bubbles

Drop collision simulations involving interface reconstruction to merge two bubbles have been conducted and compared with the volume of fluid based computations of Annaland et al. (2005). The computational parameters of $M = 2x10^{-4}$, $Eo = 16$, density and viscosity ratio = 100 are used for both the simulations. Bubble diameter and property of fluid outside the interface are used as relevant scale to define non-dimensional parameters.

5.6.1 Head-on Coalescence

The computational domain and initial grid shown in Figure 5-21 has dimensions $(4d, 8d, 4d)$ discretized by three levels of refinement over $10x20x10$ base grid resulting in a maximum resolution of $80x160x80$ equivalent to twenty grid cells per bubble diameter. The initial centers of the top and bottom bubbles are at $(2d, 2.5d, 2d)$ and $(2d, d, 2d)$ with the origin located at the bottom left corner of the domain. For comparison of the time history of the bubble shape with literature, the dimensional parameters used in the current simulation and that of Annaland et al. (2005), are presented in the following table.

The trailing bubble at the bottom rises faster in the wake of the leading bubble resulting in an eventual collapse. The time history of the development is presented in Figure 5-22. It shows the leading bubble acquiring a spherical cap shape (at $t \sim 0.75s$) which is consistent with respect to the shape diagram for the chosen computational
parameters. The trailing bubble has a significant influence of the leading bubble and
acquires a different shape. The computed results are in good agreement with the
simulations of Annaland et al. (2005) and follow the developments indicated by the
experiments of Brereton and Korotney (1991) shown in Figure 5-25(a). Figure 5-23 show
the computational grid and streamlines at 0.125s containing ~125K grid cells. The
interface reconstruction was performed at time $t = 0.116s$ (Figure 5-24) producing nearly
1.4% interface-volume error that was explicitly corrected. The computational time was
142 hours on an Intel Xeon 1.8 GHz processor with time step $(t\sqrt{g/d}) = 1.0 \times 10^{-4}$.

Table 5-12. Computational parameter for rising bubble-coalescence tests.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble diameter</td>
<td>0.01 m</td>
</tr>
<tr>
<td>Computational domain</td>
<td>(0.04, 0.08, 0.04)m</td>
</tr>
<tr>
<td>Grid resolution</td>
<td>80x160x80</td>
</tr>
</tbody>
</table>
| Bubble fluid property              | Density ($\rho_2$) = 10 kg/m³  
                                  | Viscosity ($\mu_2$) = 0.001 kg/ms   |
| Outer fluid property               | Density ($\rho_1$) = 1000 kg/m³  
                                  | Viscosity ($\mu_1$) = 0.1 kg/ms     |
| Surface tension ($\sigma$)         | 0.1 N/m             |

Figure 5-21. Computational set-up for head-on coalescence test. (a) Domain and (b) a
cross-section of the initial grid generated by three levels of refinement over a
10x20x10 base grid.
Figure 5-22. Head-on coalescence. (a) Computed time history of the two rising bubbles and (b) computations of Annaland et al. (2005). [Reprinted with permission]

Figure 5-23. Grid and streamlines at $t = 0.125$ sec for head-on coalescence. (a) Interface shape and a cross-section view of the grid, (b) streamlines with respect to the topmost point on the bubble.
Figure 5-24. Instance of interface reconstruction during head-on coalescence of two rising bubbles at $t = 0.116$ s. (a) Cross-section of two interfaces before reconstruction and (b) after reconstruction.

Figure 5-25. Experimental photographs of bubble coalescence taken at 0.03s time intervals. (a) A head-on coalescence and (b) an off-axis coalescence. (Taken from Brereton and Korotney, 1991)

5.6.2 Off-axis Coalescence

All the computational parameters used for off-axis collision are the same as for head-on collision except the initial bubble centers that are located at coordinates (2.8d, d, 2d) and (2d, 2.5d, 2d). The computed time histories are presented in Figure 5-26(a). The computational results are in good agreement with the volume of fluid based computations of Annaland et al. (2005) (Figure 5-26(a)) and exhibit the behavior similar to the experimental photographs by Brereton and Korotney (1991) shown in Figure 5-25(b).
Figure 5-26. Off-axis coalescence. (a) Computed, (b) computations of Annaland et al. (2005). [Reprinted with permission]

5.7 Binary Drop Collision

Bubble coalescence tests for two rising bubbles demonstrated the potency of the developed algorithms for handling complex interface shapes and topology changes. These cases however were relatively benign in terms of the dynamics of collision/coalescence along with relatively lower density ratios of 100. Next we consider collision of equal sized tetradecane (C\textsubscript{14}H\textsubscript{30}) drops in nitrogen and evaluate the
performance by comparing with the experiments of Qian and Law (1997) for the dynamics of collision and volumes of the satellite droplets from the experimental observations of Estrade et al. (1999).

A typical binary drop collision process results in either bouncing or coalescence that may be followed by separation that may result in one or more satellite droplets. Figure 5-27 shows various collision regimes depending on the Weber number and impact parameter that distinguishes between off-centre and head-on collisions. Some of the experimental works on drop collisions can be found in Ashgriz and Poo (1990), Jiang et al. (1992), Qian and Law (1997), Estrade et al. (1999), Brenn et al. (2001) and Willis and Orme (2003). In comparison, full three-dimensional numerical simulations are relatively sparse and some of the known computations are by: Rieber and Frohn (1997) using volume of fluid; Pan and Suga (2005) and Tanguy and Berlemont (2005) using level set; Nobari and Tryggvason (1996) and Shin and Juric (2002) using immersed boundary method; Mashayek et al. (2003) using finite element method. Besides computations with modest density ratios (20–30) by Nobari and Tryggvason (1996) and Shin and Juric (2002) no other 3D immersed boundary computation could be found in literature.

The non-dimensional parameters used for defining the collision of two equal size drops (diameter = d) are impact parameter \( B (= h/d \) from Figure 5-28), Weber number \( (We) \) and Reynolds number \( (Re) \). The density and viscosity ratios are used to define the properties of the surrounding medium. The parameter \( h \) in Figure 5-28 is the distance between the velocity vectors of the two drops and \( U_{\text{impact}} \) is the impact velocity. The impact parameter defines the eccentricity of collision with \( B = 0 \) representing a head-on
collision. All the non-dimensional parameters are computed with the fluid properties, drop diameter and the impact velocity as the relevant scales.

Due to excessive computational time and grid resolution required for attempting bouncing regimes at low Weber numbers, only regimes 3, 4 and 5 in Figure 5-27 are attempted. The fluid properties shown in Table 5-13 are taken from Pan and Suga (2005). The chosen fluid properties produce density and viscosity ratios of 666.081 and 179.28, respectively. The parameters for the computed cases are shown in Table 5-14.

![Figure 5-27. Binary drop collision regimes marking the location of computed parameters with solid circles. [Adapted from Qian and Law (1997)]](image)

![Figure 5-28. Impact parameter $B$ is defined as $B=\frac{h}{d}$.](image)

### Table 5-13. Properties of tetradecane and nitrogen.

<table>
<thead>
<tr>
<th>Medium</th>
<th>Density (Kg/m$^3$)</th>
<th>Viscosity (Kg/ms)</th>
<th>Surface tension (Kg/s$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tetradecane (C$<em>{14}$H$</em>{30}$)</td>
<td>758.0</td>
<td>2.128x10$^{-3}$</td>
<td>0.026 (in nitrogen)</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>1.138</td>
<td>1.187x10$^{-5}$</td>
<td></td>
</tr>
</tbody>
</table>

Note: Adapted from Pan and Suga (2005)
Table 5-14. Binary drop collision computation with density ratio = 666.081 and viscosity ratio = 179.28.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>$We$, $Re$, $B$</th>
<th>Expected outcome from Qian and Law (1997)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32.8, 210.8, 0.08</td>
<td>Coalescence</td>
</tr>
<tr>
<td>2</td>
<td>37.2, 228.0, 0.01</td>
<td>Near head-on separation without satellite</td>
</tr>
<tr>
<td>3</td>
<td>61.4, 296.5, 0.06</td>
<td>Near head-on separation with satellite</td>
</tr>
<tr>
<td>4</td>
<td>60.1, 302.8, 0.55</td>
<td>Off-axis collision with satellites</td>
</tr>
</tbody>
</table>

The cases 1, 2 and 3 in Table 5-14 were performed on a $7.5d \times 7.5d \times 7.5d$ domain with four levels of refinement over $11\times11\times11$ base grid resulting in approximately 24 cells per diameter. Case 4 was performed on a $9d \times 4.5d \times 4.5d$ domain with four levels of refinement over $16\times8\times8$ base grids resulting in approximately 29 cells per diameter. The collisions take place in x-y plane with different impact parameter and impact velocity producing the desired non-dimensional numbers. The initial horizontal separation between the drop-centers in x-y plane is approximated from the figures of Qian and Law (1997) and it is set to one diameter (surfaces touching) for Case 1 and 1.25 diameters for the other cases. All the figures showing liquid drop shapes, cross-section of the computational grid and velocity vectors are in x-y plane through the center of the initial drops. The initial velocity conditions are obtained by imposing the impact velocity field in the cells which have indicator function less than 1.0. The imposed velocity field is projected in a divergence-free space and used as the initial velocity condition. All the computational domains are assigned outlet condition imposing zero normal velocity gradient and constant atmospheric pressure on the boundaries.

Outcome of the collision is strongly dependent on the interaction of the kinetic energy of collision competing with the surface energy due to tension forces. The eccentricity of collision is an important factor in the overall collision outcome due to introduction of rotation energy in the coalesced drop resulting in production of
centrifugal forces responsible for a stretching type separation. At Weber numbers below a critical value of 34 estimated in the work of Qian and Law (1997) the collision is expected to result in a permanent coalescence as the surface tension forces have a dominating influence. Figure 5-29 has Weber number = 32.8 (Case 1) demonstrating permanent coalescence due to high surface tension forces pulling the long cylindrical surface (t = ~ 1.22 ms) back towards a spherical shape consistent with the experimental observations. Figure 5-30 shows the computational grid at various time instances along with time history of the grid data size exhibiting the relationship between the grid data size and instantaneous drop shapes due to solution and geometry based adaptation.

Case 2 was computed with the Weber number of 37.2 with a near head-on collision (B = 0.01). The surface forces are still strong enough to create a dimpled disc shape converting the entire kinetic energy into surface energy. High curvature at the rim of the disc creates a high pressure zone as compared to the almost flat disc-center (t = 0.33ms in Figure 5-31 and Figure 5-32) causing contraction that eventually moves towards producing a cylindrical shape. However, the axial velocities shown in Figure 5-32 at t = 1.26ms and 1.71ms are strong enough to overcome the surface energy and continue the outward motion of the blobs at cylinder ends resulting in eventual breakup into two equal size droplets. Figure 5-33 shows a snapshot of the grid at several time instances and the corresponding time evolution of the grid data size.

Further increase in the Weber number (61.4 in Case 3) produces stronger kinetic energy as compared to the surface forces and the resulting dynamics is shown in Figure 5-34. The disc creation following the initial coalescence is similar to the earlier case with We = 37.2. However, an impact parameter of 0.06 was seen to produces a large rotation
of nearly 45 degrees seen in Figure 5-34 and Figure 5-35. Most of the rotation takes place around the instance of disc formation that subsequently begins elongation into a cylindrical shape taking place approximately between $t = 0.23\text{ms}$ to $0.87\text{ms}$. No significant rotation is observed during the separation process leading to formation of a satellite. Figure 5-36 shows the computed velocity profile in x-y plane across the bubbles along with the pressure distribution to highlight the collision mechanism. The separation and satellite formation occurs via elongation of the cylinder creating bulbous shape owing to surface tension forces. The separation creates two primary droplets of equal size and a satellite with an equivalent diameter that is 25.8% of the initial diameter of the individual drops. The computed satellite drop size is in good agreement with the experimental observation of Estrade et al. (1999) reporting a value of 26% for collision of ethanol droplets that have very similar fluid properties as tetradecane. Similar computations by Pan and Suga (2005) produced a visually larger satellite drop although no quantitative information was made available.

The magnitude and direction of the rotation computed for Case 3 seem to be visibly different than the experimental observations of Qian and Law (1997). Also experimental figures seem to exhibit most of the rotation after formation of the cylindrical shape leading to separation. The reasons for this apparent difference is still unknown and requires further investigation regarding the orientation of the collision plane and photographs in the experimental figures. This is besides the apparent differences in the symmetry between computation and experiment seen in Figure 5-34 around $t = 0.18\text{ms}$ and $0.23\text{ms}$ for the impact parameter of 0.06.
Case 4 is an off-center collision with high impact parameter of 0.55 shown in Figure 5-37. As shown in the computation as well as the experiment, most of the rotation takes place with the coalesced mass before breakup process due to stretching begins. The experiment shows creation of two satellite droplets at $t \approx 1.92\text{ms}$ that quickly merge into a single satellite. However, the computed results show a single satellite droplet. A possible cause may be the relatively coarse computational grid but grid refinement studies could not be conducted due to exceedingly large computational time on a serial computing platform. Further research evaluating grid refinement and the sensitivity of the computational parameters on the collision dynamics need to be conducted. However, the equivalent diameter of the computed satellite droplet is found to be 15.7% of the initial individual bubble diameter and it is in good agreement with Estrade et al. (1999) observing a value of 17%. Figure 5-38 shows the computational grid and grid data size evolution in time and Figure 5-39 provides the velocity field in x-y plane across the interface.

5.7 Phase Change Computation

A one dimensional phase change test problem used by Son and Dhir (1998), Welch and Wilson (2000) is performed to evaluate the phase change computation technique. The temperature equation was solved sharply without considering any smearing of the fluid properties in the temperature equation. The capability of the technique to capture Stefan condition accurately is demonstrated. Another test case in 3D for a stationary bubble growth in a superheated fluid is considered to examine the asymptotic growth rate in time as compared to the theoretically predicted radial growth that is proportional to the square root of time (Prosperetti & Plesset, 1978).
5.7.1 1-D Phase Change

The left wall shown in Figure 5-40 was maintained at a constant temperature $T_{\text{wall}}$ with the interface location denoted by $\delta(t)$ separating the vapor and liquid phases on the left and right sides respectively. The right side of the domain is kept open to allow the liquid to flow out. The liquid is kept at the saturation temperature. The vapor velocity is zero and as evaporation takes place, liquid is pushed out from the right side.

The theoretical interface location and temperature distribution are given by Equation (5.12) where $\alpha$ and $C$ are diffusivity and heat capacity of the vapor phase and $\lambda$ is the latent heat of evaporation. Parameter $\zeta$ is obtained by solving the transcendental Equation (5.13). The computations are started with the initial interface at $x = 0.2$ and two cases with conductivity ratios of 1 and 10 from Table 5-15 are computed. The parameter $\xi = 1.05968701$ is used based on the solution of transcendental equation for the chosen fluid properties. Computational domain is from $x = 0$ to $x = 1$ with 500 grid cells. The computations are performed from the time when $\delta(t) = 2\xi \sqrt{at} = 0.2$ with the theoretical solution imposed as the initial condition.

Table 5-15. Computational parameters for one-dimensional phase change computation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{wall}}$</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$T_{\text{sat}}$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Liquid and vapor density</td>
<td>(1.0, 0.2)</td>
<td>(1.0, 0.2)</td>
</tr>
<tr>
<td>Liquid and vapor heat capacity</td>
<td>(1.0, 5.0)</td>
<td>(1.0, 5.0)</td>
</tr>
<tr>
<td>Liquid and vapor thermal conductivity</td>
<td>(1.0, 1.0)</td>
<td>(10.0, 1.0)</td>
</tr>
</tbody>
</table>

\[
\delta(t) = 2\zeta \sqrt{at}
\]

\[
T(x,t) = T_{\text{wall}} - \left( \frac{T_{\text{wall}} - T_{\text{sat}}}{\text{erf} \left( \zeta \right)} \right) \text{erf} \left( \frac{x}{2\sqrt{at}} \right)
\]  

(5.12)
\[ \zeta \exp\left(\zeta^2\right) \text{erf}\left(\zeta\right) = C\frac{T_{\text{wall}} - T_{\text{sat}}}{\lambda \sqrt{\pi}} \]  

(5.13)

The computed and theoretical interface location and temperature distribution for Case 1 assigning the same conductivity for both the fluids are shown in Figure 5-41. Also, the density ratio for this case was increased to 1000 and tests were conducted with smeared treatment of the density and heat capacity in the temperature equation. In this case both sharp and smeared treatments produced results within 0.1% of each other.

The second case assigning different conductivity to the two fluids is used to assess the differences between the sharp and smeared treatment for the thermal conductivity in the temperature equation. For this case, the smoothed thermal conductivity in the temperature was computed using Equation (5.14). The computed results with smoothed and sharp treatment are presented in Figure 5-42 with the sharp treatment following the theoretical profile much better than the smoothed treatment.

\[ \frac{1}{K} = \frac{1}{K_i} + \left(\frac{1}{K_i} - \frac{1}{K_v}\right)I \]  

(5.14)

5.7.2 Stationary Bubble Growth:

Growth rate of a stationary vapor bubble in one degree superheated liquid was computed and compared with the theoretical predicted growth rate. A high density ratio of 1000 and viscosity ratio of 22, resembling a water-vapor combination, is used. The thermal conductivity and heat capacity ratios were set to 27 and 2, respectively. The Weber number \((We)\) is 2.5e-5, Reynolds number \((Re)\) is 0.58 and Jacob number \((Ja)\) is set to 3.0. Since the diameter of the bubble changes in time the length scale is defined arbitrarily as \(R_0 = 0.5\). The reference velocity was defined as \(\alpha/R_0\) where \(\alpha\) is the
thermal diffusivity of the liquid. The computation grid was resolved with ~ 25 grid cells per diameter for the initial bubble radius of $R_0$.

Figure 5-43 shows the computed growth rate exhibiting the asymptotic growth rate proportional to $\sqrt{t}$. Since the temperature equation is solved sharply, the interface temperature is well maintained as shown in Figure 5-43 where a thermal boundary layer develops in the liquid phase while the vapor phase remains at saturation temperature.

![Diagram of binary drop collision Case 1 from Table 5-14](image)

Figure 5-29. Binary drop collision Case 1 from Table 5-14. (a) Computed time history, (b) experimental photographs of Qian and Law (1997). [Reprinted with permission]
Figure 5-30. A cross section of the computational grid and the time history of grid data-size for Case 1 from Table 5-14.
Figure 5-31. Binary drop collision Case 2 from Table 5-14. (a) Computed time history, (b) experimental photographs of Qian and Law (1997). [Reprinted with permission]

Figure 5-32. A cross section of the computational domain showing the interface and the velocity vectors for Case 2 from Table 5-14.
Figure 5-33. A cross section of the computational grid and the time history of grid data-size for Case 2 from Table 5-14.
Figure 5-34. Binary drop collision Case 3 from Table 5-14. (a) Computed time history, (b) experimental photographs of Qian and Law (1997). [Reprinted with permission]
Figure 5-35. A cross section of the computational grid and the time history of grid data-size for Case 3 from Table 5-14.
Figure 5-36. A cross section of the computational domain showing the interface and the velocity vectors on the left and pressure profile on the right for Case 3 from Table 5-14.
Figure 5-37. Binary drop collision Case 4 from Table 5-14. (a) Computed time history and (b) the experimental photographs of Qian and Law (1997). [Reprinted with permission]
Figure 5-38. A cross section of the computational grid and the time history of grid data-size for Case 4 from Table 5-14.
Figure 5-39. A cross section of the computational domain showing the interface and the velocity vectors for Case 4 from Table 5-14.
Figure 5-40. One dimensional phase change setup.

(a) Interface location and (b) temperature profile at time = 0.1.

Figure 5-41. Comparison of computed and theoretical solution for Case 1 in Table 5-15.
(a) Interface location and (b) temperature profile at time = 0.1.
Figure 5-42. One-dimensional phase change computation for Case 2 in Table 5-15 showing interface location on the left and temperature distribution at $t = 0.1$ on the right. (a) Computation with smoothed and (b) sharp treatment of thermal conductivity in the temperature equation.
Figure 5-43. Stationary bubble growth rate. (a) Non-dimensional radius and time with dashed line showing the asymptotic growth rate proportion to $\sqrt{t}$; (b) non-dimensional temperature distribution along a line through the center of bubble at $t = 0.18$ with the inside and outside of the interface marked by fluid flags of one and zero, respectively.
CHAPTER 6
SUMMARY AND FUTURE WORK

6.1 Summary

The primary objective of the present work was to develop an effective computational framework that can handle complex interfacial flow computations. The main components of the present work can conceptually be placed into the following four sections:

- Triangulated surface grid tracking to represent the multiphase interface.
- Cartesian grid generation and dynamic adaptation based on interface location and solution features. Development of a staggered grid incompressible Navier-Stokes pressure based solver for spatially variable density, viscosity flows.
- Immersed boundary modeling of interfacial dynamics to account for fluid/flow property changes across interface and surface tension effects.
- Modeling of mass transfer due to phase change across interface.

Brief comments are provided below to summarize the individual aspects and present status of the above four components.

A conservative interface restructuring was used to add and delete markers (interface restructuring) without introducing any error in the interface volume. The volume errors while using conservative restructuring with immersed boundary method occur solely due the errors in the marker advection using the velocity field interpolated from the background Cartesian grids. Since any interface volume error is corrected artificially at the end of every time step, the purpose of using conservative restructuring is to reduce the magnitudes of such artificial corrections. The magnitude of correction
required with conservative restructuring is much smaller and hence reduces solution
disturbance buildup due to artificial relocation of markers to correct the volume errors.

The interface reconstruction is performed as sparingly as possible to minimize
errors caused by reconstruction. Since the interface reconstruction causes artificial
smoothing of the interface, it too comes with some error in the interface volume. These
errors are corrected explicitly and the required corrections were shown to exhibit
quadratic grid convergence.

Using connectivity free tracking of Shin and Juric (2002) could have simplified the
interface tracking but would require frequent global reconstruction of the interface to
maintain marker spacing as well as to accommodate topology changes. It was shown that
a frequent reconstruction may produce undesirable effects for fast moving time scales of
the flow. In this regards, it is believed that the conservative restructuring along with
level-contour based reconstruction provides an enhanced flexibility by maintaining the
interface resolution at all times and performing reconstruction only if topology changes
are expected.

The single phase flow solver was tested for a number of cases to examine the
accuracy of the hydrodynamics and temperature equation solvers with the staggered grid
method. Since the velocity fields on fine faces of a large cell are constrained to be equal
for algorithmic simplicity, a second order accurate solution is obtainable only in the
uniform grid zones. In practice, the grid is uniformly adapted in the zones of interest
(around interface), such a practice cause little degradation in actual performance of the
computation.
The spurious velocity currents in immersed boundary method were tested and shown to be of the order of $10^{-4}$. The overall method was found to exhibit a spatial accuracy of the order of $\sim 1.3$ typical to a continuous interface method where the fluid properties are smeared over few cells across the interface. The interface tracking and its ability to handle complex topology changes and large fluid property jumps were demonstrated via a set of rising bubble coalescence and binary drop collision computations.

Although the hydrodynamics was solved using immersed boundary method with smoothed treatment of the fluid properties, the temperature equation itself was solved utilizing a sharp interface treatment without property smearing. The idea was borrowed from the well known sharp interface level-set method called ghost fluid method and combined with the immersed boundary tracking to solve the mass transfer phenomena as accurately as possible. The idea was primarily to let the diffusion term ‘see’ the same temperature gradients as seen by the numerically computed mass-transfer term. As a consequence, a locally non-conservative discretization of the diffusion term was used near the interface to incorporate the interface temperature as a boundary condition. Such a treatment along with a local interface correction made it easier to maintain the interface at saturation temperature and thus avoided any need to assume constant temperature inside the interface as done in the work of Son (2001).

### 6.2 Future Work

The work conducted during this period is part of a collaborative work aimed at developing an efficient computational capability exploiting the advantages of parallel computing. In view of the work needed for three dimensional multiphase fluid flow computations, parallel computing is highly desirable. However, with dynamically evolving interface geometry and speed, and possible topological changes, it is
challenging to attain desirable parallel efficiency. Several methods and tools developed during this work form a base for further efforts in this direction.

The current capabilities are confined to computations performed in a box-shaped computational domain. Immersed boundary method to model moving/stationary solid boundaries has been used by several researchers for a wide range of computations ranging from turbulent flows (Yang & Elias, 2006), fluid-structure interaction (Zhu & Peskin, 2002; Gilmanov et al., 2003) to multiphase flow (Liu et al., 2005; Udaykumar et al., 2001; Ye et al., 1999). Among these, the method of Ye et al. (1999) and Udaykumar et al. (2001) uses cut-cell approach and is usually more involved than the others. In this regard, the finite difference based method of Liu et al. (2005) is one of the simplest to use. It uses finite difference method near interface and computes the gradients of the flow using the interface conditions (e.g., no slip wall) as a boundary condition. The technique used to discretize the temperature equation in the presented work is similar in nature as it also modeled the interface as a boundary condition and used finite difference for interface cells and finite volume elsewhere.

The computational procedure developed for phase change involves several aspects that need further research for assessing the numerical accuracy and seeking necessary enhancement. An example of this is the artificial correction of temperature in cells across the interface. Its effect with respect to grid resolution, thermal boundary layer thickness and convection dominated flows needs a closer scrutiny although such a correction technique was employed and shown to perform well for a number of nucleate boiling computations by Morgan (2005). The overall phase change computation with explicit marker tracking was shown to perform well for cases considered, further work on more
complex cases such as rising bubbles need to be conducted to evaluate the numerical accuracy of this approach. In addition, for morphological level of phase changes, such as crystal growth problems, when the geometry is highly irregular and the curvatures and surface tension/energy effects more difficult to accurately capture, significant efforts will be required before truly three-dimensional physics of practical interest can be simulated.
LIST OF REFERENCES


BIOGRAPHICAL SKETCH

Rajkeshar Singh was born in 1977 in Uttar Pradesh, India. He was brought up in Mumbai and received his Bachelor of Technology degree in aerospace engineering from the Indian Institute of Technology, in 2000. Thereafter he joined Mississippi State University and graduated in 2002 with Master of Science degree in computational engineering. Since then he has been pursuing his Ph.D. degree in mechanical and aerospace engineering at the University of Florida. His current research interests lie in computational modeling of two phase flows and adaptive grid based computing.