Stony Brook University

Front Tracking Method with High-Order Enhancement and Its Application in Two-Phase Micromixing of Incompressible Viscous Fluids

A Dissertation Presented

by

Yijie Zhou

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

May 2014

Stony Brook University

The Graduate School

Yijie Zhou

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

James Glimm - Dissertation Advisor
Professor, Department of Applied Mathematics and Statistics

Xiangmin Jiao - Chairperson of Defense
Associate Professor, Department of Applied Mathematics and Statistics

Xiaolin Li - Member
Professor, Department of Applied Mathematics and Statistics

Hong Qin - Outside Member
Professor, Department of Computer Science

This dissertation is accepted by the Graduate School.

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Front Tracking Method with High-Order Enhancement and Its Application in Two-Phase Micromixing of Incompressible Viscous Fluids

by

Yijie Zhou

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2014

We develop a front tracking method based on the hydrodynamic library *FronTier* for the solution of the governing equations of motion for two-phase micromixing of incompressible, viscous, liquid-liquid solvent extraction processes. The method is used for accurate simulation of the turbulent micromixing dynamics of an aqueous and an organic phase exposed to intense centrifugal force and shearing stress. The onset of mixing is the result of the combination of the classical Rayleigh-Taylor and Kelvin-Helmholtz instabilities. We demonstrate verification and convergence results for one-phase and unmixed,

two-phase flows. For mixed, two-phase flow a mixing environment that emulates a sector of the annular mixing zone of a centrifugal contactor is used with the mathematical domain small enough to allow for resolution of the individual interfacial structures and large enough to allow for an analysis of their statistical distribution of sizes and shapes. Such a statistical picture provides the information needed for building a consistent coarsened model applicable to the entire mixing device. We reach a stable two phase configuration as a statistically steady state in late time after going through a fully mixed transient chaotic flow regime with a high surface area. To handle problems introduced by the extreme complexity of interfaces, a new parallel triangular mesh library called *HiProp* is implemented which serves as the basis for high-order mesh algorithms. The new library keeps a full list of parallel information for each point and triangle so that each element has a unique master processor and global ID. No floating point comparison is needed after the parallel information is built. The utilities for building ghost triangles while keeping the parallel information updated based on either connectivity or domain decomposition are implemented for applying different high-order mesh algorithms. We develop parallel high-order mesh smoothing, parallel high-order normal and curvature calculation and point propagation based on the new structure. A novel high-order functional mesh propagation algorithm is also developed for propagating local polynomial patches instead of separate points to get high-order results not only for point positions but also for higher order differential quantities such as normals and curvatures. To have a complete mesh propagation package, we also implement a tangle detection algorithm and with the

I/O interface between *FronTier* and *Hiprop* we only go back to *FronTier* for untangling the self-intersection. It minimizes the time for transfering data between two libraries. In the future a more accurate untangling algorithm would be developed based on the new structure and the data transfer could be entirely eliminated.

Key Words: front tracking, Taylor-Couette flow, turbulent fluid flow mixing, parallel triangular mesh, high-order algorithms.

To my family,

My Parents, Jian Zhou, Ling Ma
My Wife, Fangchen Liu and Our Son Derek Zhou

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost I would like to thank my advisor Professor James Glimm. It has been an honor to be his Ph.D. student and he has been a tremendous mentor for me. He always encourages my research and allows me to grow not only as a Ph.D. student but also as a researcher. I appreciate all his contributions of time, ideas and funding to make my Ph.D. experience productive and stimulating. His advice on both research subject and, more importantly, on working and researching attitude have been priceless. The excellent role model he provided as a scientist would always be valueable for me in the future no matter what area I would be working in.

I would like to give my special thanks to two of my defense committee members, Professor Xiangmin Jiao and Professor Xiaolin Li. Both of them are very intellegient researchers and gave me valueable guidance during my Ph.D.. The high-order polynomial fitting technique used in the thesis comes originally from Prof. Jiao and most of the code for high-order mesh algorithms are part of his NUMGEOM project. As an expert and major developer for *FronTier*, Prof. Li gave me lots of help on *FronTier*, which is the foundation for our front-trakcing method. I would also like to thank Professor Hong Qin for his time and valueable suggestions as my defense committee member.

Also I would like to thank Hyunkyung Lim, with whom I worked together on the contactor project. She has always been a great researcher and a good friend. Without her rich knowledge on both physics and computer science the simulation could not be done.

I would like to thank all my friends and colleagues during my Ph.D. study at Stony Brook for thier friendship and encouragement. In particular, I would like to mention Drs. Wurigen Bo, Shuqiang Wang, Tongfei Guo, Tulin Kaman, Ying Xu, Yan Li, Yijing Hu and Wenlin Hu, Shuai Xue. They have shared with me many interesting and inspiring ideas.

Last but not least, I would like to thank my family. My parents has always been supportive for whatever I choose to do in my life. With out their unconditional love I wouldn't have been able to accomplish all that I have so far. My lovely wife, Fangchen Liu, has been standing by me all the time, good or bad. Your love has always been the most useful cure when I was down and I love you from the bottom of my heart. Also our newly born son, Derek Zhou, you are the most amazing gift that I have received in my life. This disseration would never be finished without the support of my family so it is dedicated to my parents, my wife and my son.

In the end, for all the other people who helped me in my life that I did not mention, thank you.

Chapter 1

Introduction

1.1 Overview and Motivation

Mixing of multiple phases, as a problem in computational continuum physics, has both practical and basic scientific interest. We consider here two-phase mixtures of immiscible, incompressible, viscous liquids with non-vanishing interfacial tension. A variety of methods have been proposed for solving the equations of motion for two-phase incompressible flow, all of which include a numerical description of the interface between two fluids and a numerical scheme for solving multiphase incompressible Navier-Stokes equations with interface. A high-order description of a discrete mesh is also critical for getting a high-order interface method especially when interfacial tension plays an important rule in the application. After a brief overview of all three major parts for an accurate interface method, we present in the following chapters our computational framework which has high-order in some if not all aspects of the front tracking method we used for our simulation of the micromixing problem in a high-speed rotating centrifugal contactor. In addition, for compli-

cated fluid dynamics problems, parallel computing with domain decomposition is needed even with high-order methods. Thus a robust and efficient parallel mesh structure based on message passing is also implemented serving as the basis for parallel high-order mesh algorithms.

1.1.1 Interface methods

Many methods have been proposed to represent and evolve an interface between two immiscible fluids. Of these, the volume of fluid (VOF), the level set, and front-tracking methods are the most popular. For a complete review of these methods, readers are referred to the papers of [26], [64], [22], [24], [73], [61]. When compared to the front capturing methods (VOF and level set), front tracking preserves a more accurate interface representation with less numerical dissipation. In addition, the implicit methods suffer from the drawback that they are unable to capture interface details near or below the resolution of the underlying grid. Even for recently developed gradient-augmented level-set method [53], which is based on a Hermite interpolation for reconstruction of subgrid meshes, non negligible volume loss is still observed in the benchmark tests. For those physics problems such as fluid mixing which needs an accurate representation of the interface, front tracking is a better choice in our opinion. A comparison of these three methods [17] concluded that the front tracking to be more accurate and faster to converge for a number of benchmark problems.

We use front tracking methods to simulate multiphase flows, based on the front tracking code *FronTier*. This is a hydrodynamic library for the geometrical manipulation of a mathematical surface coupled with multiple

partial differential equations. It has been employed in a variety of multiphase simulations [23, 24, 17] for problems dominated by a geometrically complex dynamic interface. For example, extensive simulation studies of the Rayleigh-Taylor instability [45, 46, 43] delivered agreement with experiments in the overall growth rate as defined by the mixing growth parameter.

The *FronTier* package provides algorithms for geometric construction and reconstruction of a surface embedded in the surrounding three-dimensional space. Surface normal, curvature, area, enclosed volume, and statistical information of disconnected surfaces are calculated robustly and accurately. These geometric operations are necessary for enabling the evolution of fluid interfaces in practical simulations.

Topological bifurcations of an interface occur frequently in many applications and require special front tracking methods. Our *FronTier* package uses a triangular representation of a surface. That is, a three-dimensional triangular surface mesh is used. Triangles on the interface must be allowed to reconnect with each other after crossing when dynamically convected by the surrounding fluid flow. Three methods are implemented in *FronTier* to handle topological transitions. The grid free method (GF) [23] is accurate but is prone to logical errors. A grid based method (GB) [24] is robust but suffers from excessive interpolation and smoothing errors. The most advanced, the locally grid-based (LGB) method [17], combines the advantages of the GF and GB methods. It identifies degenerated triangles on the propagated surface, isolates them, and preserves the intersections of the surface with the grid cell edges to allow for a GB reconstruction locally near the defective region. Triangles neighboring

this region are removed giving rise to a gap between the pristine interface and the reconstructed part. The major geometry task is to re-seal this gap.

In [7] the LGB reconstruction is improved by reducing the size of the GB region, which previously was the smallest rectangular solid containing a tangled set; and in the case of overlaps, the smallest rectangular solid containing the overlap, etc. In order to keep the triangular mesh topologically valid, two constraints on the triangular mesh are imposed: (1) an edge of a triangle connects with at most one triangle; (2) an non-boundary vertex has only one associated list of triangles formed by linking successively adjacent triangles.

To parallelize the improved LGB algorithm we construct ghost cells, typically a few layers thick, so that after ghost cell communications, each processor utilizes only local information to reconstruct the interface contained within its local cells. If the span of the tangling is too large for the ghost cells, we collect the cells containing the tangled region onto one processor for resolution. Because such situations occur very rarely, this fall back strategy does not affect the efficiency.

1.1.2 Numerical methods for fluid flows with interfaces

In recent decades, a large effort has been devoted to interface-resolving methods for the solution of the multiphase, incompressible Navier-Stokes equations. The Immersed Boundary Method (IB), Immersed Interface Method (IIM), Embedded Boundary Method (EBM) and the Ghost Fluid Method (GFM) are the most popular methods for solving this class of fluid flow problems.

The IB method, introduced by [54] to model blood flow in a human heart, uses a discrete delta function to spread the singular force on the interface to a Cartesian grid by means of a numerical width multiple of the grid spacing. This approach was extended [74] to treat three-dimensional, multi-phase incompressible fluid flows including complex topological changes. In [68] the front tracking description of the interface [74] was replaced by a level set description [55, 52].

The IB method has been successfully applied to many fluid and biological problems. It is first-order due to the smoothing of the interface. The numerical solution is continuous at the interface even if the actual interface conditions imply that the solution should be discontinuous.

The IIM, introduced by [39], uses a regular Cartesian grid with an interface represented by marker points. Instead of using a discrete delta functions and/or Heaviside functions, the velocity and pressure jump conditions at the interface are preserved and coupled into the finite difference scheme resulting in a high-order numerical method. This method was initially implemented in one-dimensional and two-dimensional elliptic equations with discontinuous coefficients and singular source terms. The IIM was then generalized to many types of equations with jump conditions at the interface, and a review of this body of work is found elsewhere [41, 40].

In [69, 70] the IIM was generalized to apply to the incompressible Navier-Stokes equation with discontinuous viscosity, using the jump conditions of [27, 38]. However, the method does not allow discontinuities in the density for the Navier-Stokes solutions. It is second-order accurate in the absence of

density discontinuities. With density discontinuities, which must be smeared over some length scale, as in the IB method, the method is first-order accurate. The IIM relies on a local coordinate system, which is quite complicated and whose generalization to three or more fluid phases appears to be difficult.

The GFM is a sharp interface method which preserves the jump conditions and uses them to construct ghost fluid states for updating each phase independently. As with the IB method, it is a first-order method but it differs from the IB in its treatment of the interface. It was used for a Poisson's equation [44] and for a multi-phase incompressible flow [34].

The GFM could be used to treat three-dimensional, multi-phase incompressible fluid flow, including the effect of viscosity, surface tension and gravity, eliminating the numerical smearing prevalent in the discrete delta function formulation of the IBM. The GFM is relatively simple to implement. The sharp interface allows complete physics including a discontinuous density. Unlike the IIM, it involves linear systems with symmetric matrices of coefficients which can be solved by many fast solvers. The GFM is only first-order accurate because of a simplification used in the jump conditions when constructing the ghost states [73, 72, 20].

The EBM uses a Cartesian grid as the computational domain with the irregular boundary embedded in the grid. The states are calculated at grid cell centers even if they are outside of the boundary. It is proved both theoretically and numerically that this method is second-order accurate. The EBM was first introduced for a Poisson's equation on an irregular domain in [32]. It was generalized to parabolic equations with a moving boundary [49], three-

dimensional elliptic and parabolic equations [62], hyperbolic conservation laws [14], and compressible [37] and incompressible [3] single-phase, Navier-Stokes equations on irregular domains.

The EBM was extended to a two-sided interface, as opposed to a one-sided boundary [77], with application to two-phase flows. The EBM incorporates the jump conditions for elliptic and parabolic equations with additional states defined not only at the cell center, but also at the interface patch center, to yield a second-order accurate algorithm for both the interface problem and for the irregular boundary problem in a unified framework. It is also easily generalized to three or more phases through additional jump conditions and states. The difficulty in obtaining second-order accuracy, especially in the case of a large density discontinuity at the interface, is associated with the projection step.

1.1.3 High-order surface reconstruction

To obtain high-order approximation for a discrete mesh, a continuous method based on polynomial fittings was proposed in [50] for calculating normals and curvature. A unified and robust weighted least squares based local polynomial fitting technique for computing high-order accurate approximations to both geometry and differential quantities was first given in [31]. The technique was generalized from height functions to parametric surfaces in [76] with a full theoretical and experimental comparison for a number of polynomial-fitting based methods using different parameterizations and numerical solvers for least squares problem was given. In [76] it was proved that

the methods based on local orthogonal projection with a safeguard against folding delivers the best result in terms of both accuracy and robustness. To overcome the discontinuity caused by local polynomial fitting, two methods called WALF (Weighted Averaging of Local Fittings) and CMF (Continuous Moving Frames) [30] was proposed to obtain continuous support for geometry. Both methods guarantee C^0 continuity while keeping high-order accuracy for the reconstructed surface. A number of applications based on this framework for high-order surface reconstruction have been presented. A high-order surface re-meshing method was introduced in [13] and a more robust version with geometrical limiter was given in [58]. Both the optimization and adaptivity operations used in this re-meshing method preserve the geometry to high-order. An untangling process for mildly folded triangles is also used in the method for generating a smooth output surface. Other applications include a high-order numerical integration method [59] based on the same framework of local polynomial fitting.

1.2 Principal Results

We build an incompressible fluid solver with sharp interface tracking based on the front tracking method referred in Sec. 1.1.1 and Immersed Boundary Method (IBM) referenced in Sec. 1.1.2. It also uses the high-order surface reconstruction referenced in Sec. 1.1.3 to obtain high-order normals and curvatures for accurately calculating the interfacial force. Through a series of verification studies for one-phase flows and unmixed, laminar two-phase flows, we demonstrate our method to be second order accurate for one-phase flows

and first order accurate with interface. For one-phase Taylor-Couette flows in various flow regimes with different Reynold's numbers, we present numerical results consistent with both experimental pictures for Taylor vortices and consistent with linear stability analysis for transition from laminar Couette flow to Taylor vortices. Also a comparison against both experimental and direct numerical simulation (DNS) results for turbulent Taylor-Couette flow with Reynold's number 8000 shows that with our subgrid scale model we have slightly improved convergence and the results agree with both experiment and DNS under mesh refinement.

For our two-phase mixing simulation with a mixing environment emulating a sector of the annular mixing zone of a centrifugal contactor, our primary discovery is the existence two distinct, and weakly communicating, flow regimes, with the heavy fluid predominantly on the outside and the light fluid predominantly on the inside in late time. We call the regime the centrifuge regime. This is not consistent with the experimental picture for the late time flow with microstructure consisting of droplets of some 60 microns in size and a single connected phase. Honeycomb structure with thin lubricating walls of some 10 microns in thickness of droplets are observed. We believe the primary difference of simulation with experiment is due to the treatment of subgrid fluctuations, which in the present simulation call for merging of droplets when they come in contact. These merged droplets give rise to the two distinct continuous flow regimes we observe in the simulations. Larger simulation domain sizes, three phase flow with air and further simulation to later time may also play a role in the difference.

To handle problems introduced by the extreme complexity of interfaces, especially arise in fluid mixing problems such as the contactor problem, we design a new parallel triangular mesh library called *HiProp*. The new library is more robust by maintaining a full parallel information list for identifying each point and triangle globally. Any global mesh operation would have a unique result without any floating point comparison. In this way the new library is more robust and efficient comparing with the old communication model in *FronTier*. A parallel ghost exchange algorithm is implemented for building ghost triangles necessary for various high-order parallel mesh algorithms and parallel fluid solvers based on domain decomposition. We implement parallel mesh smoothing, normal and curvature calculation and point propagation based on the new structure and more parallel mesh algorithms could be designed base on this library.

Based on the local polynomial fitting technique, we also present a new functional propagating algorithm which propagates the points positions and normals at the same time by propagating local polynomial patches at each point. The algorithm is parallelized under the framework of the new parallel library and we present high-order results for both point positions and high-order differential quantities such as normals and curvatures for benchmark problems.

1.3 Dissertation Organization

In Chapter 2, we will introduce the mathematical model and the numerical methods we used for contactor simulation including Front Tracking

and its coupling with an incompressible fluid solver using Immersed Boundary Method for two-phase flow. In Chapter 3, the mathematical description for local polynomial fitting, which serves as the foundation for the high-order normal and curvature calculation used in the contactor problem is introduced. Using the same technique, we also present a new high-order functional mesh propagation method, which propagates polynomial patches by solving a PDE instead of solving ODEs for separate points. Using high-order polynomial fitting, we achieve both high-order results for point positions and higher order differential quantities such as normals and curvatures. In Chapter 4, convergence and order of accuracy for our method are demonstrated for one-phase flows in both Cartesian and Cylindrical coordinates. We also present a validation test on one-phase Taylor-Couette flow in a high speed, turbulent flow regime. Accuracy and convergence specific to this flow is considered by way of calculations of the transition from laminar flow to Taylor-Couette vortices. Growth rates of perturbations slightly above the critical Taylor number are presented. Chapter 5 first demonstrates verification of solutions for laminar two-phase flows. Then under the mixing environment that emulates a sector of the annular mixing zone of a centrifugal contactor, we investigate the mixing and dispersion of organic/aqueous phases and present the existence of two separate continuous phase regions in steady state at late time. By phase visualization and statistical analysis of interfacial area and droplet size, we demonstrate that the stable segregated two phase configuration is reached after going through a complicated, fully mixed transient chaotic flow regime with a high surface area. We also discuss the difference between simulation

and experiment related to the subgrid scale modeling of thin lubrication films existing between droplets in continuous phase. In Chapter 6, we introduce a new parallel triangular mesh library called *HiProp* maintaining full list of parallel information i.e., processor domain location for each point and triangle which serves as the basis for various high-order parallel mesh algorithms. We discuss the data structure, the message communication model and the algorithms for building the parallel information and for updating it during various ghost exchange procedures based on connectivity and domain decomposition. Examples of two parallel mesh algorithms implemented based on *HiProp* are given demonstrating the use of the new mesh library. The numerical results for the high-order functional propagation algorithm introduced in Chapter 3 is also presented under the same framework of the new libraray. At the end of this chapter a full mesh propagating package is presented by coupling *HiProp* and *FronTier* with an exact tangle detection algorithm added. Chapter 7 will discuss some of the on-going work for front tracking method and parallel high-order mesh algorithms based on the *HiProp* library.

Chapter 2

Mathematical Model and Numerical Methods

2.1 Equations of Motion

The incompressible Navier-Stokes equations (NSE) for a homogeneous, Newtonian fluid phase (α) describe a relationship between the velocity $\boldsymbol{v}_\alpha(\boldsymbol{x}_\alpha, t)$ and the hydrodynamic pressure $p_\alpha(\boldsymbol{x}_\alpha, t)$ fields at any spatial point $\boldsymbol{x}_\alpha$ in the bulk of a three-dimensional, bounded domain $\Omega_\alpha(t)$ occupied by the fluid phase at the instant t, namely

$$\rho_\alpha \partial_t \boldsymbol{v}_\alpha + \rho_\alpha \mathrm{div}_{\boldsymbol{x}}(\boldsymbol{v}_\alpha \otimes \boldsymbol{v}_\alpha) = \mathrm{div}_{\boldsymbol{x}} \boldsymbol{T}_\alpha + \boldsymbol{b}_\alpha \quad \text{in} \quad \Omega_\alpha(t), \quad \text{and} \tag{2.1a}$$

$$\mathrm{div}_{\boldsymbol{x}} \boldsymbol{v}_\alpha = 0 \quad \text{in} \quad \Omega_\alpha(t), \tag{2.1b}$$

where $\boldsymbol{T}_\alpha(\boldsymbol{x}_\alpha, t)$ is the Cauchy stress tensor and $\boldsymbol{b}_\alpha(\boldsymbol{x}_\alpha, t)$ is the body force on the fluid evaluated at $\boldsymbol{x}_\alpha \in \Omega_\alpha(t)$. A fluid with homogeneous Newtonian behavior is characterized by its constitutive equation for the stress $\boldsymbol{T}_\alpha(\boldsymbol{x}_\alpha, t) := -p_\alpha \boldsymbol{I} + \mu_\alpha\big(\nabla_{\boldsymbol{x}} \boldsymbol{v}_\alpha + \nabla_{\boldsymbol{x}} \boldsymbol{v}_\alpha^T\big)$. Equation (2.1b) is the reduced form of the principle of mass conservation for an isochoric flow. Equation (2.1a) is the momentum

balance for the fluid when incompressibility holds. In a two-phase system, another set of equations similar to (2.1) are needed for the β phase. The immiscibility condition state $\Omega_\alpha \cap \Omega_\beta = \emptyset$.

The mass density ρ_α and the dynamic viscosity μ_α are constants provided by experimental measurements. The typical body force is the result of the gravitational acceleration on the mass of the phase.

When two immiscible fluid phases (α and β) are put into contact, an inhomogeneous, three-dimensional, interfacial region between the two fluids is formed. A classical continuum approximation of this region is made by the Gibbs' dividing surface, *i.e.*, the region is approximated by a mathematical surface $\mathcal{S}(t) = \operatorname{Clo}\Omega_\alpha \cap \operatorname{Clo}\Omega_\beta$, that is, the intersection of the closure of the domains occupied by the fluid phases.

The equations (2.1) hold for each fluid phase up to the interfacial surface, where a discontinuity in mass density ρ and dynamic viscosity μ may exist ($\rho_\alpha \neq \rho_\beta$ and $\mu_\alpha \neq \mu_\beta$). The intrinsic motion of the interface is governed by a momentum and mass balance on $\mathcal{S}$, also referred to as pointwise jump conditions, respectively

$$\operatorname{div}_{\mathcal{S}}\mathbb{T} + [\![\boldsymbol{T}\cdot\boldsymbol{n}]\!] = \boldsymbol{0} \quad \text{on} \quad \mathcal{S}(t), \quad \text{and} \tag{2.2a}$$

$$[\![(\boldsymbol{v} - \dot{\boldsymbol{x}})\cdot\boldsymbol{n}]\!] = 0 \quad \text{on} \quad \mathcal{S}(t), \tag{2.2b}$$

where these equations hold for the particular case of *no mass transport* between phases, and in the *absence of any mass density excess* associated to the interface. If these two conditions do not hold, then (2.2) are incomplete. The

tangential surface stress tensor, $\mathbb{T}$, defines the interfacial state of stress while the interface velocity is denoted by $\dot{\boldsymbol{x}}$. The notation $[\![\cdot]\!]$ represents the jump across the interface of the quantity inside the brackets, that is, $[\![\cdot]\!] := (\cdot)_\alpha - (\cdot)_\beta$ with $\boldsymbol{n}$ denoting the local unit normal vector on $\mathcal{S}$ pointing into α, that is, $\boldsymbol{n} := \boldsymbol{n}_{\overrightarrow{\beta\alpha}}$.

Therefore, these bracket terms couple the intrinsic motion of the interface with the neighboring phases. An interface with no intrinsic viscosity can be described by a constitutive stress of the form $\mathbb{T} := \sigma\mathbb{I}$, with a constant interfacial tension σ. Therefore in (2.2a), the surface divergence of the interfacial stress tensor for an inviscid interface results in the familiar surface curvature term

$$\mathrm{div}_{\mathcal{S}}(\sigma\mathbb{I}) = 2\mathcal{H}\sigma\boldsymbol{n},$$

where $\mathcal{H}$ is the local mean curvature of $\mathcal{S}$.

Modulo initial and boundary conditions, the governing equations (2.1) as applied within each continuous, homogeneous phase, coupled with the interfacial balances (2.2), describe the motion of immiscible, viscous phases in intimate contact by virtue of their interfaces. The rigorous derivation of the governing equations for the motion of fluid phases in contact through a Newtonian interface was first made by [63] in the so called Eulerian form.

A comment on (2.2b) is in order. Only the jump in the normal component of the velocity of the fluid relative to the interface surface velocity must vanish. This enforces no mass transfer between the phases and allows for slip of one phase past another, that is, the tangential component of the relative velocity is not constrained. This may be of significance when the fluids

are non-Newtonian, high-molecular-weight polymers. For our simulation we assume that no slip exists and (2.2b) should be replaced by

$$[\![(\boldsymbol{v} - \dot{\boldsymbol{x}})]\!] = \mathbf{0} \text{ on } \mathcal{S}(t),$$

which incorporates the continuity of the phase velocities $\boldsymbol{v}_\alpha$ and $\boldsymbol{v}_\beta$ on $\mathcal{S}$.

Calculation of the three-dimensional time evolution of $\mathcal{S}$ and the accompanying fields, $\boldsymbol{v}_\alpha$,$\boldsymbol{v}_\beta$, p_α, p_β, and $\dot{\boldsymbol{x}}$, is a formidable computational task in systems involving vigorous mixing. The topological changes subjected by $\mathcal{S}$, lead to a dispersion of the phases with many filaments, drops, and other disconnected surface elements as it is typically visualized in flow mixing experiments. Hence $\mathcal{S}$ could be highly disconnected. A central computational difficulty is how to apply the interfacial momentum balance (2.2a) on $\mathcal{S}$, while resolving topological transitions which often lead to the formation of contact points, lines, surfaces, and cusps. This difficulty is exacerbated when three-dimensional vigorous mixing is present.

In order to make (2.1)–(2.2) tractable computationally, an equivalent set of equations is first introduced and an approximation is made by considering the mass density and dynamic viscosity as varying in time and space across the union of the domains occupied by the phases, while the divergence of the interfacial stress term is incorporated in the momentum balance as a source term in a distribution sense. The reformulation and approximation are elaborated next and comments are made on the associated adverse effects.

2.1.1 Reformulated equations of motion

The interfacial equations (2.2) can be absorbed into (2.1) to obtain equations defined over an entire fixed domain, $\Omega := \Omega_\alpha(t) \cup \Omega_\beta(t)$, by prescribing a variable mass density and dynamic viscosity in space and time so that a fluid phase is identified by the value of these quantities on each side of the surface $\mathcal{S}$(t). Hence

$$\rho \partial_t \boldsymbol{v} + \rho \mathrm{div}_{\boldsymbol{x}}(\boldsymbol{v} \otimes \boldsymbol{v}) = \mathrm{div}_{\boldsymbol{x}} \boldsymbol{T} + \boldsymbol{b} + \boldsymbol{f} \quad \text{in} \quad \Omega, \quad \text{and} \tag{2.3a}$$

$$\mathrm{div}_{\boldsymbol{x}} \boldsymbol{v} = 0 \quad \text{in} \quad \Omega, \tag{2.3b}$$

where

$$\rho(\boldsymbol{x}, t) := \begin{cases} \rho_\alpha & \forall\ \boldsymbol{x} \in \Omega_\alpha(t), \\ \rho_\beta & \forall\ \boldsymbol{x} \in \Omega_\beta(t), \end{cases} \qquad \boldsymbol{v}(\boldsymbol{x}, t) := \begin{cases} \boldsymbol{v}_\alpha & \forall\ \boldsymbol{x} \in \mathrm{Clo}\,\Omega_\alpha(t), \\ \boldsymbol{v}_\beta & \forall\ \boldsymbol{x} \in \mathrm{Clo}\,\Omega_\beta(t), \\ \dot{\boldsymbol{x}} = \boldsymbol{v}_\alpha = \boldsymbol{v}_\beta & \forall\ \boldsymbol{x} \in \mathcal{S}(t), \end{cases} \tag{2.4a}$$

$$\boldsymbol{T}(\boldsymbol{x}, t) := \begin{cases} \boldsymbol{T}_\alpha & \forall\ \boldsymbol{x} \in \Omega_\alpha(t), \\ \boldsymbol{T}_\beta & \forall\ \boldsymbol{x} \in \Omega_\beta(t), \end{cases} \qquad \boldsymbol{b}(\boldsymbol{x}, t) := \begin{cases} \boldsymbol{b}_\alpha & \forall\ \boldsymbol{x} \in \Omega_\alpha(t), \\ \boldsymbol{b}_\beta & \forall\ \boldsymbol{x} \in \Omega_\beta(t), \end{cases} \tag{2.4b}$$

and

$$\boldsymbol{f}(\boldsymbol{x}, t) := \int_{\Omega(t)} \mathrm{div}_{\mathcal{S}}(\sigma \mathbb{I})\, \delta(\boldsymbol{x} - \boldsymbol{x}_{\mathcal{S}})\, ds. \tag{2.5}$$

The apparent appeal of this reformulation is that the equations of change, specified on Ω, are reduced in number. In addition, it provides a easier way

to reason regarding develop approximate solution strategies based on classical partial differential equations defined on a fixed domain. Therefore conventional solution methods based on the partition of fixed domains (meshing) can be designed, in principle. The basic idea is to solve the problem on a fixed domain with an embedded, moving surface $\mathcal{S}(t)$ wherein the source force $\boldsymbol{f}$ is applied. This can be accomplished in two different ways, namely, the interface can be either explicitly tracked or implicitly captured (Sec. 1.1.1).

The proof that (2.3)–(2.5) is a valid mathematical statement of equivalence to (2.1)–(2.2) hinges in the definition of the continuous velocity field in (2.4), and the source force (2.5). In the latter, the convolution of the interfacial stress divergence with a delta function centered on $\mathcal{S}$, applies a force impulse on the interface which reproduces the jump condition (2.2a). Hence, $\boldsymbol{f}(\boldsymbol{x},t) = \boldsymbol{0}\ \forall\ \boldsymbol{x} \notin \mathcal{S}(t)$. Note that the traction jump in (2.2a), that is, $[\![\boldsymbol{T}\cdot\boldsymbol{n}]\!]$, is taken into account via the definition of the discontinuous Cauchy stress tensor in (2.4) and (2.3a) and in a weak sense.

In summary, problem (2.3)–(2.5) is an equivalent statement of the discontinuous equations of motion presented in the previous section. However, there exists an obvious difficulty when building numerical solution scheme based on the above equations, namely, evaluating (2.5) with sufficient accuracy. In practical computations, a variety of ways exist for approximating this integral and for defining $\mathcal{S}$. The particular case wherein a fixed domain is swept by $\mathcal{S}$, produces numerical artifacts called spurious currents, which are artificial flow recirculations near the interface due to insufficient accuracy when balancing pressure gradients and interfacial tension [56, 61, 28]. The adverse effects

range from uncontrollable instabilities to difficulties in accurately computing gradients near the interface. In dispersed mixing flows, this can be a significant drawback since it limits the calculation of volume fraction and interfacial area in the long-time regimes.

2.1.2 Continuous approximation of the equations of motion

The Immersed Boundary Method, which will be discussed in detail in Sec. 2.2.2, assumes a continuous approximation for the original discontinuous interface model. This entails proposing a particular form of continuous fields for ρ, μ, $\boldsymbol{b}$, and a volume integral approximate for the source force $\boldsymbol{f}$ area integral. To achieve this objective, the balance equations become

$$\rho\,\partial_t \boldsymbol{v} + \rho \mathrm{div}_{\boldsymbol{x}}(\boldsymbol{v}\otimes\boldsymbol{v}) = \mathrm{div}_{\boldsymbol{x}}\boldsymbol{T} + \boldsymbol{b} + \boldsymbol{f} \quad \text{in} \quad \Omega, \quad \text{and} \tag{2.6a}$$

$$\mathrm{div}_{\boldsymbol{x}}\boldsymbol{v} = 0 \qquad\qquad\qquad\qquad \text{in} \quad \Omega, \tag{2.6b}$$

where

$$\rho(\boldsymbol{x},t) := \begin{cases} \rho_\alpha & \forall\ \boldsymbol{x}\in\Omega_\alpha(t) \mid (\boldsymbol{x}-\boldsymbol{x}_{\mathcal{S}})\cdot\boldsymbol{n} > \varepsilon, \\ \rho_\beta & \forall\ \boldsymbol{x}\in\Omega_\beta(t) \mid (\boldsymbol{x}-\boldsymbol{x}_{\mathcal{S}})\cdot\boldsymbol{n} < \varepsilon, \\ \varphi_\rho(\rho_\alpha,\rho_\beta,\boldsymbol{x}) & \forall\ \boldsymbol{x}\in\Omega \ \mid -\varepsilon \le (\boldsymbol{x}-\boldsymbol{x}_{\mathcal{S}})\cdot\boldsymbol{n} \le \varepsilon, \end{cases} \tag{2.7a}$$

$$\mu(\boldsymbol{x},t) := \begin{cases} \mu_\alpha & \forall\ \boldsymbol{x} \in \Omega_\alpha(t) \mid (\boldsymbol{x}-\boldsymbol{x}_\mathcal{S})\cdot\boldsymbol{n} > \varepsilon, \\ \mu_\beta & \forall\ \boldsymbol{x} \in \Omega_\beta(t) \mid (\boldsymbol{x}-\boldsymbol{x}_\mathcal{S})\cdot\boldsymbol{n} < \varepsilon, \\ \varphi_\mu(\mu_\alpha,\mu_\beta,\boldsymbol{x}) & \forall\ \boldsymbol{x} \in \Omega\ \mid -\varepsilon \leq (\boldsymbol{x}-\boldsymbol{x}_\mathcal{S})\cdot\boldsymbol{n} \leq \varepsilon, \end{cases} \tag{2.7b}$$

$$\boldsymbol{b}(\boldsymbol{x},t) := \begin{cases} \boldsymbol{b}_\alpha & \forall\ \boldsymbol{x} \in \Omega_\alpha(t) \mid (\boldsymbol{x}-\boldsymbol{x}_\mathcal{S})\cdot\boldsymbol{n} > \varepsilon, \\ \boldsymbol{b}_\beta & \forall\ \boldsymbol{x} \in \Omega_\beta(t) \mid (\boldsymbol{x}-\boldsymbol{x}_\mathcal{S})\cdot\boldsymbol{n} < \varepsilon, \\ \boldsymbol{\varphi_b}(\boldsymbol{b}_\alpha,\boldsymbol{b}_\beta,\boldsymbol{x}) & \forall\ \boldsymbol{x} \in \Omega\ \mid -\varepsilon \leq (\boldsymbol{x}-\boldsymbol{x}_\mathcal{S})\cdot\boldsymbol{n} \leq \varepsilon. \end{cases} \tag{2.7c}$$

$$\boldsymbol{v}(\boldsymbol{x},t) := \begin{cases} \boldsymbol{v}_\alpha & \forall\ \boldsymbol{x} \in \operatorname{Clo}\Omega_\alpha(t), \\ \boldsymbol{v}_\beta & \forall\ \boldsymbol{x} \in \operatorname{Clo}\Omega_\beta(t), \\ \dot{\boldsymbol{x}} = \boldsymbol{v}_\alpha = \boldsymbol{v}_\beta & \forall\ \boldsymbol{x} \in \mathcal{S}(t). \end{cases} \tag{2.7d}$$

All continuous fields in (2.7), with the exception of the velocity field, interpolate the value of the field in the bulk of the β phase to the α phase within a thin region of thickness $\varepsilon > 0$. If $\boldsymbol{x}$ is a point near $\mathcal{S}$, $\boldsymbol{x}_\mathcal{S}$ is the point on $\mathcal{S}$ colinear to $\boldsymbol{x}$ along $\boldsymbol{n}(\boldsymbol{x}_\mathcal{S})$, thus $(\boldsymbol{x}-\boldsymbol{x}_\mathcal{S})\cdot\boldsymbol{n} = \left\|\boldsymbol{x}-\boldsymbol{x}_\mathcal{S}\right\|\ \forall\ \boldsymbol{x}$. The interpolants φ_ρ, φ_μ, and $\boldsymbol{\varphi_b}$ are constructed from an analytical function within the interpolation region around $\mathcal{S}$, which dynamically changes with time.

The accuracy of this continuum approximation depends on the preservation of the incompressibility of the fluid and its dynamical viscosity in the region of thickness ε. From mass conservation in the thin region around the

interface one obtains

$$\mathrm{div}_{\boldsymbol{x}}\boldsymbol{v} = -\frac{D_t\varphi_\rho}{\varphi_\rho} \quad \forall\ \boldsymbol{x} \in \Omega \ \mid -\varepsilon \leq (\boldsymbol{x} - \boldsymbol{x}_{\mathcal{S}}) \cdot \boldsymbol{n} \leq \varepsilon, \tag{2.8}$$

where $D_t(\cdot)$ is the material time derivative. Therefore in view of (2.6b) a sufficient condition for maintaining incompressibility in the thin region around the interface is to build the interpolant $\varphi_\rho(\cdot)$ such that $D_t\varphi_\rho = 0$ on all points in the interpolation region, that is,

$$\partial_t\varphi_\rho + \nabla_{\boldsymbol{x}}\varphi_\rho \cdot \boldsymbol{v} = 0 \quad \forall\ \boldsymbol{x} \in \Omega \ \mid -\varepsilon \leq (\boldsymbol{x} - \boldsymbol{x}_{\mathcal{S}}) \cdot \boldsymbol{n} \leq \varepsilon \tag{2.9}$$

for all times.

Note that the flow is isochoric everywhere but it does not imply incompressibility of the fluid in the thin region near the interface unless the interpolant φ_ρ is compatible through (2.9). An argument similar to (2.9) must hold for the dynamic viscosity field wherein the material derivative must vanish to recover the original constant value. That is, φ_μ should be build with the condition that $D_t\varphi_\mu = 0$ for all points in the interpolation interfacial region, hence

$$\partial_t\varphi_\mu + \nabla_{\boldsymbol{x}}\varphi_\mu \cdot \boldsymbol{v} = 0 \quad \forall\ \boldsymbol{x} \in \Omega \ \mid -\varepsilon \leq (\boldsymbol{x} - \boldsymbol{x}_{\mathcal{S}}) \cdot \boldsymbol{n} \leq \varepsilon. \tag{2.10}$$

The interpolant for the body force $\varphi_{\boldsymbol{b}}$ can be obtained from φ_ρ for the typical gravitational force.

The interpolants φ_ρ and φ_μ are called compatible if they satisfy (2.9) *and*

(2.10) *for all times.*

The continuous approximation (2.6)–(2.7) induces a segregated solution approach for evolving the explicit representation of the interface $\mathcal{S}(t_0)$ given at a particular instant in time, t_0. The equations are solved for $\boldsymbol{v}(\boldsymbol{x}, t_0)$ and $p(\boldsymbol{x}, t_0)$ using a previous value for the mass density, dynamic viscosity, body force, interfacial force, and velocity fields; in addition to boundary conditions. Once the new velocity and pressure fields are computed, the interface is updated according to

$$d_t \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{v}\big(\boldsymbol{x}_{\mathcal{S}}, t_0\big) \quad \forall\ \boldsymbol{x}_{\mathcal{S}} \in \mathcal{S}(t_0). \tag{2.11}$$

The new locus of $\mathcal{S}(t_1)$ produces the fields $\rho(\boldsymbol{x}, t_1)$, $\mu(\boldsymbol{x}, t_1)$, $\boldsymbol{b}(\boldsymbol{x}, t_1)$, and $\boldsymbol{f}(\boldsymbol{x}, t_1)$ by virtue of (2.7) and (2.5) which can be used, in the next iteration, to solve for the new velocity and pressure fields at the new time instant t_1.

2.2 Numerical Methods

This section elaborates on some details of the numerical algorithms and underlying mathematical approximation used in our method to solve the equations described in Sec. 2.1. The approach consists of a loop over two main tasks:

- Update of the position of the interface with the newly computed interfacial velocity field.

- Calculation of the velocity and pressure fields in a fixed domain with

given mass density, viscosity, interfacial position and momentum source.

The first task consists of an explicit time-step for advancing the position of the interface. We use the front-tracking method discussed in Sec. 1.1.1 to represent and evolve a mathematical surface. An updated locally grid based algorithm in [7] is used for resolving complicated topological changes in the contactor simulation.

We choose the Immersed Boundary Method (IBM) referenced in Sec. 1.1.2 for numerically discretizing the continuous approximated equations of motion given in Sec. 2.1.2. Then by applying the Projection Method for the solution of the one-phase Navier-Stokes equations with variable mass density, dynamic viscosity, interfacial force and body force fields we resolve the second task.

2.2.1 LGB front-tracking method

As mentioned at the beginning of Sec. 2.2, we use the locally grid based front-tracking method (LGB) referred in Sec. 1.1.1 to update the surface which undergoes topological changes. Our goal is to get a topological valid surface at the end of each time step. Using T to denote the set of triangles on a surface S, we have the following assumptions for a topological valid surface S:

(A) For any $t \in T$, there is one and only one neighboring triangle on each edge of t.

(B) For any $t \in T$ and vertex $\boldsymbol{p} \in t$, the triangle set $\{\hat{t} | \hat{t} \in T, \boldsymbol{p} \in \hat{t}\}$ forms only one triangle list around $\boldsymbol{p}$.

(C) The surface constructed by T is on orientable surface.

The surface S represents a 2D-manifold under the first two assumptions and the orientation of each triangle could be defined under the third assumption. Fig. 2.1 shows two examples of the invalid cases. In the left figure, assumption (A) fails while in the right figure, assumption (B) is violated.

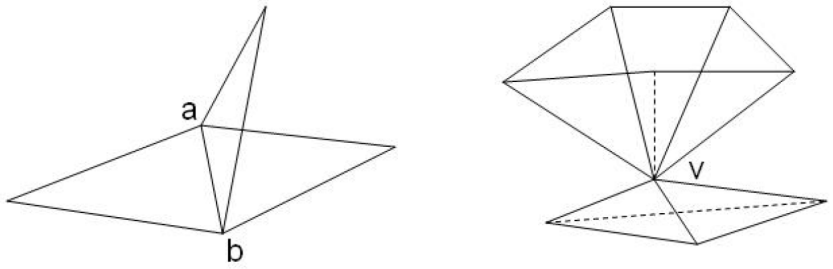

Figure 2.1: Example of non-manifold surfaces

For each time step of the surface propagation, three major steps are included:

1. Interface point propagation with parallel communication.

2. Surface redistribution for better mesh quality.

3. Locally Grid Based untangling for resolving self intersection.

Given the velocity field, the first step is to advance interface through the propagation of the discretized interface points via the simple ordinary equation (2.11) solved by a first order Euler forward scheme. After the point propagation, we use a key routine scatter_front() to point-to-point communicate interface between different processors which is described in detail later in this section.

With a user-specified frequency, the second step of the algorithm would redistribute the surface to maintain the quality of triangles. Two major components of surface redistribution are splitting long edges (Fig. 2.2) in the middle and contracting small edges to the middle (Fig. 2.3). We note that when contracting edges, a non-manifold surface could be generated and a fix of topology would be called. Details for the topology fixes are given in [6]. The interface communication is also called after this step to ensure consistency of interface on each processor.

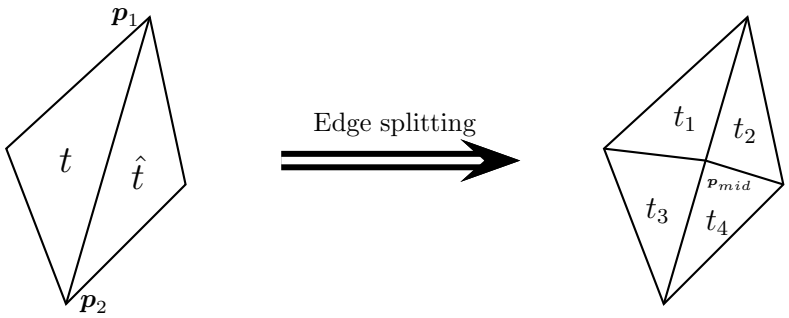

Figure 2.2: Edge splitting operation

For the third step we identify some bad regions of the propagated surface, isolate these, and preserve the intersections of the surface with the grid cell edges to allow a Grid Based reconstruction locally near the bad region. Triangles neighboring the bad region are removed, and so there is a gap, separating the good part of the interface from the reconstruction of the bad part of the interface. We re-seal this gap by adding triangles in a way that keeps the mesh topologically valid. The intersection detection algorithm is approximate in that it will miss bifurcations totally internal to a single mesh cell. In Chapter 6 we improve the algorithm by replacing it with an exact triangle-triangle intersection detection algorithm. Readers are referred to [6] for details of the LGB reconstruction. Fig. 2.4 shows the entire algorithm for our interface propagation.

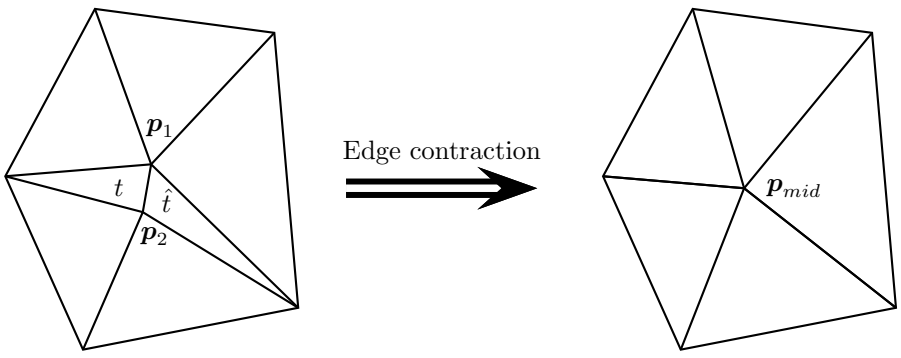

Figure 2.3: Edge contraction operation

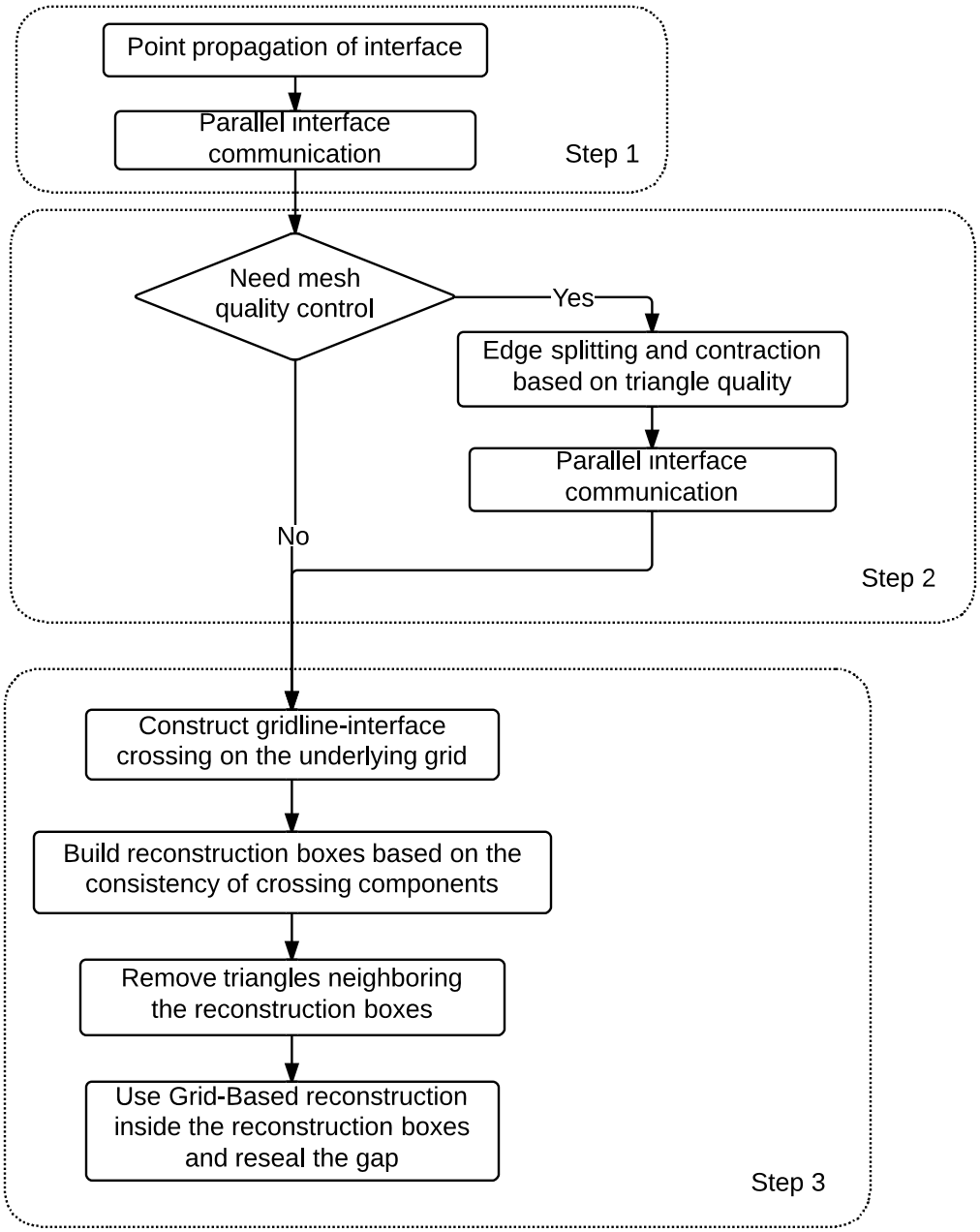

Figure 2.4: Flow chart for the LGB interface propagation algorithm

At the end of this section we give a brief description of the key routine that we use for interface communication. For processor 1 and processor 2, which are neighboring processors in direction k, we denote the surface on processor 1 to be S_1 with triangle set T_1, the computational domain for processor 1 to be D_1, the surface on processor 2 to be S_2 with triangle set T_2 and the computational domain for processor 2 to be D_2. Let the computational boundary between processor 1 and processor 2 be $B_{12} = D_1 \cap D_2$ and set $\hat{T_1}$ and $\hat{T_2}$ to be:

$$\hat{T_1} = \{t | t \in T_1, t \cap B_{12} \neq \phi\}, \quad \hat{T_2} = \{t | t \in T_2, t \cap B_{12} \neq \phi\}.$$

When $\hat{T_1} = \hat{T_2}$ scatter_front() sends $\tilde{T_1}$ from processor 1 to processor 2 and send $\tilde{T_2}$ from processor 2 to processor 1 where $\tilde{T_1}$ and $\tilde{T_2}$ are interior triangles with in certain distance with B_{12}:

$$\tilde{T_1} = \{t | t \in T_{1,} \exists \boldsymbol{p}_i \in t, \boldsymbol{p}_i \in D_1, |\boldsymbol{p}_i - B_{12}| \leq 4\Delta h\},$$
$$\tilde{T_2} = \{t | t \in T_{2,} \exists \boldsymbol{p}_i \in t, \boldsymbol{p}_i \in D_2, |\boldsymbol{p}_i - B_{12}| \leq 4\Delta h\},$$

where Δh is the cell length in direction k. Otherwise scatter_front() would report an error and print triangles $\hat{T_1}$ and $\hat{T_2}$ as diagnostics. The implementation assumes a distributed memory computer. The low level communications use the message passing interface (MPI) standard.

2.2.2 Immersed Boundary method

In this section we describe some numerical details for using front tracking method combined with the IB method which is similar to [74]. With the interface represented by a topologically valid triangular mesh defined in Sec. 2.2.1, the interpolant (2.7) for mass density and viscosity is built with the aid of a Heaviside function [44]

$$H(\phi) = \begin{cases} 0, & \phi < -\varepsilon, \\ \dfrac{1}{2} + \dfrac{\phi}{2\varepsilon} + \dfrac{1}{2\pi}\sin\left(\dfrac{\pi\phi}{\varepsilon}\right), & -\varepsilon \leq \phi \leq \varepsilon, \\ 1, & \phi > \varepsilon\ , \end{cases} \tag{2.12}$$

where $\phi := (\boldsymbol{x} - \boldsymbol{x}_{\mathcal{S}}) \cdot \boldsymbol{n}_{\overrightarrow{\beta\alpha}}$ is the distance to the interface in the direction of the surface normal. We compute the density and viscosity close to the interface as

$$\rho(\phi) = \rho_\beta + (\rho_\alpha - \rho_\beta)\, H(\phi), \tag{2.13}$$

$$\mu(\phi) = \mu_\beta + (\mu_\alpha - \mu_\beta)\, H(\phi). \tag{2.14}$$

For approximating the evaluation of the volume interfacial force (2.5) we use a discrete delta function formulation defined in [36]. That is, for each grid cell in Ω, I_{ijk}, the interfacial force per unit of volume given by the IB method at the cell center equals to

$$\boldsymbol{F}_{ijk} = \sum_l D(\boldsymbol{x}_{ijk} - \boldsymbol{x}^{(l)})\, \boldsymbol{f}^{(l)}, \tag{2.15}$$

where $\boldsymbol{x}_{ijk}$ and $\boldsymbol{x}^{(l)}$ are the position of the grid cell center and the centroid of the front triangular element with index l, respectively. The discrete delta function in (2.5) is approximated by $D(\cdot)$, above, and

$$\boldsymbol{f}^{(l)} := 2\mathcal{H}^{(l)}\sigma\boldsymbol{n}^{(l)}\Delta s^{(l)} \tag{2.16}$$

is the interfacial force on the lth interface triangle, $\mathcal{H}^{(l)}$ is the mean curvature of the interface at the centroid of the corresponding triangle, $\boldsymbol{n}^{(l)}$ is the unit normal of the triangle, and $\Delta s^{(l)}$ is the surface area of the associated triangle. Here the mean curvature for each triangle $\mathcal{H}^{(l)}$ is the average of the mean curvatures on the three vertices which are calculated by a polynomial fitting given by [31] with a second-order accuracy. The polynomial fitting will be discussed seperately in detail in Sec. 3.1 as it also serves as the basis for the high-order parallel mesh algorithms that will be introduced in later chapters.

In this work the particular discrete delta function used to compute the volumetric interfacial force (2.15) has the form

$$D(\boldsymbol{x}_{ijk} - \boldsymbol{x}^{(l)}) = \frac{\delta_h\big(\frac{x_1^{ijk}-x_1^{(l)}}{h_1}\big)\,\delta_h\big(\frac{x_2^{ijk}-x_2^{(l)}}{h_2}\big)\,\delta_h\big(\frac{x_3^{ijk}-x_3^{(l)}}{h_3}\big)}{V(I_{ijk})}, \tag{2.17}$$

where $V(I_{ijk})$ is the volume of cell I_{ijk} and the function δ_h is given by [36]:

$$\delta_h(r) = \begin{cases} \dfrac{3-2|r|+\sqrt{1+4|r|-4r^2}}{8}, & \text{if} \quad |r| < 1, \\ \dfrac{5-2|r|-\sqrt{-7+12|r|-4r^2}}{8}, & \text{if} \quad 1 \leqslant |r| < 2, \\ 0, & \text{if} \quad 2 \leqslant |r|. \end{cases} \tag{2.18}$$

When implementing formula (2.15), we select a triangle on the interface, choose the cells within a radius of three cell diameters around the triangle, and distribute the interfacial force onto the cells using the discrete delta function (2.17). In this way the interfacial force is approximated by a volume integral

$$\int_{\Omega} 2\mathcal{H}\sigma\boldsymbol{n}\,\delta(\boldsymbol{x}-\boldsymbol{x}_{\mathcal{S}})\,dv = \int_{\mathcal{S}} 2\mathcal{H}\sigma\boldsymbol{n}\,ds = \int_{\Omega} \boldsymbol{F}_{ijk}\,dv. \tag{2.19}$$

The Courant-Friedrichs-Lewy (CFL) condition related to interfacial tension from [8] is implemented as a requirement of stability

$$\Delta t_1 < \left[\frac{\langle\rho\rangle(\Delta\ell)^3}{2\pi\sigma}\right]^{1/2}, \tag{2.20}$$

where $\langle\rho\rangle$ is the average density, $\Delta\ell$ is the smallest cell edge and σ is the interfacial tension coefficient. Together with the convective time step restriction

$$\Delta t_2 \leq \frac{1}{\frac{|u|_{\max}}{\Delta x} + \frac{|v|_{\max}}{\Delta y} + \frac{|w|_{\max}}{\Delta z}}, \tag{2.21}$$

where $|u|_{\max}$, $|v|_{\max}$ and $|w|_{\max}$ are the maximum Cartesian velocity components magnitude, the final time step follows

$$\Delta t = C_{\mathrm{CFL}} \times \min\{\Delta t_1, \Delta t_2\},$$

where C_{CFL} is the CFL number (less than 1).

2.2.3 Projection method

The IB method smoothes the density and viscosity and spreads the singular force into a strip. Thus at each step, the incompressible Navier-Stokes equations with variable density and viscosity need to be solved. In rectangular coordinate, these equations become

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{2.22}$$

$$u_t + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{p_x}{\rho} + \frac{(2\mu u_x)_x + (\mu(u_y + v_x))_y}{\rho} + F_x + G_x, \tag{2.23}$$

$$v_t + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{p_y}{\rho} + \frac{(\mu(u_y + v_x))_x + (2\mu v_y)_y}{\rho} + F_y + G_y. \tag{2.24}$$

In 2D and in 3D they take the form

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{2.25}$$

$$\begin{aligned} &u_t + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \\ &= -\frac{p_x}{\rho} + \frac{(2\mu u_x)_x + (\mu(u_y + v_x))_y + (\mu(u_z + w_x))_z}{\rho} + F_x + G_x, \end{aligned} \tag{2.26}$$

$$\begin{aligned} &v_t + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \\ &= -\frac{p_y}{\rho} + \frac{(\mu(u_y + v_x))_x + (2\mu v_y)_y + (\mu(v_z + w_y))_z}{\rho} + F_y + G_y, \end{aligned} \tag{2.27}$$

$$\begin{aligned} &w_t + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \\ &= -\frac{p_z}{\rho} + \frac{(\mu(u_z + w_x))_x + (\mu(v_z + w_y))_y + (2\mu w_z)_z}{\rho} + F_z + G_z. \end{aligned} \tag{2.28}$$

Transforming into cylindrical coordinates, we have

$$\frac{\partial u_z}{\partial z} + \frac{\partial u_r}{\partial r} + \frac{1}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{u_r}{r} = 0, \tag{2.29}$$

$$\begin{aligned}
&\frac{\partial u_\theta}{\partial t} + \frac{\partial (u_z u_\theta)}{\partial z} + \frac{\partial (u_r u_\theta)}{\partial r} + \frac{1}{r}\frac{\partial (u_\theta u_\theta)}{\partial \theta} + 2\frac{u_r u_\theta}{r} \\
&= \frac{1}{\rho}\left[-\frac{1}{r}\frac{\partial p}{\partial \theta} + \frac{\partial d_{z\theta}}{\partial z} + \frac{\partial d_{r\theta}}{\partial r} + \frac{1}{r}\frac{\partial d_{\theta\theta}}{\partial \theta} + 2\frac{d_{r\theta}}{r}\right] + F_\theta + G_\theta,
\end{aligned} \tag{2.30}$$

$$\begin{aligned}
&\frac{\partial u_z}{\partial t} + \frac{\partial (u_z u_z)}{\partial z} + \frac{\partial (u_r u_z)}{\partial r} + \frac{1}{r}\frac{\partial (u_\theta u_z)}{\partial \theta} + \frac{u_r u_z}{r} \\
&= \frac{1}{\rho}\left[-\frac{\partial p}{\partial z} + \frac{\partial d_{zz}}{\partial z} + \frac{\partial d_{rz}}{\partial r} + \frac{1}{r}\frac{\partial d_{\theta z}}{\partial \theta} + \frac{d_{rz}}{r}\right] + F_z + G_z,
\end{aligned} \tag{2.31}$$

$$\begin{aligned}
&\frac{\partial u_r}{\partial t} + \frac{\partial (u_z u_r)}{\partial z} + \frac{\partial (u_r u_r)}{\partial r} + \frac{1}{r}\frac{\partial (u_\theta u_r)}{\partial \theta} + \frac{(u_r u_r - u_\theta u_\theta)}{r} \\
&= \frac{1}{\rho}\left[-\frac{\partial p}{\partial r} + \frac{\partial d_{zr}}{\partial z} + \frac{\partial d_{rr}}{\partial r} + \frac{1}{r}\frac{\partial d_{\theta r}}{\partial \theta} + \frac{(d_{rr} - d_{\theta\theta})}{r}\right] + F_r + G_r,
\end{aligned} \tag{2.32}$$

where the components of the viscous tensor, $d_{ij}(i, j = \theta, z, r)$ are given by:

$$\begin{aligned}
&d_{zz} = 2\mu\left(\frac{\partial u_z}{\partial z}\right),\ d_{rr} = 2\mu\left(\frac{\partial u_r}{\partial r}\right),\ d_{\theta\theta} = 2\mu\left(\frac{1}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{u_r}{r}\right), \\
&d_{zr} = d_{rz} = \mu\left(\frac{\partial u_r}{\partial z} + \frac{\partial u_z}{\partial r}\right), \\
&d_{z\theta} = d_{\theta z} = \mu(\left(\frac{\partial u_\theta}{\partial z} + \frac{1}{r}\frac{\partial u_z}{\partial \theta}\right), \\
&d_{r\theta} = d_{\theta r} = \mu\left(\frac{\partial u_\theta}{\partial r} - \frac{u_\theta}{r} + \frac{1}{r}\frac{\partial u_r}{\partial \theta}\right),
\end{aligned}$$

with μ as the dynamic viscosity. Note that the continuous approximation of the governing equations of motion (2.6)–(2.7) calls for a continuously varying dynamical viscosity. Thus the above equations for different directions are

coupled together for two-phase simulations and divergence of the deviatoric part of the stress tensor, $\nabla \cdot \mu(\boldsymbol{x}, t) \left(\nabla_{\boldsymbol{x}} \boldsymbol{v} + \nabla_{\boldsymbol{x}} \boldsymbol{v}^T \right)$ degenerates to $\mu \Delta_{\boldsymbol{x}} \boldsymbol{v}$ when μ is a constant.

The projection method, which is a fractional step method using some intermediate velocity is the mostly well known method for solving these equations. Since the pioneering work of Chorin [11, 12], many high-order projection methods were derived by Bell *et.al* [4, 5], Kim and Moin [35], and Van Kan [33]. All the projection methods have three steps: First we obtain an intermediate velocity field $\mathbf{u}^*$ by solving a convection-diffusion equation with proper boundary conditions. Then we perform the projection to enforce the velocity field to satisfy the divergence free condition for incompressible flow. This step requires the solution of an elliptic equation. Lastly we update the pressure. More details can be found in [9].

We follow the projection method advanced in [4, 5], which is second-order accurate in velocity for single-phase incompressible fluid flow and known as PMI [9]. This method is only first-order accurate in pressure, however the alternate method used here, PMII, is second-order accurate in pressure provided appropriate boundary conditions are used.

We use a second-order Godunov scheme to solve the nonlinear advection term. This algorithm generates a source term in the diffusion equation. These two steps are solved in a coupled manner to achieve higher order time accuracy. The implicit second-order time integration Crank-Nicholson scheme is used to solve the diffusion equation. The Laplacian equation in the projection step is solved by a standard five-point discretization. Both the linear systems formed

in the diffusion and projection step are sparse matrices, solved in parallel using PETSc [2].

We use the GMRES linear solver for the diffusion step and the BiCGSL linear solver for the projection step to obtain a faster convergence rate. However, the BiCGSL algorithm is not theoretically stable and occasionally it does not converge, in which case we use GMRES instead after a large residual has been observed for the iterative linear solver.

Chapter 3

High-Order Surface Reconstruction and Propagation

3.1 Local Polynomial Fitting

For calculating the mean curvature and normal for each point used in (2.16), the high-order local polynomial fitting with local orthogonal projection parameterization given by [31] is used. This polynomial fitting also serves as the foundation for both the high-order mesh propagation algorithm which will be introduced in the next section, and the high-order mesh optimization algorithm discussed in later chapters.

Specifically, in a rectangular Cartesian coordinate system, for each surface point $\boldsymbol{x}_0$ on $\mathcal{S}$, a local parameterization for the neighboring points is established based on the approximated unit normal $\hat{\mathbf{m}}_0$. The polynomial fitting may be defined either over a local uvw coordinate system, where the uv-plane is approximately parallel with the tangent plane of the surface at $\boldsymbol{x}_0$ and the neighboring points transformed into the form $[u, v, h(u, v)]$, or more generally,

overthe global xyz coordinate system with any parameterization. $h(u, v)$ is known as the local height function.

For either local height function or the coordinate function for a parametric surface, given a set of points $\{\boldsymbol{x}_i\}$ with parameterizations $\{\mathbf{u}_i = (u_i, v_i)\}$ and function values $\{f_i\}$ for $i = 1, \ldots, m-1$ sampled from the neighboring points of $\boldsymbol{x}_0$, a set of approximate equations are generated from a Taylor series expansion with degree of fitting d:

$$f_i \approx \sum_{p=0}^{d} \sum_{j,k \geq 0}^{j+k=p} c_{jk} \frac{u_i^j \, v_i^k}{j!k!}, \qquad i = 0, \ldots, m-1. \tag{3.1}$$

We write the equations in compact form

$$\mathbf{V}\mathbf{c} \approx \mathbf{f}, \tag{3.2}$$

where $\mathbf{V}$ is a $m \times n$ matrix and $n = (d+1)(d+2)/2$.

Equation (3.2) can be solved using weighted linear least squares to minimize a weighted norm:

$$\min_{\mathbf{c}} ||\mathbf{V}\mathbf{c} - \mathbf{f}||_{\mathbf{W}} = \min_{\mathbf{c}} ||\mathbf{W}(\mathbf{V}\mathbf{c} - \mathbf{f})||_2, \tag{3.3}$$

which is equivalent to a new linear least squares problem:

$$\widetilde{\mathbf{V}}\mathbf{c} \approx \mathbf{b}, \quad \text{where} \quad \widetilde{\mathbf{V}} = \mathbf{W}\mathbf{V} \quad \text{and} \quad \mathbf{b} = \mathbf{W}\mathbf{f}. \tag{3.4}$$

$\mathbf{W}$ is a $m \times m$ diagonal weighting matrix. Let $\hat{\mathbf{m}}_i$ denote the approximate

unit normal on point $\mathbf{x}_i$. We choose the weight at ith vertex as

$$w_i = \frac{\gamma_i^+}{(||\mathbf{u}_i||^2/h + \epsilon)^{d/2}}, \tag{3.5}$$

where $\gamma_i^+ = \max(0, \hat{\mathbf{m}}_i{}^T \hat{\mathbf{m}}_0)$, $h = \sum_{i=1}^m ||\mathbf{u}_i||^2/m$, and $\epsilon \approx 0.01$. The numerator of this weight serves as a safeguard for non-smooth areas where normals change dramatically. The denominator prevents the weights from becoming too large at points too close to $\boldsymbol{x}_0$.

For $\widetilde{\mathbf{V}}$ with large condition numbers, a diagonal scaling matrix $\mathbf{S}$ with ith element $\mathbf{S}_{ii} = 1/||\tilde{\mathbf{v}}_i||_2$ is applied to improve the conditioning of the system. With a rescaled weighted system, a QR factorization

$$\mathbf{WVS} = \mathbf{QR}$$

is used to solve for $\mathbf{c}$. The major benefit for using QR factorization is that we can reduce the degree of fitting by simply removing the last few columns in $\mathbf{Q}$ and corresponding rows and columns in $\mathbf{R}$ depending on the condition number of the upper-triangular matrix $\mathbf{R}$. Let $\widetilde{\mathbf{Q}}$ and $\widetilde{\mathbf{R}}$ denote the reduced matrices of $\mathbf{Q}$ and $\mathbf{R}$, the final solution of $\mathbf{c}$ is given by

$$\mathbf{c} = \mathbf{S}\widetilde{\mathbf{R}}^{-1}\widetilde{\mathbf{Q}}^T\mathbf{b}, \tag{3.6}$$

where $\widetilde{\mathbf{R}}^{-1}$ denotes a back substitution step.

Returning to the normal and curvature calculation involved in (2.16), the

local coordinate system prompts the orthogonal transformation matrix

$$\mathcal{Q} := \left[\begin{array}{ccc} \boldsymbol{t}_1 & \boldsymbol{t}_2 & \boldsymbol{m} \end{array} \right],$$

where $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ are the unit vectors, in the global coordinate system x, y, z, along the positive direction of the u and v axes while $\boldsymbol{m} := \boldsymbol{t}_1 \times \boldsymbol{t}_2$ is the unit vector along the positive w direction. With the local polynomial fitting coefficients c_{jk} calculated for height function, the local unit normal $\hat{\boldsymbol{n}}$ and mean curvature $\mathcal{H}$ can be computed [31]. The normal unit vector in the global rectangular Cartesian coordinate system is computed by the transformation $\boldsymbol{n} = \mathcal{Q} \cdot \hat{\boldsymbol{n}}$ while the mean curvature $\mathcal{H}$ is an invariant of the curvature tensor therefore independent of the coordinate system.

For the cylindrical coordinate system used in the contactor simulation, a set of points $\{(r_i, \theta_i, z_i)\}$ is selected around the point $\boldsymbol{x}_0 := (r_0, \theta_0, z_0)$, and transformed into the x, y, z coordinate system

$$x_i = r_i \cos\theta_i, \quad y_i = r_i \sin\theta_i, \quad z_i = z_i,$$

which are used as the input points for the polynomial fitting. Hence, by transforming the normal vector (n_x, n_y, n_z), obtained from the method described above, in cylindrical coordinates (n_r, n_θ, n_z) via

$$n_\theta = -n_x \sin\theta_0 + n_y \cos\theta_0, \;\; n_r = n_x \cos\theta_0 + n_y \sin\theta_0, \;\; n_z = n_z,$$

the accuracy of the surface normal and mean curvature calculation is preserved.

3.2 High-Order Functional Interface Propagation

The classical front tracking formulation has been based on a point propagation, where each point is propagated by solving an ODE separately. No high-order differential quantity is used or preserved in the point propagation thus we need to calculate the normals and curvatures using the technique introduced in Sec. 3.1 after point propatation in each time step. In this section, we propose a new framework for functional interface propagation, which can deliver higher-order accuracy for both point positions and higher-order differential quantities such as normal and curvature.

3.2.1 Mathematical description

In a point propagation, each point on the interface is propagated by solving an ODE

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{\psi}(\boldsymbol{x}, t), \tag{3.7}$$

where $\boldsymbol{\psi}$ is the velocity and may depend on surface normal, curvatures, etc. The disadvantage of point propagation is that it omits the interaction of different points on the interface. As a result, it does not consider discontinuities in its formulation. The face offsetting method [29] considers the interactions of interface but only at a discrete level, so it was limited to only second-order accuracy.

We propose a *functional propagation* formulation for front tracking. In particular, in the case of surface propagation, consider a local parameterization

of the surface

$$\boldsymbol{x}(\boldsymbol{u}; t) : \mathcal{U} \times \mathbb{R} \to \mathbb{R}^3,$$

where $\mathcal{U} \subset \mathbb{R}^2$. In other words, interface propagation is the evolution of the function $\boldsymbol{x}(\boldsymbol{u})$ over time. This leads to a differential equation

$$\frac{\partial \boldsymbol{x}(\boldsymbol{u}; t)}{\partial t} = \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}, t); t), \tag{3.8}$$

where the velocity is a function of the geometry and potentially other time-dependent variables. As we will show in the following, this formulation is more general and will allow more accurate approximations.

3.2.2 Semi-discrete form of functional propagation

Let us derive a semi-discrete form of functional propagation, which discretizes the equation in space but not yet in time. At a point $\boldsymbol{x}_0(t) \equiv \boldsymbol{x}(\boldsymbol{u}_0, t)$ on the interface, suppose we have a stencil $\boldsymbol{X}(t) = \{\boldsymbol{x}_k \equiv \boldsymbol{x}(\boldsymbol{u}_k, t)\}$ around $\boldsymbol{x}_0$ on the interface. In general, $\boldsymbol{x}_0$ is a point in $\boldsymbol{X}$, so we number the points in $\boldsymbol{X}$ from 0 to $|\boldsymbol{X}| - 1$. We also refer to the set $\{\boldsymbol{u}_k\}$ as the stencil around $\boldsymbol{u}_0$ in the parametric space. Using $\boldsymbol{u}_k$ is more convenient than $\boldsymbol{x}_k$ for computational purpose, because $\boldsymbol{u}_k$ is not a function of time, in the sense that $\boldsymbol{u}_k$ does not change within a single time step from t_j to t_{j+1} in the fully discrete setting (although one can use a different local parameterizations from time step t_{j+1} to t_{j+2}).

Substituting $\boldsymbol{u}_k$ into (3.8), we obtain an equation

$$\frac{\partial \boldsymbol{x}(\boldsymbol{u}_k;t)}{\partial t} = \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k,t);t) \tag{3.9}$$

for each point $\boldsymbol{u}_k$ in the stencil of $\boldsymbol{u}_0$. Our goal now is to use these $|\boldsymbol{X}|$ equations to obtain a local approximation to $\boldsymbol{x}(\boldsymbol{u},t)$ around $\boldsymbol{u}_0$. Consider the Taylor series expansion of $\boldsymbol{x}$ about a point $\boldsymbol{u}_0$,

$$\boldsymbol{x}(\boldsymbol{u};t) = \boldsymbol{x}(\boldsymbol{u}_0;t) + \left(\boldsymbol{\nabla}_{\boldsymbol{u}}\boldsymbol{x}(\boldsymbol{u}_0;t)\right)\delta\boldsymbol{u} + \frac{1}{2}(\delta\boldsymbol{u})^T\left(\boldsymbol{H}_{\boldsymbol{u}}\boldsymbol{x}(\boldsymbol{u}_0;t)\right)\delta\boldsymbol{u} + O(\|\delta\boldsymbol{u}\|^3), \tag{3.10}$$

where $\boldsymbol{\nabla}_{\boldsymbol{u}}\boldsymbol{x} = \begin{bmatrix} x_u & x_v \\ y_u & y_v \\ z_u & z_v \end{bmatrix}$ and $\boldsymbol{H}_{\boldsymbol{u}}\boldsymbol{x}$ denote a rank-3 tensor containing the Hessian of $\boldsymbol{x}$ with respect to $\boldsymbol{u}$. We can construct fourth-order accurate semi-discrete form by taking into account third-order derivatives. Let $\mathcal{M}(\boldsymbol{u})$ denote the monomial basis of $\boldsymbol{u} = [u,v]^T$, i.e.,

$$\mathcal{M}(\boldsymbol{u}) = [1, u, v, u^2, uv, v^2]^T.$$

We can then express (3.10) as a

$$\boldsymbol{x}(\boldsymbol{u};t) = \boldsymbol{K}^T(t)\mathcal{M}(\delta\boldsymbol{u}),$$

where $\boldsymbol{K}(t) \in \mathbb{R}^{6\times 3}$ corresponds to the coefficients in the Taylor series expansion of the three components of $\boldsymbol{x}(\boldsymbol{u};t)$, respectively. In particular, the first

columns is

$$\boldsymbol{K}_{:,1}(t) = \left[x_0, \frac{\partial x(\boldsymbol{u}_0)}{\partial u}, \frac{\partial x(\boldsymbol{u}_0)}{\partial v}, \frac{1}{2}\frac{\partial^2 x(\boldsymbol{u}_0)}{\partial u^2}, \frac{\partial^2 x(\boldsymbol{u}_0)}{\partial u \partial v}, \frac{1}{2}\frac{\partial^2 x(\boldsymbol{u}_0)}{\partial v^2}\right]^T,$$

and the second and third columns correspond to y and z, correspondingly. If the velocity $\boldsymbol{\psi}$ is smooth, we obtain an approximation to (3.9) with third-order truncation error

$$\frac{d\boldsymbol{K}^T(t)}{dt}\mathcal{M}(\delta \boldsymbol{u}_k) = \boldsymbol{\psi}\left(\boldsymbol{K}^T(t)\mathcal{M}(\delta \boldsymbol{u}_k); t\right) + O(\|\delta \boldsymbol{u}\|^3). \tag{3.11}$$

This gives three equations per point in the stencil, and we obtain a system of $m = 3|\boldsymbol{X}|$ ODEs for 18 unknowns in $\boldsymbol{K}(t)$. If $|\boldsymbol{X}| \geq 6$, we then obtain a rectangular system for $\boldsymbol{K}(t)$, which can then be solved as a (weighted) least squares problem. Note that in point propagation, we solve this equation for only the first row of $\boldsymbol{K}$ with $\boldsymbol{x}_0$ as the stencil.

3.2.3 More compact stencil

In (3.11), we need at least six points in the stencil and the number of points required to be used is normally larger than six. This is because we are using only (3.8) for the propagation of $\boldsymbol{x}(t)$ (which corresponds to the first row of $\boldsymbol{K}(t)$) to determine all the coefficients in $\boldsymbol{K}(t)$. In the following, we derive an additional equation to propagate the coefficients corresponding to the first-derivatives $\partial \boldsymbol{x}/\partial \boldsymbol{u}$, which are calculated directly from the result of functional propagation so that we will have more equations per point in the stencil and in turn reduce the required size of the stencil. A compact stencil

is more robust for polynomial fitting and also reduces the number of layers of ghosts needed to be exchanged in a parallel computation.

Differentiating both sides of (3.9) with respect to $\boldsymbol{u}$, we obtain

$$\boldsymbol{\nabla}_{\boldsymbol{u}} \frac{\partial \boldsymbol{x}(\boldsymbol{u_k}; t)}{\partial t} = \boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u_k}, t); t).$$

In the left-hand side, we can change the order of differentiation if $\boldsymbol{x}$ is smooth, and hence

$$\frac{\partial \left(\boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{x}(\boldsymbol{u_k}; t)\right)}{\partial t} = \boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u_k}, t); t).$$

Note that $\boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{x}(\boldsymbol{u}; t)$ corresponds to the surface normal at point $\boldsymbol{u_k}$. More specifically, given $\boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{x}(\boldsymbol{u}_k; t)$, the surface normal is uniquely defined, and given the surface normal and a parameterization, $\boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{x}(\boldsymbol{u}_k; t)$ is also uniquely defined. Therefore, this equation corresponds to the propagation of the normal to the interface.

Using the Taylor series of the derivatives, we then obtain

$$\boldsymbol{\nabla}_{\boldsymbol{u}} \boldsymbol{x}(\boldsymbol{u}_k; t) = \boldsymbol{K}^T(t) \boldsymbol{\nabla}_{\boldsymbol{u}} \mathcal{M}(\delta \boldsymbol{u}_k) + O(\|\delta \boldsymbol{u}\|^2),$$

where

$$\boldsymbol{\nabla}_{\boldsymbol{u}} \mathcal{M}(\delta \boldsymbol{u}_k) = \left[\begin{array}{cccccc} 0 & 1 & 0 & 2u & v & 0 \\ 0 & 0 & 1 & 0 & u & 2v \end{array} \right]^T_{\boldsymbol{u}=\delta \boldsymbol{u}_k}.$$

To ensure the same length scale as (3.11), we introduce a normalization matrix $\boldsymbol{C} = \text{diag}(\max\{|\delta u_k|\}, \max\{|\delta u_k|\}, \max\{|\delta u_k|\})$ and obtain a set of six new

equations

$$\boldsymbol{C}\frac{d\boldsymbol{K}^T(t)}{dt}\boldsymbol{\nabla}_{\boldsymbol{u}}\mathcal{M}(\delta\boldsymbol{u}_k) = \boldsymbol{C}\boldsymbol{\nabla}_{\boldsymbol{u}}\boldsymbol{\psi}\left(\boldsymbol{K}^T(t)\mathcal{M}(\delta\boldsymbol{u}_k);t\right) + O(\|\delta\boldsymbol{u}\|^3). \qquad (3.12)$$

Equations (3.11) and (3.12) define nine equations per point in the stencil, which are linearly independent if all the first three rows of $\boldsymbol{K}(t)$ are computed and propagated over time. Therefore, we now need as few as two points in the stencil to achieve third-order accuracy. For better robustness, it is more advantageous to use three or four points in the stencil. In the numerical tests we use one ring of neighboring points as the stencil.

When the local parameterization is changed, we need to define a transformation of $\boldsymbol{K}(t)$ between different coordinate systems. This is done by first converting $\boldsymbol{K}_{1,:}(t)$ into points and converting $\boldsymbol{K}_{2:3,:}(t)$ into surface normals. Next we transform the points and normals in the coordinate system, and then convert points and normals back to coefficients $\boldsymbol{K}_{1:3,:}(t)$.

In the right-hand side, if $\boldsymbol{\psi}$ is defined over $\mathbb{R}^3$ instead of only on the interface, we can apply the chain rule to differentiate it with respect to $\boldsymbol{x}$ and obtain

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{u}}\psi(\boldsymbol{x}(\boldsymbol{u}_k;t),t) &= \frac{\partial\boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k;t),t)}{\partial\boldsymbol{x}}\left(\boldsymbol{\nabla}_{\boldsymbol{u}}\boldsymbol{x}(\boldsymbol{u}_k;t)\right)\\ &= \frac{\partial\boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k;t),t)}{\partial\boldsymbol{x}}\left(\boldsymbol{K}^T(t)\boldsymbol{\nabla}_{\boldsymbol{u}}\mathcal{M}(\delta\boldsymbol{u}_k)\right).\end{aligned}$$

3.2.4 Explicit time discretization

For a first-order explicit time stepping, from (3.11) and (3.12) we form a rectangular linear system solved using a weighted least squares formulation at a point $\boldsymbol{x}(\boldsymbol{u}_0)$ and neighboring points $\boldsymbol{x}(\boldsymbol{u}_k), k = 1, \ldots, |\boldsymbol{X}| - 1$.

$$\boldsymbol{K}^T(t_{n+1})\mathcal{M}(\delta\boldsymbol{u}_k) = \boldsymbol{x}(\boldsymbol{u}_k; t_n) + \delta t\boldsymbol{\psi}\left(\boldsymbol{x}(\boldsymbol{u}_k; t_n); t_n\right), \tag{3.13}$$

$$\begin{aligned}\boldsymbol{K}^T(t_{n+1})\boldsymbol{\nabla}_{\boldsymbol{u}}\mathcal{M}(\delta\boldsymbol{u}_k) = \boldsymbol{K}^T(t_n)\boldsymbol{\nabla}_{\boldsymbol{u}}\mathcal{M}(\delta\boldsymbol{u}_k) \\ + \delta t\frac{\partial\boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n), t_n)}{\partial\boldsymbol{x}}\left(\boldsymbol{K}^T(t_n)\boldsymbol{\nabla}_{\boldsymbol{u}}\mathcal{M}(\delta\boldsymbol{u}_k)\right).\end{aligned} \tag{3.14}$$

From the equations (3.11) and (3.12) and the first order discretization in time t. We derive that the new point positions has a $O(h^3) + O(\Delta t)$ error and the first derivative which corresponding to normals has a $O(h^2) + O(\Delta t)$ error, where h is the average edge length of the input triangular mesh and Δt is the time step.

For higher-order time stepping, we can use the four-stage Runge-Kutta scheme. Note the solutions of different points affect each other. Therefore, within each stage we need to propagate all the points before we can advance to the next stage, similar to using Runge-Kutta for a system of ODEs.

Using the same notation as above, we have:

$$\begin{aligned}\boldsymbol{A}_1 &= \delta t\boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n); t_n), \\ \boldsymbol{B}_1 &= \delta t\frac{\partial\boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n), t_n)}{\partial\boldsymbol{x}}\left(\boldsymbol{K}^T(t_n)\boldsymbol{\nabla}_{\boldsymbol{u}}\mathcal{M}(\delta\boldsymbol{u}_k)\right),\end{aligned}$$

$$
\begin{aligned}
\boldsymbol{A}_2 &= \delta t \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n) + \frac{\boldsymbol{A}_1}{2}; t_n + \frac{\delta t}{2}), \\
\boldsymbol{B}_2 &= \delta t \frac{\partial \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n) + \frac{\boldsymbol{A}_1}{2}, t_n + \frac{\delta t}{2})}{\partial \boldsymbol{x}} \left(\boldsymbol{K}^T(t_n) \boldsymbol{\nabla}_{\boldsymbol{u}} \mathcal{M}(\delta \boldsymbol{u}_k) + \frac{\boldsymbol{B}_1}{2} \right), \\
\boldsymbol{A}_3 &= \delta t \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n) + \frac{\boldsymbol{A}_2}{2}; t_n + \frac{\delta t}{2}), \\
\boldsymbol{B}_3 &= \delta t \frac{\partial \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n) + \frac{\boldsymbol{A}_2}{2}, t_n + \frac{\delta t}{2})}{\partial \boldsymbol{x}} \left(\boldsymbol{K}^T(t_n) \boldsymbol{\nabla}_{\boldsymbol{u}} \mathcal{M}(\delta \boldsymbol{u}_k) + \frac{\boldsymbol{B}_2}{2} \right), \\
\boldsymbol{A}_4 &= \delta t \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n) + \boldsymbol{A}_3); t_n + \delta t), \\
\boldsymbol{B}_4 &= \delta t \frac{\partial \boldsymbol{\psi}(\boldsymbol{x}(\boldsymbol{u}_k; t_n) + \boldsymbol{A}_3, t_n + \delta t)}{\partial \boldsymbol{x}} \left(\boldsymbol{K}^T(t_n) \boldsymbol{\nabla}_{\boldsymbol{u}} \mathcal{M}(\delta \boldsymbol{u}_k) + \boldsymbol{B}_3 \right).
\end{aligned}
$$

and the linear system is constructed as:

$$
\boldsymbol{K}^T(t_{n+1}) \mathcal{M}(\delta \boldsymbol{u}_k) = \boldsymbol{x}(\boldsymbol{u}_k; t_n) + \frac{\boldsymbol{A}_1 + 2\boldsymbol{A}_2 + 2\boldsymbol{A}_3 + \boldsymbol{A}_4}{6}, \tag{3.15}
$$

$$
\boldsymbol{K}^T(t_{n+1}) \boldsymbol{\nabla}_{\boldsymbol{u}} \mathcal{M}(\delta \boldsymbol{u}_k) = \boldsymbol{K}^T(t_n) \boldsymbol{\nabla}_{\boldsymbol{u}} \mathcal{M}(\delta \boldsymbol{u}_k) + \frac{\boldsymbol{B}_1 + 2\boldsymbol{B}_2 + 2\boldsymbol{B}_3 + \boldsymbol{B}_4}{6}. \tag{3.16}
$$

Again from the equations (3.11) and (3.12) and the fourth order discretization in time t. We derive that the new point positions has a $O(h^3) + O(\Delta t^4)$ error and the first derivative which corresponding to normals has a $O(h^2) + O(\Delta t^4)$ error, where h is the edge length of the input triangular mesh and Δt is the time step. Using the same argument we could also derive easily that with a higher degree of fitting d in the Taylor series expansion (3.1) and a k-th order time stepping, the point positions converges with error $O(h^{d+1}) + O(\Delta t^k)$.

Chapter 4

Verification and Validation of One-Phase Flow

In this chapter we consider the order of accuracy of the projection method for solving the single-phase incompressible Navier-Stokes equations and the verification/validation studies for Taylor-Couette flow in different flow regimes.

4.1 Verication of One-Phase Flow by the Method of Manufactured Solutions

As mentioned in Sec. 2.2.3, the method tested here uses PMII which is a second-order projection method for single-phase, incompressible fluid flow. We use initial and boundary data and associated source terms for which the exact solution is known. This method is applied in 2D and 3D Cartesian coordinates and in 3D cylindrical coordinates. The exact solution and data for the 2D solution we use here were employed previously [9]. The 3D exact solution comes from [19] and a modification is needed for the case of 3D cylindrical coordinates.

4.1.1 Time-dependent solutions in 2D Cartesian coordinates

With the domain $\Omega = [0,1] \times [0,1]$ and parameters $\rho = 1$ and $\mu = 1$, the Navier-Stokes equations are augmented with a forcing term so that the functions

$$u = \cos\Big(2\pi\big(x - \omega(t)\big)\Big)\,(3y^2 - 2y) \tag{4.1}$$

$$v = 2\pi \sin\Big(2\pi\big(x - \omega(t)\big)\Big)\,y^2\,(y-1) \tag{4.2}$$

$$\begin{aligned} p = &- \frac{\omega'(t)}{2\pi} \sin\Big(2\pi\big(x - \omega(t)\big)\Big)\,\Big(\sin(2\pi y) - 2\pi y + \pi\Big) \\ &- \cos\Big(2\pi\big(x - \omega(t)\big)\Big)\,\big(-2\sin(2\pi y) + 2\pi y\big) - \pi\big)\ , \end{aligned} \tag{4.3}$$

with $\omega(t) = 1 + \sin(2\pi t^2)$, become an exact solution [9]. The boundary condition for the x direction is periodic and Dirichlet for the y direction.

The required source terms are derived by substituting (4.1)–(4.3) into (2.22)–(2.24). We derive the formulas for the source term using Maple with C code generated directly from symbolic computation. A uniform time step of $\Delta t = 0.4h$ was used, corresponding to a CFL number of 0.4. The errors are calculated in both L_1 and L_∞ norms at time 0.5 for $N \times N$ grids with N equal to 10, 20, 40 and 80. The errors and order of accuracy for the u and v component of the velocity field and the pressure p are shown in Tables 4.1–4.3. The results are very close to the theoretically predicted second-order accuracy.

Table 4.1: Verification of convergence rate of the u-component of the velocity for 2D NSE in Cartesian coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.02505001	NA	0.05749996	NA
20×20	0.00596152	2.0711	0.02271585	1.3399
40×40	0.00138930	2.1013	0.00620064	1.8732
80×80	0.00033850	2.0371	0.00159809	1.9561

Table 4.2: Verification of convergence rate of the v-component of the velocity for 2D NSE in Cartesian coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.02187048	NA	0.04523046	NA
20×20	0.00706664	1.8890	0.01307569	1.6299
40×40	0.00179580	1.9671	0.00334820	1.9764
80×80	0.00045103	1.9879	0.00084932	1.9933

Table 4.3: Verification of convergence rate of the pressure p for 2D NSE in Cartesian coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.20987610	NA	0.91315993	NA
20×20	0.04786609	2.1325	0.29326793	1.6391
40×40	0.00970739	2.3018	0.06626968	2.1458
80×80	0.00219919	2.1421	0.01504217	2.1393

4.1.2 Time-dependent solutions in 3D Cartesian coordinates

We set the domain $\Omega = [-1,1] \times [-1,1] \times [-1,1]$, and choose parameters $\rho = 1$ and $\mu = 1$. The Navier-Stokes equations are augmented with a forcing term in order to the solution be [19]:

$$u = -a\big[e^{ax}\sin(ay+dz) + e^{az}\cos(ax+dy)\big]e^{-d^2t} \tag{4.4}$$

$$v = -a\big[e^{ay}\sin(az+dx) + e^{ax}\cos(ay+dz)\big]e^{-d^2t} \tag{4.5}$$

$$w = -a\big[e^{az}\sin(ax+dy) + e^{ay}\cos(az+dx)\big]e^{-d^2t} \tag{4.6}$$

$$\begin{aligned} p = -\frac{a^2}{2}\big[&e^{2ax} + e^{2ay} + e^{2az} + 2\sin(ax+dy)\cos(az+dx)e^{a(y+z)} \\ &+ 2\sin(ay+dz)\cos(ax+dy)e^{a(z+x)} \\ &+ 2\sin(az+dx)\cos(ay+dz)e^{a(x+y)}\big]e^{-2d^2t}\ . \end{aligned} \tag{4.7}$$

We set $a = 1$ and $d = 1$, with Dirichlet boundary conditions on all six boundaries.

As in the 2D test, we derive the source term using Maple from (2.25)–(2.28). The maximum flow speed of the solution is about 10, so a uniform time step of $\Delta t = 0.05h$ is used, corresponding to a CFL number of 0.5 to make the algorithm stable. The errors are calculated in both L_1 and L_∞ norms at time 0.2 for $N \times N \times N$ grids with N equal to 10, 20, 40 and 80. In Table 4.4, we omit the v and w velocity components as their convergence rates are similar to that of the u velocity component.

We observe in Table 4.5 that the L_∞ norm error of pressure p is only first-

Table 4.4: Verification of convergence rate of u-component velocity for 3D NSE in Cartesian coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.04540364	NA	0.01354348	NA
20×20	0.01177651	1.9469	0.00373252	1.8594
40×40	0.00299105	1.9772	0.00127001	1.5553
80×80	0.00075277	1.9904	0.00038637	1.7168

Table 4.5: Verification of convergence rate of pressure p for 3D NSE in Cartesian coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.56668074	NA	0.57665525	NA
20×20	0.14954512	1.9220	0.53674946	0.1035
40×40	0.03779002	1.9845	0.34788539	0.6256
80×80	0.00947893	1.9952	0.19797179	0.8133

order accurate when using Dirichlet boundary condition for all boundaries. The normal mode analysis of [9] for second-order accuracy in pressure requires the boundary conditions to be periodic except in one direction. Since the 3-D exact solution we used does not satisfy this requirement, the obtained reduced order of accuracy in the L_∞ norm appears to be justified; this is not a verified assertion.

4.1.3 Time-dependent solutions in 3D cylindrical coordinates

We generalize the second-order projection method [4, 5] to cylindrical coordinates for equations (2.29)-(2.32). The manufactured solution in cylindrical

coordinate is

$$u_\theta(r,\theta,z) = \frac{1}{12}\big(-r^2\sin\theta + r^2\cos\theta - z^2\sin\theta + z^2\cos\theta\big)e^{-t} \tag{4.8}$$

$$u_z(r,\theta,z) = -\frac{1}{6}zr\big(\cos\theta + \sin\theta\big)e^{-t} \tag{4.9}$$

$$u_r(r,\theta,z) = -\frac{1}{12}\big(r^2\cos\theta + r^2\sin\theta + z^2\cos\theta + z^2\sin\theta\big)e^{-t} \tag{4.10}$$

$$p(r,\theta,z) = \frac{1}{12}(r^2+z^2)e^{-t}\ . \tag{4.11}$$

We set the computational domain to be $\theta \in [0, 2\pi], z \in [-1, 1], r \in [1, 2]$ and the other parameters as in the 3D manufactured solution in Cartesian coordinates. The source term was derived by substituting the exact solution into the incompressible Navier-Stokes equations in cylindrical coordinates (2.29)–(2.32). The results, summarized in Tables 4.6–4.9, are very close to the theoretically predicted convergence rates.

Table 4.6: Verification of convergence rate of u_θ for 3D NSE in cylindrical coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.21461366	NA	0.03974838	NA
20×20	0.05242220	2.0335	0.01018509	1.9643
40×40	0.01297461	2.0145	0.00256724	1.9882
80×80	0.00323912	2.0020	0.00064171	2.0002

Again we see here that the convergence rate of the L_∞ norm error for pressure is first-order accurate rather than second-order when we refine the

Table 4.7: Verification of convergence rate of u_z for 3D NSE in cylindrical coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.01729539	NA	0.00352590	NA
20×20	0.00483825	1.8378	0.00135969	1.3747
40×40	0.00123408	1.9710	0.00035601	1.9333
80×80	0.00031296	2.0020	0.00011209	1.6673

Table 4.8: Verification of convergence rate of u_r for 3D NSE in cylindrical coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.01569499	NA	0.00346419	NA
20×20	0.00427474	1.8764	0.00144748	1.2590
40×40	0.00109416	1.9660	0.00053794	1.4280
80×80	0.00028311	1.9504	0.00017048	1.6553

Table 4.9: Verification of convergence rate of p for 3D NSE in cylindrical coordinates.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	7.41387743	NA	0.76635043	NA
20×20	1.78645195	2.0531	0.19345287	1.9860
40×40	0.43545378	2.0365	0.05218918	1.8902
80×80	0.10759368	2.0169	0.02173881	1.2535

mesh because of the Dirichlet boundary condition on all six boundaries; not a verified assertion.

4.2 Verication and Validation of One-Phase Annular Couette Flow

The Taylor-Couette flow regime occurs in the annulus between differentially rotating concentric cylinders, most often with the inner cylinder rotating and the outer cylinder fixed. The flow becomes unstable when the rotation rate exceeds a critical value (Fig. 4.1) [47] and a secondary motion in the form of vortices appears in the cross section of the primary flow. The critical rotation rate is different depending on whether the inner or the outer cylinder is moving. Since the pioneering work of [71], in which linear stability analysis was used to predict the appearance of Taylor vortices, the Taylor-Couette flow has been studied extensively, theoretically, experimentally and computationally.

Taylor-Couette flow is important not only because of its applications, such as the design of a viscometer, fluid mixture apparatus, and in biology; it also has been studied to observe flow patterns during the transition from laminar to turbulent flow. With increasing Reynolds number, the flow undergoes a series of transitions from circular Couette flow, to axially periodic Taylor vortex flow, to a state with time-dependent waves imposed on the vortices (wavy Taylor vortex flow) and to chaotic and turbulent Taylor vortex flow. A visualization of Taylor vortex flow is displayed in Fig. 4.1 [47].

Section 4.2.1 develops the analytic solution for the annular Couette flow

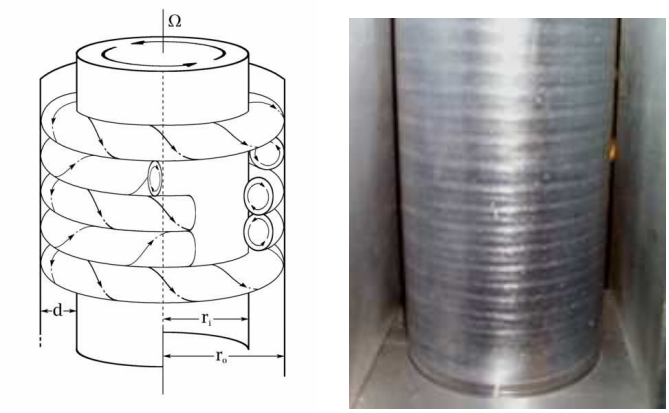

Figure 4.1: Left: Schematic diagram of counter-rotating axisymmetric vortices of Taylor-Couette flow (©2000, Mike Minbiole and Richard M. Lueptow). Right: Axisymmetric Taylor vortices visualized using titanium dioxide-coated mica flakes (©2002, Alp Akonur and Richard M. Lueptow)

used for code verification. Section 4.2.2 presents simulation results for the transition from circular Couette flow regime to the Taylor-Couette flow regime. The critical value for this transition is predicted from 3-D simulations as demonstrated in Section 4.2.4, while fully turbulent flow is studied in Section 4.2.5.

The dimensionless parameters which characterize the system are: the radii ratio $\eta = R_\mathrm{i}/R_\mathrm{o}$ of the inner to the outer cylinders respectively; the cylinder height to the gap width aspect ratio $\Gamma := \dfrac{L_z}{R_\mathrm{o} - R_\mathrm{i}}$, where the gap width between cylinders is also denoted $d := R_\mathrm{o} - R_\mathrm{i}$; the Reynolds number Re; and the Taylor number T. The definitions for the last two parameters vary in the literature, here we use

$$Re := \frac{\Omega_\mathrm{i} R_\mathrm{i}(R_\mathrm{o} - R_\mathrm{i})}{\nu}, \text{ and} \tag{4.12}$$

$$T := 4Re^2 \left(\frac{1-\eta}{1+\eta}\right) \ , \tag{4.13}$$

where Ω_i is the rotation speed of the inner cylinder and ν is the kinematic viscosity of the fluid.

4.2.1 Steady-state 1-D Couette flow in an annular sector

For a sub-critical Reynolds number, Couette flow is laminar. Under this assumption, the Navier-Stokes equations simplify to one-dimensional flow. In cylindrical coordinates, we derived an exact solution for an infinite cylinder ($\Gamma = \infty$),

$$v_\theta = C_1 r + \frac{C_2}{r} \tag{4.14}$$

$$v_z = 0; \tag{4.15}$$

$$v_r = 0; \tag{4.16}$$

$$P = \rho\Big[\frac{C_1^2(r^2 - R_\mathrm{i}^2)}{2} + 2C_1C_2 \ln\frac{r}{R_\mathrm{i}} - \frac{C_2^2}{2}\big(\frac{1}{r^2} - \frac{1}{R_\mathrm{i}^2}\big)\Big] + P_0 \ , \tag{4.17}$$

with

$$C_1 = \frac{\Omega_\mathrm{i} R_\mathrm{i}^2}{R_\mathrm{i}^2 - R_\mathrm{o}^2}$$

$$C_2 = -C_1 R_\mathrm{o}^2$$

$$P_0 = P(R_\mathrm{i}) \ .$$

This solution was used as a verification and convergence rate test for the numerical solution method of the *three-dimensional* Navier-Stokes equations. For this purpose, we used periodic boundary conditions in the z and θ directions and no slip conditions on the boundaries in the r direction. We set the computational domain to be $\theta \in [0, 0.628]$ rad, $z \in [0, 0.628]$ cm, $r \in [2.538, 3.166]$ cm, and $\Omega_{\mathrm{i}} = 0.05$ rad/s (0.48 RPM). A time-dependent, three-dimensional simulation was performed using $\rho = 1.0$ g/cm^3 and $\nu = 0.0089$ cm^2/s which resulted in $Re = 9.0$. The initial velocity and pressure fields for the simulation were enforced using the foregoing exact solution fields. The simulation results obtained for a small Re compared well with the exact solution (4.14)–(4.17) obtained for $Re = 0$. Errors were calculated in both L_1 and L_∞ norms at time 1 s (0.008 rotations) for $N \times N \times N$ grids with N equal to 10, 20, 40 and 80. As $u_z = 0$ within round off error, we omit analysis of this variable and display convergence results for u_θ, u_r and the pressure p in Tables 4.10–4.12.

Table 4.10: Verification of the convergence rate of u_θ for 1D Couette flow.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	8.9689427e-06	NA	4.2889586e-05	NA
20×20	2.3392513e-06	1.9445	1.2213023e-05	1.8122
40×40	5.8993089e-07	1.9874	3.2289907e-06	1.9193
80×80	1.4774364e-07	1.9974	8.2839859e-07	1.9627

All convergence results confirm a second-order accurate solution algorithm.

Table 4.11: Verification of the convergence rate of u_r for 1D Couette flow.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	3.2098794e-06	NA	1.3955543e-05	NA
20×20	2.3019156e-07	3.8016	3.2426953e-06	2.1056
40×40	1.4560125e-08	3.9827	4.0839172e-07	2.9892
80×80	9.1318711e-10	3.9950	5.0681518e-08	3.0104

Table 4.12: Verification of the convergence rate of p for 1D Couette flow.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	4.0876435e-06	NA	2.2784380e-05	NA
20×20	7.9287009e-07	2.3661	4.6255377e-06	2.3004
40×40	1.9027897e-07	2.0590	1.2303281e-06	1.9106
80×80	4.6547912e-08	2.0313	3.2019695e-07	1.9420

4.2.2 Transition from 1-D to 3-D Taylor vortices

A qualitative demonstration of the existence of the vortex transition is made in this section at the Reynolds number $Re = 2686$. The simulation parameters were $R_\mathrm{i} = 2.538$ cm, $R_\mathrm{o} = 3.166$ cm, $\eta = R_\mathrm{i}/R_\mathrm{o} = 0.8$, $\Omega_\mathrm{i} = 15$ rad/s (or 38.07 cm/s, or 143.2 RPM), $\rho = 1.0$ g/cm^3, $\nu = 0.0089$ cm^2/s and

$$Re = \frac{\Omega_\mathrm{i} R_\mathrm{i}(R_\mathrm{o} - R_\mathrm{i})}{\nu} = 2686.3, \quad T = 4Re^2 \left(\frac{1-\eta}{1+\eta}\right) = 3.2 \times 10^6.$$

The critical Reynolds number for the transition to vortex flow is 94.7 [57], so this simulation is well above the transition point. The computational domain

is $\theta \in [0, 0.628]$ rad, $z \in [0, 1.256]$ cm, $r \in [2.538, 3.166]$ cm. We used periodic boundary conditions for both the θ and z directions and a no slip boundary condition for the velocity in the r direction. The initialization of the velocity and pressure fields were as in (4.14)–(4.17), that is, the unperturbed laminar solution for one-phase Couette flow. A simulation up to $t = 2s$ was performed (about 5 rotations of the inner cylinder). The grid size for this simulation is $40 \times 80 \times 40$. Streamlines of the v_r and v_z velocity component fields indicate the formation of the Taylor vortices (Fig. 4.2) which is consistent with the linear stability theory (Sec. 4.2.3).

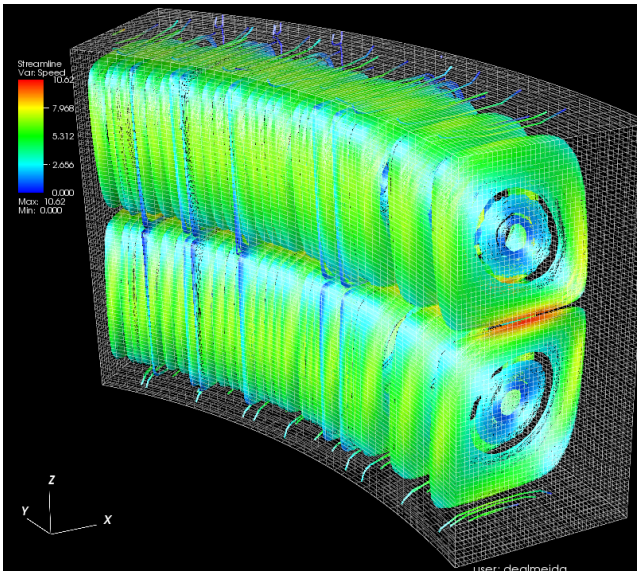

Figure 4.2: Computed velocity streamlines in the r–z plane of a Taylor vortex flow at $Re = 2686.3$ (*i.e.* $T = 3.2 \times 10^6$).

4.2.3 Axisymmetric linear stability analysis of Couette flow

Before verifying the critical value for the transition from laminar Couette flow to Taylor vortex flow we summarize the classical linear stability analysis of small perturbations for getting a theoretical critical Reynolds number. Infinitesimal disturbances are applied to the Couette flow in the domain

$$R_1 < r < R_2\ ,\ 0 \leq \theta < 2\pi \text{ and } -\infty < z < \infty, \tag{4.18}$$

with an aspect ratio $\Gamma = \infty$.

The velocity field for the Couette flow in cylindrical coordinates is given by

$$v_r = v_z = 0\ ,\ v_\theta = V(r) = Ar + \frac{B}{r}\ ,\ \frac{\partial p}{\partial r} = \rho\frac{V^2(r)}{r}, \tag{4.19}$$

where ρ is the density and

$$A = \frac{R_2^2\Omega_2 - R_1^2\Omega_1}{R_2^2 - R_1^2} = -\Omega_1\frac{\eta^2 - \mu}{1 - \eta^2}, \tag{4.20}$$

$$B = -\frac{R_1^2R_2^2(\Omega_2 - \Omega_1)}{R_2^2 - R_1^2} = \Omega_1R_1^2\frac{1 - \mu}{1 - \eta^2} \tag{4.21}$$

$$\mu = \frac{\Omega_2}{\Omega_1}\ ,\ \eta = \frac{R_1}{R_2}. \tag{4.22}$$

The Couette flow can be characterized by the parameters μ, η, and Reynolds number. Here the values for the parameters are

$$\Omega_2 = 0\ ,\ \mu = 0\ ,\ \eta = \frac{R_1}{R_2} = \frac{2.538}{3.166} = 0.8. \tag{4.23}$$

Let u', v', w' and p' denote the perturbations of v_r, v_θ, v_z and p for Couette flow. Substituting $v_r \to v_r + u' = u',\ v_\theta \to v_\theta + v' = V + v',\ v_z \to v_z + w' = w'$ and $p \to p + p'$ in the Navier-Stokes equations, we get

$$\begin{aligned} r:\ & \frac{\partial u'}{\partial t} + \frac{V}{r}\frac{\partial u'}{\partial \theta} - \frac{2V}{r}v' + \frac{1}{\rho}\frac{\partial p'}{\partial r} - \nu\left(\nabla^2 u' - \frac{2}{r^2}\frac{\partial v'}{\partial \theta} - \frac{u'}{r^2}\right) \\ & = -\left(u'\frac{\partial u'}{\partial r} + \frac{v'}{r}\frac{\partial u'}{\partial \theta} + w'\frac{\partial u'}{\partial z}\right) + \frac{v'^2}{r} \end{aligned} \tag{4.24}$$

$$\begin{aligned} \theta:\ & \frac{\partial v'}{\partial t} + \frac{dV}{dr}u' + \frac{V}{r}\frac{\partial v'}{\partial \theta} + \frac{V}{r}u' + \frac{1}{\rho r}\frac{\partial p'}{\partial \theta} - \nu\left(\nabla^2 v' + \frac{2}{r^2}\frac{\partial u'}{\partial \theta} - \frac{v'}{r^2}\right) \\ & = -\left(u'\frac{\partial v'}{\partial r} + \frac{v'}{r}\frac{\partial v'}{\partial \theta} + w'\frac{\partial v'}{\partial z}\right) - \frac{v'u'}{r} \end{aligned} \tag{4.25}$$

$$z:\ \frac{\partial w'}{\partial t} + \frac{V}{r}\frac{\partial w'}{\partial \theta} + \frac{1}{\rho}\frac{\partial p'}{\partial z} - \nu\nabla^2 w' = -\left(u'\frac{\partial w'}{\partial r} + \frac{v'}{r}\frac{\partial w'}{\partial \theta} + w'\frac{\partial w'}{\partial z}\right)\ . \tag{4.26}$$

Similarly for the equation of continuity

$$\frac{\partial u'}{\partial r} + \frac{u'}{r} + \frac{1}{r}\frac{\partial v'}{\partial \theta} + \frac{\partial w'}{\partial z} = 0\ . \tag{4.27}$$

The no slip condition at the inner and outer cylinder gives the boundary conditions

$$u' = v' = w' = 0 \text{ at } r = R_1 \text{ and } r = R_2\ . \tag{4.28}$$

We consider infinitesimal disturbances that are axisymmetric, *i.e.* independent of θ, and periodic in the axial direction. Linearizing the equations (4.24), (4.25), (4.26) and (4.27) and taking into account that all the disturbances are

independent of θ we obtain

$$\frac{\partial u'}{\partial t} - \frac{2V}{r}v' = -\frac{1}{\rho}\frac{\partial p'}{\partial r} + \nu\left(\nabla^2 u' - \frac{u'}{r^2}\right) \tag{4.29}$$

$$\frac{\partial v'}{\partial t} + \left(\frac{dV}{dr} + \frac{V}{r}\right)u' = \nu\left(\nabla^2 v' - \frac{v'}{r^2}\right) \tag{4.30}$$

$$\frac{\partial w'}{\partial t} = -\frac{1}{\rho}\frac{\partial p'}{\partial z} + \nu\nabla^2 w' \tag{4.31}$$

$$\frac{\partial u'}{\partial r} + \frac{u'}{r} + \frac{\partial w'}{\partial z} = 0\ , \tag{4.32}$$

where $\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{\partial^2}{\partial z^2}$.

The normal mode solutions for the disturbances have the form

$$u' = u(r)e^{\beta t + i\lambda z},\ v' = v(r)e^{\beta t + i\lambda z},\ w' = w(r)e^{\beta t + i\lambda z},\ p' = P(r)e^{\beta t + i\lambda z}. \tag{4.33}$$

In these disturbances β is the growth rate and λ is the wavenumber in the axial direction. Substituting these expressions in (4.29), (4.30), (4.31) and (4.32) we get

$$\nu\left(\frac{d^2}{dr^2} + \frac{1}{r}\frac{d}{dr} - \frac{1}{r^2} - \lambda^2 - \frac{\beta}{\nu}\right)u = \frac{1}{\rho}\frac{dP}{dr} - \frac{2V}{r}v \tag{4.34}$$

$$\nu\left(\frac{d^2}{dr^2} + \frac{1}{r}\frac{d}{dr} - \frac{1}{r^2} - \lambda^2 - \frac{\beta}{\nu}\right)v = 2Au \tag{4.35}$$

$$\nu\left(\frac{d^2}{dr^2} + \frac{1}{r}\frac{d}{dr} - \lambda^2 - \frac{\beta}{\nu}\right)w = \frac{1}{\rho}P(i\lambda z) \tag{4.36}$$

$$\frac{du}{dr} + \frac{u}{r} + i\lambda w = 0\ . \tag{4.37}$$

Finally, eliminating P and w we obtain

$$\nu\left(\frac{d^2}{dr^2}+\frac{1}{r}\frac{d}{dr}-\frac{1}{r^2}-\lambda^2-\frac{\beta}{\nu}\right)\left(\frac{d^2}{dr^2}+\frac{1}{r}\frac{d}{dr}-\frac{1}{r^2}-\lambda^2\right)u=2\lambda^2\Omega(r)v, \tag{4.38}$$

$$\nu\left(\frac{d^2}{dr^2}+\frac{1}{r}\frac{d}{dr}-\frac{1}{r^2}-\lambda^2-\frac{\beta}{\nu}\right)v=2Au, \tag{4.39}$$

where $\Omega(r)=\frac{V}{r}$. From (4.28) we get at $r=R_1$ and $r=R_2$

$$u=v=0\ . \tag{4.40}$$

From equation of continuity we get at $r=R_1$ and $r=R_2$

$$\frac{du}{dr}=0\ . \tag{4.41}$$

Using the transformations [57]:

$$\begin{aligned}
&r=R_0+dx,\ R_0=\frac{R_1+R_2}{2}\\
&\delta=\frac{d}{R_0},\ \xi(x)=\frac{1}{1+\delta x}\\
&\Omega=\Omega_1 G(x),\ G(x)=\frac{1}{1-\eta^2}\left[\mu-\eta^2+4\xi^2(x)\eta^2\frac{1-\mu}{(1+\eta)^2}\right]\\
&\sigma=\frac{\beta d^2}{\nu},\ \alpha=\lambda d,\ T=-\frac{4A\Omega_1 d^4}{\nu^2}\\
&D=\frac{d}{dx},\ D^*=D+\delta\xi(x)\ .
\end{aligned} \tag{4.42}$$

(4.43)

on equations (4.38) and (4.39), we obtain

$$(DD^* - \alpha^2 - \sigma)(DD^* - \alpha^2)u_{11} = -\alpha^2 TG(x, \mu, \eta)v_{11}, \tag{4.44}$$

$$(DD^* - \alpha^2 - \sigma) = u_{11} \ . \tag{4.45}$$

with the boundary conditions

$$u_{11} = v_{11} = Du_{11} = 0 \text{ at } x = \pm\frac{1}{2} \ . \tag{4.46}$$

Equations (4.44)-(4.46) form an eigenvalue problem in two parameters along with the boundary conditions. The flow can be described in terms of the geometry parameter η, the fluid parameters (μ and Taylor's number T) and the disturbance parameters α and σ. It can be shown that for given μ and η there exists a value T_c such that for $T < T_c$, $Re(\sigma) < 0$ for all $\alpha > 0$ and for $T > T_c$, $Re(\sigma) > 0$ for a band of wave numbers ($\alpha_- < \alpha < \alpha_+$). This defines a set of points in T, α space which is called the neutral curve (see Fig. 4.3) . Clearly the minimum point of the neutral curve is the T_c below which all disturbances of any wavenumber decay with time and above which there is a band of wave numbers for which the disturbances grow exponentially with time. Table 4.13 shows the critical Reynolds and wave numbers for different radius ratios η [10, 57, 75, 60, 67]. At marginal stability the growth rate of the critical wavenumber is zero ($\sigma = 0$). For a supercritical disturbance the growth rate will be positive. From linearized theory, the disturbance grows exponentially with time. However, the disturbances do not show continual amplification with time and instead attains a finite equilibrium amplitude.

Table 4.13: Critical Reynolds number ($\Gamma = \infty, \mu = 0$) of axisymmetric, linear stability analysis.

η	Re_{crit}	k_{crit}
0.975	260.9	3.13
0.90	131.6	3.13
0.80	94.7	3.13
0.70	79.5	3.14
0.60	71.7	3.15
0.50	68.2	3.16

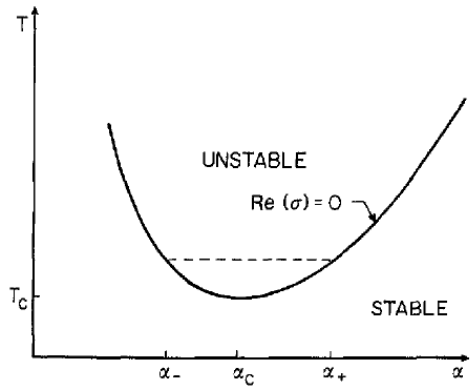

Figure 4.3: Neutral curve of axisymmetric, linear stability analysis [57].

In [15, 78] Davey showed that the growth rate is proportional to the reduced Reynolds number $\varepsilon = 1 - \frac{R_c^2}{Re^2}$. He also calculated the coefficient of the reduced Reynolds number for the growth rate for the wide gap case ($R_2 = 2R_1$) and the narrow gap case ($\eta \to 1$) [15].

4.2.4 Verifying the Couette flow perturbation growth rate

The axisymmetric linear stability analysis of 3-D small perturbations imposed on a 1-D Couette base flow predicts the minimum Reynolds number for the growth of the disturbances for any wave number. The analysis also predicts the growth rate to be time dependent. Therefore in this section we present a verification of the growth of the disturbances (by means of numerical analysis) through a direct time-dependent, 3-D simulation of the evolution of the initial 1-D Couette flow field.

Using (4.14)–(4.17) as the initial conditions for a 3-D flow simulation in the same annular sector used in Sec. 4.2.2, we calculate the growth rate

$$\begin{aligned} K(t) :=& \frac{\sqrt{\langle v_r \rangle^2 + \langle v_z \rangle^2}}{\left| \langle v_\theta \rangle \right|}, \\ \approx& \frac{\sqrt{(u')^2 + (w')^2}}{\left| V + v' \right|}, \\ \approx& \left[e^{\beta t} \frac{e^{i\lambda z} \sqrt{u^2(r) + w^2(r)}}{\left| V + v' \right|} \right], \\ \ln K(t) \approx& \beta t + C(r, \theta, z). \end{aligned}$$

where C is a constant, in time, according to the linear stability analysis assumption (4.33). Thus the slope β of the graph of $\ln K$ versus t gives the growth rate of the fundamental mode for each Reynolds number (Fig. 4.4 and Fig. 4.5).

As indicated in Fig. 4.4 and Fig. 4.5, K has two flat sections and one linearly growing section; the slope is calculated in the interval $10^{-6} < K < 10^{-4}$. The linearly growing section is associated to an exponential growth of the instability $e^{\beta t}$ from linear stability analysis. The first flat section results from the initialization of the instability, which is seeded by numerical grid effects, and not by a perturbation at the maximally growing wave length. The results confirm that at a higher Reynolds number the flow disturbances grow faster. The second flat section corresponds to the approach to the steady state flow with Taylor vortices.

Fig. 4.6 shows the growth rate β as a function of Reynolds number. The growth rate for supercritical disturbances grows with the Reynolds number Re. As $\beta \to 0$ then $Re \to R_c$ and by virtue of a first order Taylor expansion we know from linear stability analysis that $\beta(Re) = c_0(Re - R_c)$ when $Re - R_c$ is small. Thus from a linear fitting to the slope of $\beta(Re)$ we compute c_0 and find R_c setting $\beta = 0$ (inset of fig. 4.6). We obtain the value $R_c = 95.35$, which is very close to the theoretical value of $R_c = 94.7$ [57].

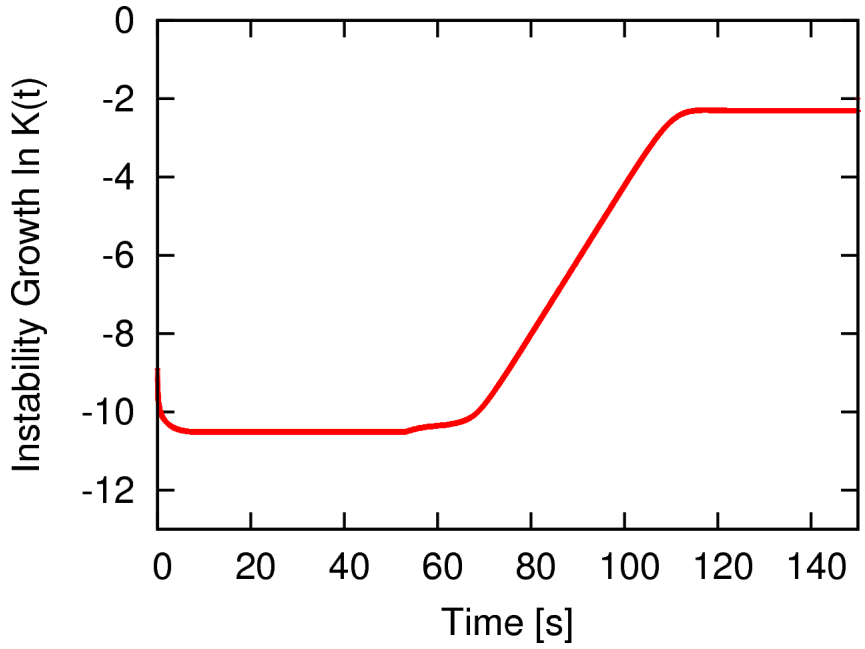

Figure 4.4: Computed instability growth $\ln K$ at $Re = 125$ for a Taylor-Couette flow.

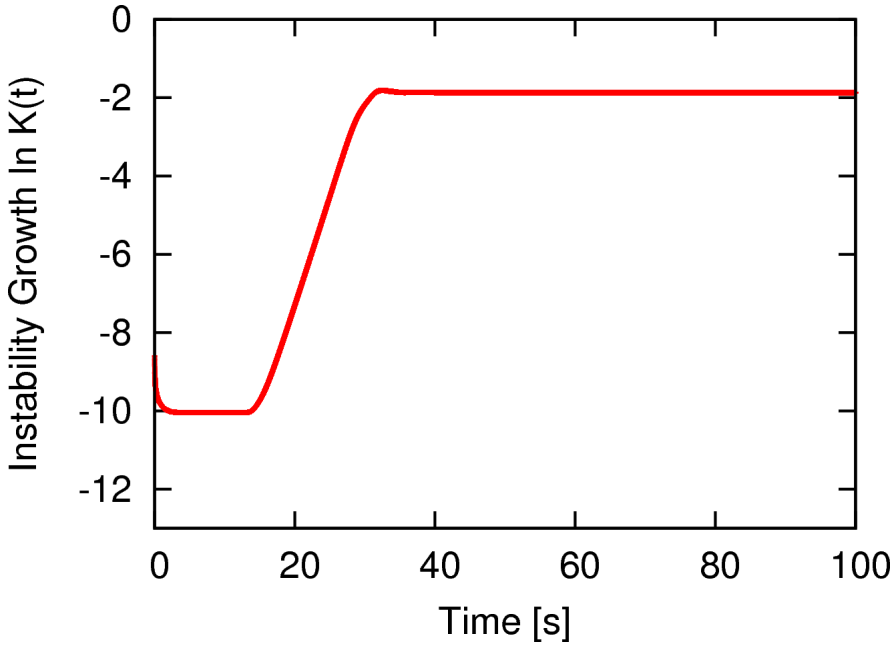

Figure 4.5: Computed instability growth $\ln K$ at $Re = 200$ for a Taylor-Couette flow.

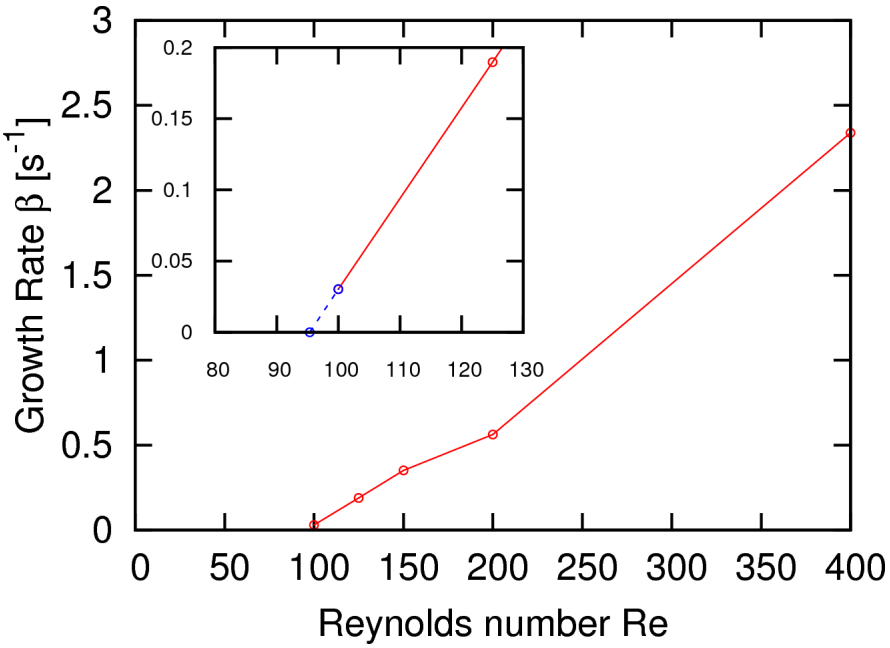

Figure 4.6: Calculated growth rate vs. Reynolds number for a Taylor-Couette flow. Critical Reynolds number $R_c = 95.35$ found by extrapolation.

4.2.5 Turbulent Taylor-Couette flow: verification and validation

In this section a comparison of flow simulation with and without turbulence modeling is made against experimental results and direct numerical simulation of the Navier-Stokes equations for the Taylor-Couette flow at $Re = 8000$. The same geometry as in Sec. 4.2.4 was used. Our results are compared against the experimental results of [66], and the direct numerical simulation (DNS) results of [16].

For turbulent flow simulation, the incompressible, LES-filtered Navier-Stokes equations in cylindrical coordinates that we used and presented below. The low-pass filtered quantities are considered to be mesh cell averages and

denoted with an overbar.

$$\frac{\partial \overline{u}_z}{\partial z} + \frac{\partial \overline{u}_r}{\partial r} + \frac{1}{r}\frac{\partial \overline{u}_\theta}{\partial \theta} + \frac{\overline{u}_r}{r} = 0, \tag{4.47}$$

$$\begin{aligned}
&\frac{\partial \overline{u}_\theta}{\partial t} + \frac{\partial \overline{u}_z \overline{u}_\theta}{\partial z} + \frac{\partial \overline{u}_r \overline{u}_\theta}{\partial r} + \frac{1}{r}\frac{\partial \overline{u}_\theta \overline{u}_\theta}{\partial \theta} + 2\frac{\overline{u}_r \overline{u}_\theta}{r} \\
&= \frac{1}{\overline{\rho}}\left(-\frac{1}{r}\frac{\partial \overline{p}}{\partial \theta} + \frac{1}{r}\frac{\partial \overline{d_{\theta\theta}}}{\partial \theta} + \frac{\partial \overline{d_{z\theta}}}{\partial z} + \frac{\partial \overline{d_{r\theta}}}{\partial r} + 2\frac{\overline{d_{r\theta}}}{r}\right) - \frac{1}{r}\frac{\partial \tau_{\theta\theta}}{\partial \theta} - \frac{\partial \tau_{z\theta}}{\partial z} - \frac{\partial \tau_{r\theta}}{\partial r} \\
&- 2\frac{\tau_{r\theta}}{r} + \overline{f}_\theta,
\end{aligned} \tag{4.48}$$

$$\begin{aligned}
&\frac{\partial \overline{u}_z}{\partial t} + \frac{\partial \overline{u}_z \overline{u}_z}{\partial z} + \frac{\partial \overline{u}_r \overline{u}_z}{\partial r} + \frac{1}{r}\frac{\partial \overline{u}_\theta \overline{u}_z}{\partial \theta} + \frac{\partial \overline{u}_r \overline{u}_z}{r} \\
&= \frac{1}{\overline{\rho}}\left(-\frac{\partial \overline{p}}{\partial z} + \frac{1}{r}\frac{\partial \overline{d_{\theta z}}}{\partial \theta} + \frac{\partial \overline{d_{zz}}}{\partial z} + \frac{\partial \overline{d_{rz}}}{\partial r} + \frac{\overline{d_{rz}}}{r}\right) - \frac{1}{r}\frac{\partial \tau_{\theta z}}{\partial \theta} - \frac{\partial \tau_{zz}}{\partial z} - \frac{\partial \tau_{rz}}{\partial r} \\
&- \frac{\tau_{rz}}{r} + \overline{f}_z,
\end{aligned} \tag{4.49}$$

$$\begin{aligned}
&\frac{\partial \overline{u}_r}{\partial t} + \frac{\partial \overline{u}_z \overline{u}_r}{\partial z} + \frac{\partial \overline{u}_r \overline{u}_r}{\partial r} + \frac{1}{r}\frac{\partial \overline{u}_\theta \overline{u}_r}{\partial \theta} + \frac{\overline{u}_r \overline{u}_r - \overline{u}_\theta \overline{u}_\theta}{r} \\
&= \frac{1}{\overline{\rho}}\left(-\frac{\partial \overline{p}}{\partial r} + \frac{1}{r}\frac{\partial \overline{d_{\theta r}}}{\partial \theta} + \frac{\partial \overline{d_{zr}}}{\partial z} + \frac{\partial \overline{d_{rr}}}{\partial r} + \frac{\overline{d_{rr}} - \overline{d_{\theta\theta}}}{r}\right) - \frac{1}{r}\frac{\partial \tau_{\theta r}}{\partial \theta} - \frac{\partial \tau_{zr}}{\partial z} \\
&- \frac{\partial \tau_{rr}}{\partial r} - \frac{\tau_{rr} - \tau_{\theta\theta}}{r} + \overline{f}_r.
\end{aligned} \tag{4.50}$$

where $\overline{u}_\theta$, $\overline{u}_z$ and $\overline{u}_r$ are the velocity components in the azimuthal, axial and radial directions, respectively, and $\overline{\rho}$ and $\overline{p}$ represent the density and the pressure, and the $\overline{f}_i$ $(i = \theta, z, r)$ denote body forces.

The components of the viscous stress tensor $\overline{d_{ij}}$ $(i, j = \theta, z, r)$ are given by

$$\begin{aligned}
&\overline{d_{\theta\theta}} = 2\overline{\mu}\left(\frac{1}{r}\frac{\partial \overline{u}_\theta}{\partial \theta} + \frac{\overline{u}_r}{r}\right), \quad \overline{d_{zz}} = 2\overline{\mu}\frac{\partial \overline{u}_z}{\partial z}, \quad \overline{d_{rr}} = 2\overline{\mu}\frac{\partial \overline{u}_r}{\partial r}, \\
&\overline{d_{r\theta}} = \overline{\mu}\left(\frac{\partial \overline{u}_\theta}{\partial r} + \frac{1}{r}\frac{\partial \overline{u}_r}{\partial \theta} - \frac{\overline{u}_\theta}{r}\right), \quad \overline{d_{\theta z}} = \overline{\mu}\left(\frac{\partial \overline{u}_\theta}{\partial z} + \frac{1}{r}\frac{\partial \overline{u}_z}{\partial \theta}\right),
\end{aligned}$$

$$\overline{d_{zr}} = \overline{\mu}\left(\frac{\partial \overline{u}_r}{\partial z} + \frac{\partial \overline{u}_z}{\partial r}\right). \tag{4.51}$$

The subgrid scale (SGS) variables τ_{ij} $(i, j = \theta, z, r)$ are defined by

$$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j. \tag{4.52}$$

We use the dynamic eddy viscosity model [21, 48] for τ_{ij} . The anisotropic part of τ_{ij} is modeled as

$$\tau_{ij}^M = \tau_{ij} - \frac{\delta_{ij}}{3}\tau_{kk} = -2C_S\Delta^2|\overline{S}|\overline{S}_{ij}, \tag{4.53}$$

where

$$\begin{aligned}
&\overline{S}_{\theta\theta} = \frac{1}{r}\frac{\partial \overline{u}_\theta}{\partial \theta} + \frac{\overline{u}_r}{r}, \quad \overline{S}_{zz} = \frac{\partial \overline{u}_z}{\partial z}, \quad \overline{S}_{rr} = \frac{\partial \overline{u}_r}{\partial r}, \\
&\overline{S}_{r\theta} = \frac{1}{2}\left(\frac{\partial \overline{u}_\theta}{\partial r} + \frac{1}{r}\frac{\partial \overline{u}_r}{\partial \theta} - \frac{\overline{u}_\theta}{r}\right), \quad \overline{S}_{\theta z} = \frac{1}{2}\left(\frac{\partial \overline{u}_\theta}{\partial z} + \frac{1}{r}\frac{\partial \overline{u}_z}{\partial \theta}\right), \\
&\overline{S}_{zr} = \frac{1}{2}\left(\frac{\partial \overline{u}_r}{\partial z} + \frac{\partial \overline{u}_z}{\partial r}\right).
\end{aligned} \tag{4.54}$$

and

$$|\overline{S}|^2 = \sum 2S_{ij}^2 \ . \tag{4.55}$$

The C_S is a model coefficient to be computed dynamically. Let a spatially test-filtered quantity be denoted by a caret. The test filtered stress T_{ij} is

defined by

$$T_{ij} = \widehat{\overline{u_i u_j}} - \widehat{\overline{u_i}}\ \widehat{\overline{u_j}}, \tag{4.56}$$

and the anisotropic part of T_{ij} is modeled as

$$T_{ij}^M = -2C_S\widehat{\Delta}^2|\widehat{\overline{S}}|\widehat{\overline{S}}_{ij}. \tag{4.57}$$

From the Germano's identity,

$$L_{ij} = T_{ij} - \widehat{\tau_{ij}} = \widehat{\overline{u_i}\overline{u_j}} - \widehat{\overline{u_i}}\ \widehat{\overline{u_j}}, \tag{4.58}$$

where L_{ij} is the Leonard stress. The right hand side is completely computable from the resolved variables. In addition,

$$L_{ij}^a = T_{ij}^M - \widehat{\tau_{ij}^M} = 2C_S\Delta^2\widehat{|\overline{S}|\overline{S}_{ij}} - 2C_S\widehat{\Delta}^2|\widehat{\overline{S}}|\widehat{\overline{S}}_{ij} = C_S M_{ij}\ , \tag{4.59}$$

where

$$M_{ij} = 2\Delta^2\widehat{|\overline{S}|\overline{S}_{ij}} - 2\widehat{\Delta}^2|\widehat{\overline{S}}|\widehat{\overline{S}}_{ij} \tag{4.60}$$

and L_{ij}^a is the anisotropic part of L_{ij}. We introduce an spatial averaging operation $\langle\cdot\rangle$ to avoid numerical problem. The specification of the average is problem dependent, as the universal definition of an ensemble average is inconvenient to use.

Applying this average to (4.59) and using least squares in the resulting

equations leads to the formula

$$C_S = \frac{\langle \sum L_{ij}^a M_{ij} \rangle}{\langle \sum M_{ij} M_{ij} \rangle} \ . \tag{4.61}$$

The dimensionless values of the parameters in our simulation are $L_z = \pi$, $R_\mathrm{i} = 1$, $R_\mathrm{o} = 2$, density $\rho = 1$ and dynamic viscosity $\mu = 1/8000$. The Reynolds number in this case is $Re = U_1(R_\mathrm{o} - R_\mathrm{i})\rho/\mu = 8000$. Periodic boundary conditions are imposed in the axial and azimuthal direction with no-slip boundary conditions at the inner and outer cylinder. The initialization is a uniform velocity field $U = 0.5$. The simulation was not terminated until a statistically-stationary state was achieved and at this stage, data was collected for a period of 250 dimensionless time units. We used three sets of meshes $64 \times 32 \times 32$, $64 \times 64 \times 32$ and $128 \times 64 \times 32$, in the coordinate directions: θ, z, r.

Fig. 4.7 compares the mean angular momentum $\langle u_\theta \, r \rangle / U_1 R_\mathrm{i}$ with experimental values ($Re = 8698$) which are available only for the region near the inner cylinder. The most refined mesh agrees with the experimental data for the region near the inner cylinder [66]. Fig. 4.7 also shows the agreement of our results with DNS [16] as the mesh is refined.

Similarly, in Fig. 4.8, the profile of the mean azimuthal velocity $\langle u_\theta \rangle / U_1$ agrees with that of the DNS simulation upon mesh refinement. The results using the subgrid scale model show slightly improved convergence. Results in both figures compare well with DNS [16] and with the experimental result of [66]. This verification and validation test evaluates positively the implemen-

tation of our fluid flow solver.

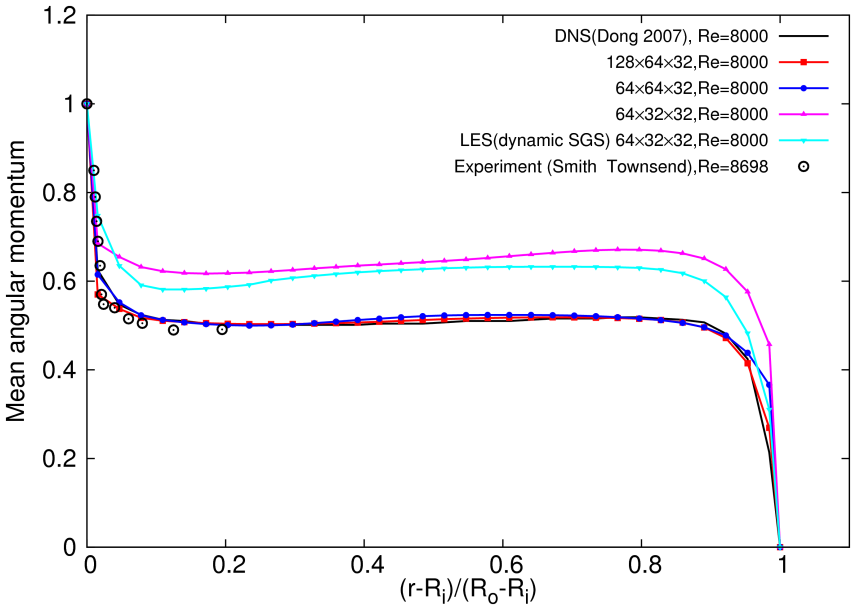

Figure 4.7: Mean angular momentum $\langle v_\theta\, r\rangle/U_1 R_\mathrm{i}$ for a turbulent (LES model) single-phase Couette flow.

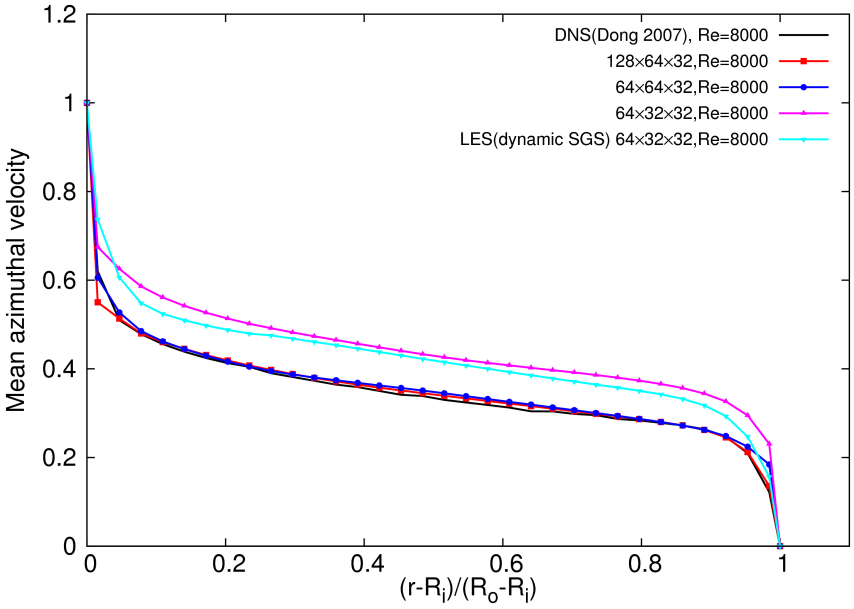

Figure 4.8: Mean azimuthal velocity $\langle v_\theta \rangle / U_1$ for a turbulent (LES model) single-phase Couette flow.

Chapter 5

Two-Phase Flow in Annular Sector

Two-phase Taylor-Couette flow has multiple applications, including liquid-gas oxygenators, and, as considered here, liquid-liquid extraction. However in various practical liquid-liquid systems of industrial interest the mass density and dynamic viscosity differences combined with fast cylinder rotation produce an extremely complex flow pattern of mixing that has received far less attention than the one-phase Taylor vortex flow counterpart.

Using the mathematical models and numerical methods developed in Chapter 2, we are able to present interaction of micron-sized drops and bubbles from the annular turbulent mixing in chemically reacting flows relevant to optimized design of contactor devices used in solvent extraction processes of nuclear spent fuel reprocessing. The macroscopically observed chemical reaction rate is critical in liquid-liquid solvent extraction and it depends on two hydrodynamic factors, which are the interfacial surface area and the extent of turbulent induced fluid diffusion, to transport unreacted chemicals to the fluid interface.

In this chapter we first present a verification study on low-speed two-phase couette flow with and without surface tension in the annular sector for demonstrating the order of accuracy of our methods. Then we present our simulation result and statistical picture for distribution of the droplets in the middle and at the end of the simulation. We also discuss about the role of turbulent diffusion in Taylor-Couette flow utilizing the analysis of other flows.

5.1 Verification of Two-Phase Annular Couette Flow

When the flow is circular 1-D in the presence of gravity pointing in the axial direction, the incompressible Navier-Stokes equations in cylindrical coordinates simplify to a one, non-zero velocity component in the azimuthal direction. In addition, if the pressure gradient is only a function of the radial direction, an exact solution can be constructed for the velocity and pressure unknowns as function of the radius and axial coordinates.

Denoting μ_1 and μ_2 the kinematic viscosity of fluid 1 and fluid 2, respectively, and R_I the radial position of the interface, the following constants of integration are obtained

$$B_1 = \frac{-\Omega_\mathrm{i} R_\mathrm{I}^2}{1 - R_\mathrm{Ii}^2 + \mu_{12} R_\mathrm{Io}^2 - \mu_{12}}, \tag{5.1}$$

$$A_1 = \Omega_\mathrm{i} - \frac{B_1}{R_\mathrm{i}^2}, \tag{5.2}$$

$$B_2 = \mu_{12} B_1, \tag{5.3}$$

and

$$A_2 = -\mu_{12}\frac{B_1}{R_{\rm o}^2}\ . \tag{5.4}$$

where $\mu_{12} = \mu_1/\mu_2,\ R_{\rm Ii} = R_{\rm I}/R_{\rm i},\ R_{\rm Io} = R_{\rm I}/R_{\rm o}$. Then in the fluid phase 1, $r \le R_{\rm I}$, the azimuthal velocity and pressure fields are

$$v_{\theta,1}(r) = A_1 r + \frac{B_1}{r}, \tag{5.5}$$

$$\begin{aligned} P_1(r,z) = \rho_1 \Bigg[\frac{A_1^2}{2}(r^2 - R_{\rm i}^2) + 2A_1B_1 \ln\frac{r}{R_{\rm i}} - \frac{B_1^2}{2}\Big(\frac{1}{r^2} - \frac{1}{R_{\rm i}^2}\Big)\Bigg] \\ - \rho_1 g(z - z_0) + P_0, \end{aligned} \tag{5.6}$$

and in the fluid phase 2, $r \ge R_{\rm I}$,

$$v_{\theta,2}(r) = A_2 r + \frac{B_2}{r}, \tag{5.7}$$

$$\begin{aligned} P_2(r,z) = \rho_1 \Bigg[\frac{A_1^2}{2}(R_{\rm I}^2 - R_{\rm i}^2) + 2A_1B_1 \ln\frac{R_{\rm I}}{R_{\rm i}} - \frac{B_1^2}{2}\Big(\frac{1}{R_{\rm I}^2} - \frac{1}{R_{\rm i}^2}\Big)\Bigg] \\ + \rho_2 \Bigg[\frac{A_2^2}{2}(r^2 - R_{\rm I}^2) + 2A_2B_2 \ln\frac{r}{R_{\rm I}} - \frac{B_2^2}{2}\Big(\frac{1}{r^2} - \frac{1}{R_{\rm I}^2}\Big)\Bigg] \\ - \rho_1 g(z - z_0) - \frac{\sigma}{R_{\rm I}} + P_0, \end{aligned} \tag{5.8}$$

where

$$C_1 = \frac{\Omega_{\rm i} R_{\rm i}^2}{R_{\rm i}^2 - R_{\rm o}^2}\ , \qquad C_2 = -C_1 R_{\rm o}^2\ . \tag{5.9}$$

The pressure jump across the interface is computed by subtracting (5.8) from (5.6)

$$P_1(R_{\rm I}, z) - P_2(R_{\rm I}, z) = \frac{\sigma}{R_{\rm I}}.$$

The above expressions are used as a verification and convergence rate test for our front tracking solution method as described next. Periodic boundary conditions in the z and θ directions are enforced in conjunction with no slip boundary conditions on the r-direction boundaries. The computational annular sector domain is $\theta \in [0, 0.628]$ rad, $z \in [0, 0.628]$ cm and $r \in [2.538, 3.166]$ cm, and the following values were selected

$$\rho_1 = 0.811 \text{ g/cm}^3, \qquad \rho_2 = 1.03 \text{ g/cm}^3,$$
$$\mu_1 = 57.6 \text{ g/cm s (5760 cP)}, \qquad \mu_2 = 36.72 \text{ g/cm s (3672 cP)}.$$

with the angular speed of the inner cylinder is $\Omega_{\text{i}} = 0.22367$ rad/s (2.1249 RPM) which results in the Reynolds number $Re = 0.01$ to approximate a zero inertial flow. The gravity g is neglected and the interface position at $r = 2.9$ cm is used as the initial condition.

In addition, three different values of interfacial tension coefficients were used in various tests, namely, $\sigma = 0$, $\sigma = 0.001$, and $\sigma = 10$ dyn/cm. Relative errors of the interface position, velocity field u_θ, and pressure field p were calculated using reference values in both L_1 and L_∞ norms for $N \times N \times N$ grids with N equal to $10, 20$ and 40. In the calculation of relative errors, the exact position of the interface was used, *i.e.*, $R_{\text{I}} = 2.9$ cm, the speed of the inner cylinder was chosen as the velocity scale, $\Omega_{\text{i}} R_{\text{i}} = 0.56768$ cm/s, and for the reference pressure, the maximum pressure difference Δp in the computation domain was used; this value is different for each of the three test cases. The CFL number was adjusted (reduced) for this two-phase flow test

case.

5.1.1 Couette flow with no interfacial tension

Without interfacial tension the pressure is continuous. The errors are computed at time $t = 28.1$ s (1 revolution) when the reference pressure is $\Delta p = 0.023257$ dyn/cm^2. Results for 3D simulations are collected in tables 5.1, 5.2, and 5.3.

Table 5.1: Solution verification and calculation of the convergence rate of the interface position for 1D, two-phase Couette flow without interfacial tension, $\sigma = 0$.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	2.6857e-8	NA	2.6866e-8	NA
20×20	3.1387e-9	3.0971	3.2010e-9	3.0692
40×40	2.9334e-11	6.7413	7.9670e-11	5.3283

Table 5.2: Solution verification and calculation of the convergence rate of u_θ for 1D, two-phase Couette flow without interfacial tension, $\sigma = 0$.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.0025184	NA	0.013839	NA
20×20	0.0011825	1.0907	0.0089610	0.6270
40×40	0.0005298	1.1583	0.0025929	1.7891

5.1.2 Couette flow with lowered interfacial tension

Similarly, for this test with perturbed interfacial tension, errors are computed from 3D simulations at time $t = 28.1$ s (1 revolution) and with the refer-

Table 5.3: Solution verification and calculation of the convergence rate of p for 1D, two-phase Couette flow without interfacial tension, $\sigma = 0$.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.0044984	NA	0.019847	NA
20×20	0.0013152	1.7741	0.0064381	1.6242
40×40	0.0004539	1.5348	0.0020022	1.6850

ence pressure $\Delta p = 0.022912$ dyn/cm^2. The results displayed in tables 5.4, 5.5, and 5.6 are similar to the case with no interfacial tension except for the pressure calculation which does not converge with mesh refinement in the L_∞ norm. With a small, non-zero interfacial tension, the analytical solution for pressure becomes discontinuous and this cannot be accommodated by the numerical solution method which approximates the pressure jump with a continuous field.

Table 5.4: Solution verification and calculation of the convergence rate of the interface position for 1D, two-phase Couette flow with perturbed interfacial tension, $\sigma = 0.001$ dyn/cm.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	2.6941e-8	NA	2.6951e-8	NA
20×20	3.1471e-9	3.0977	3.2094e-9	3.0699
40×40	2.4069e-11	7.0307	3.2086e-11	6.6442

5.1.3 Couette flow with interfacial tension

In this test for a larger interfacial tension, 3D calculations of the error at time $t = 28.1$ s (1 revolution) and the reference pressure $\Delta p = 3.448276$ dyn/cm

Table 5.5: Solution verification and calculation of the convergence rate of u_θ for 1D, two-phase Couette flow with perturbed interfacial tension, $\sigma = 0.001$ dyn/cm.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.0025184	NA	0.013839	NA
20×20	0.0011825	1.0907	0.0089610	0.6270
40×40	0.0005298	1.1583	0.0025929	1.7891

Table 5.6: Solution verification and calculation of the convergence rate of p for 1D, two-phase Couette flow with perturbed interfacial tension, $\sigma = 0.001$ dyn/cm.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.0041033	NA	0.020524	NA
20×20	0.0013926	1.5590	0.006899	1.5729
40×40	0.0005164	1.4312	0.002690	1.3588

were obtained. We demonstrate by numerical experiment that the algorithm

Table 5.7: Solution verification and calculation of the convergence rate of the interface position for 1D, two-phase Couette flow with interfacial tension, $\sigma = 10$ dyn/cm.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	8.2300e-7	NA	8.2364e-7	NA
20×20	9.2962e-8	3.1462	9.3070e-8	3.1456
40×40	9.3184e-9	3.3185	9.3199e-9	3.3199

is first-order accurate for the interface position and velocity field as well as for pressure in an L_1 error norm. For the $\sigma = 10$ dyn/cm interfacial tension case, the L_∞ error for pressure does not converge using the continuous surface tension model, which is consistent with the description of IB method in

Table 5.8: Solution verification and calculation of the convergence rate of u_θ for 1D, two-phase Couette flow with interfacial tension, $\sigma = 10$ dyn/cm.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.0025164	NA	0.0138301	NA
20×20	0.0011822	1.0899	0.0089558	0.6269
40×40	0.0005297	1.1582	0.0025923	1.7886

Table 5.9: Solution verification and calculation of the convergence rate of p for 1D, two-phase Couette flow with interfacial tension, $\sigma = 10$ dyn/cm.

	L_1 error	Order in L_1 norm	L_∞ error	Order in L_∞ norm
10×10	0.0382376	NA	0.2708899	NA
20×20	0.0323036	0.2433	0.4541237	-0.7454
40×40	0.0062075	2.3796	0.1725304	1.3962

Sec. 2.2.2.

5.2 Two-Phase Turbulent Mixing in Annular Sector

At a high Reynolds number, a two-phase Couette flow of two immiscible liquid phases with distinct mass density and viscosity will not remain stratified. When the inner cylinder rotates at a high speed, the heavier fluid is propelled outwards in the direction of the outer cylinder wherein high shear stresses promote vigorous mixing and dispersion formation and the phases are fully mixed by combined shear and centrifugal forces.

We focus on the annular mixing region in a centrifugal contactor and to keep the computational effort at a reasonable cost, a sector of the annulus is selected for analysis by means of periodic boundary conditions in the axial and

azimuthal directions (Fig. 5.1). This truncation of the annulus requires the neglect of gravity because of axial periodicity. When comparing to a realistic device, this assumption is only plausible in the bulk of the mixing zone where the centrifugal force imparted in the fluid surpasses gravity substantially.

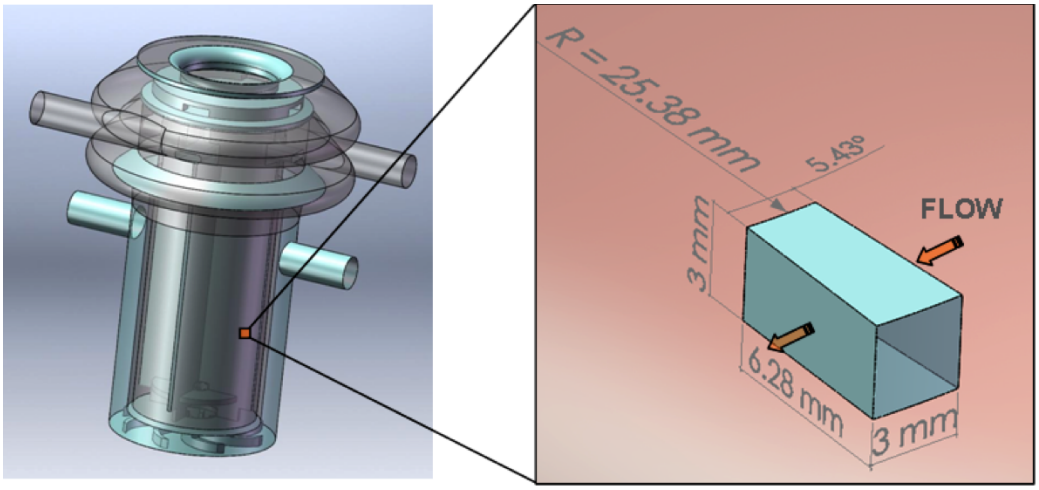

Figure 5.1: Annular sector of a centrifugal contactor used for the computation domain. On the left is the complete flow domain in a centrifugal contactor. On the right is the computational sector used.

5.2.1 Role of turbulent diffusion

Chemical species in liquids diffuse relatively slowly, so that diffusion limited chemical reactions are common. Vigorous stirring leads to high Reynolds number flow and turbulence. At high Reynolds numbers, turbulent transport is usually more important than molecular transport. In Table 5.10, we list molecular and turbulent properties for the TC flow. The Schmidt number, the ratio of viscosity to species diffusion, is a dimensionless measure of species

diffusion; it influences the rate of chemical reactions, especially diffusion limited ones.

Since in this study the flow has only a single species in each phase there is no diffusion for current simulations, we compare this flow with a Rayleigh-Taylor flow [65, 42] for the mixing of fresh and salt water at a comparable Reynolds number. See Table 5.10. Turbulent viscosity and turbulent species diffusion rates are mesh dependent, but in a recent study we observed quite stable values for their ratio, the turbulent Schmidt number, at high Reynolds numbers [51]. From Table 5.10, we conclude that turbulent viscosity plays a small role in our large eddy simulations (LES) of TC flow, but comparison to our RT flow suggests a strong role for turbulent diffusion, and the turbulent Schmidt number, once chemistry is considered, in addition to the hydrodynamics.

	RT	TC (aqueous)	TC (organic)
molecular kinematic viscosity (cm^2/s)	1.0×10^{-2}	9.9×10^{-3}	1.9×10^{-2}
turbulent kinematic viscosity (cm^2/s)	8.0×10^{-4}	3.6×10^{-4}	2.3×10^{-4}
molecular diffusion (cm^2/s)	1.8×10^{-5}	–	–
turbulent diffusion (cm^2/s)	2.4×10^{-3}	–	–
molecular Schmidt number	5.6×10^{2}	–	–
turbulent Schmidt number	3.3×10^{-1}	–	–
Reynolds number	$\approx 25 \times 10^{3}$	25×10^{3}	12×10^{3}

Table 5.10: Viscosity and mass diffusion for Rayleigh-Taylor and Taylor-Couette flow.

5.2.2 Fully developed two-phase T-C flow

We initialize the flow in a maximally unstable configuration, with the aqueous phase (heavy fluid) in the inner region and the organic phase (light fluid) in the outer region between the cylinders. The initial perturbed interface between the two phases is positioned to achieve nearly a 0.5 volume fraction for the organic and aqueous phases. We use periodic boundary conditions in the axial and azimuthal directions and no-slip boundary conditions at the inner and outer cylinder walls.

The inner cylinder speed is set at 1500 rpm while the outer cylinder is fixed. We use the analytical Couette flow fields (5.5)–(5.8) as the initial condition and the geometrical parameters and physical properties used in the simulation are collected in Table 5.11.

R_{i}	2.538 cm
R_{o}	3.166 cm
$\ell = R_{\mathrm{o}} - R_{\mathrm{i}}$	0.628 cm
$\eta = R_{\mathrm{i}}/R_{\mathrm{o}}$	0.8
Ω_{i}	157 radians/s (1500 RPM)
μ_{org}	0.016 g/cm $\cdot$ s
μ_{aqu}	0.0102 g/cm $\cdot$ s
ρ_{org}	0.811 g/cm^3
ρ_{aqu}	1.03 g/cm^3
$Re_{\mathrm{org}} := \frac{\Omega_{\mathrm{i}} R_{\mathrm{i}} \ell}{\nu_{\mathrm{org}}}$	1.27×10^4
$Re_{\mathrm{aqu}} := \frac{\Omega_{\mathrm{i}} R_{\mathrm{i}} \ell}{\nu_{\mathrm{aqu}}}$	2.53×10^4
σ	25 dyn/cm

Table 5.11: Geometric parameters and physical properties for 3D simulation of two-phase flow in an annular sector.

Fig. 5.2 shows the growth of the interfacial area from a time after the

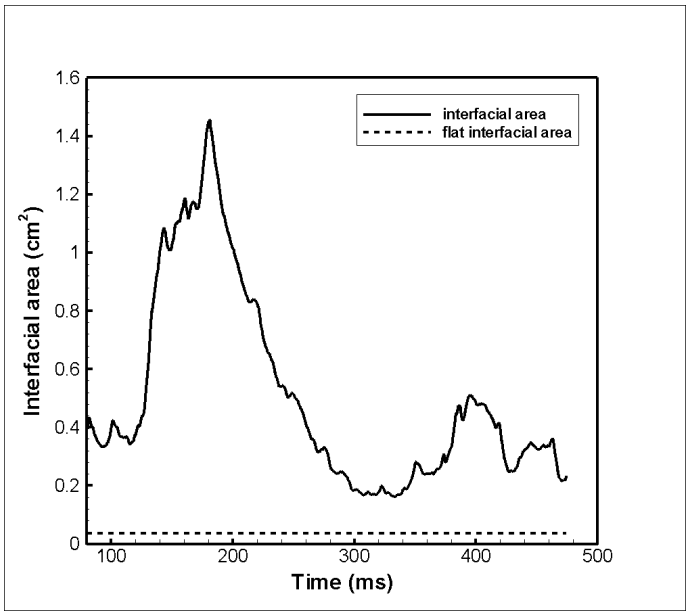

Figure 5.2: Growth of interfacial area with time.

initial fluctuation to a late time with a presumed statistically steady state. We observe that a transient chaotic flow regime with a high surface area is followed by a statistically stable flow regime at a late time. This state has a surface area much higher than that of a simple cylindrical surface to divide the two fluids, but it is also significantly lower than the peak of the interfacial area during the transient regime. The transient regime has an interfacial area some seven times larger than that observed for the late time flow. The late time interface surface area is itself about 10 times that of the simple cylindrical interface.

The transient chaotic flow regime has significant breakup of the interface between the two phases into small droplets and thus an extensive interfacial area. See Fig. 5.3 and Fig. 5.4. We found that the initial unstable configura-

tion of the two continuous phases approaches a stable two phase configuration at a late time as a statistically steady state through the transient chaotic flow regime. The droplets are formed at the unstable interface and they migrate in their respective stable directions and then are segregated into four continuous phase in layers from the inside to the outside, as oil, water, oil, water. See Fig. 5.5. The two central phases and the interface between them are unstable and the unstable interface generates new unstable droplets, migrating to the two stable phases. Finally the statistically stable flow regime has two continuous phases with the organic phase on the inner region and the aqueous phase on the outer region, each with a few droplets of the dispersed phase embedded in it. See Fig. 5.6. We call this statistically stable configuration the centrifuge mode.

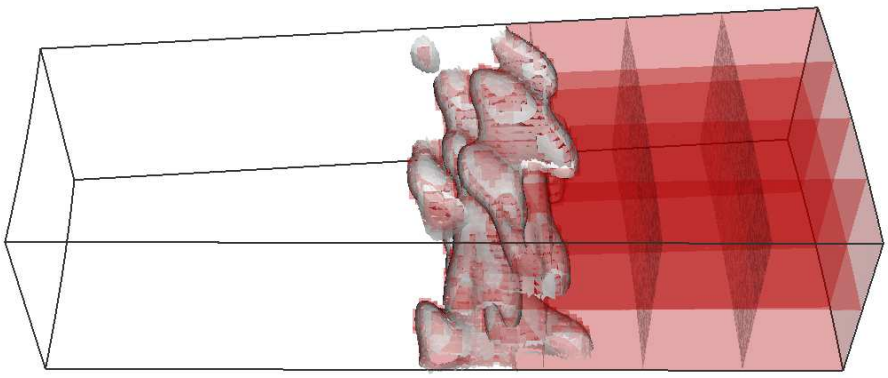

Figure 5.3: Organic phase at an early time with unstable configuration.

In Fig. 5.7 and Fig. 5.8, we show the statistical distribution at two dif-

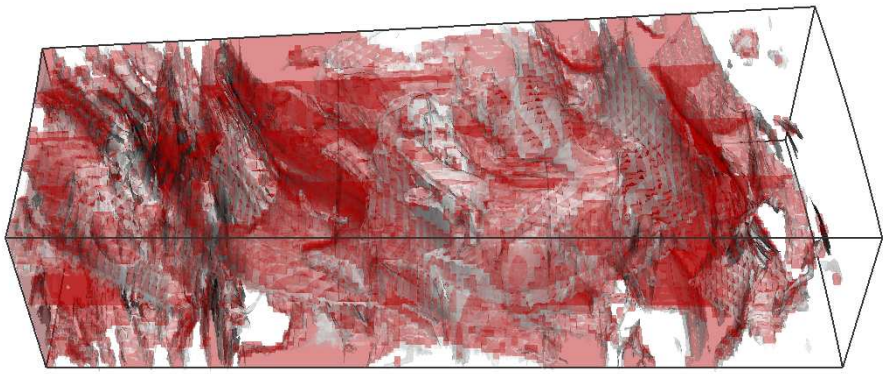

Figure 5.4: Organic phase at a transient chaotic flow regime.

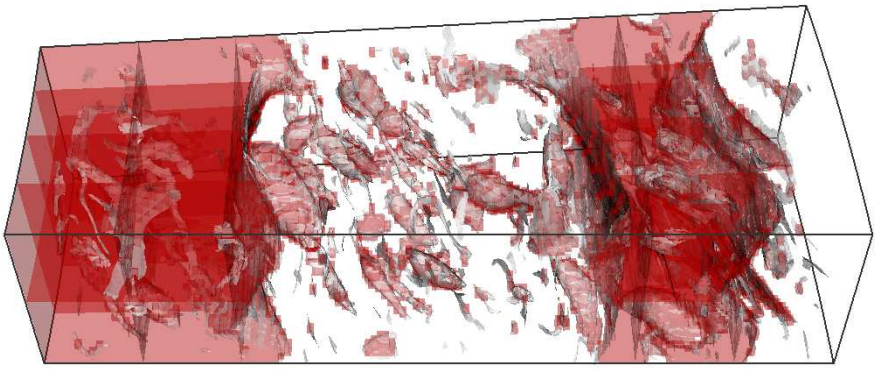

Figure 5.5: At t = 240 ms, the two phase are now broken up into four domains, arranged from inner to outer radius, in which the dominant continuous phase is (in this order) light (oil, red), heavy, light (oil, red), heavy.

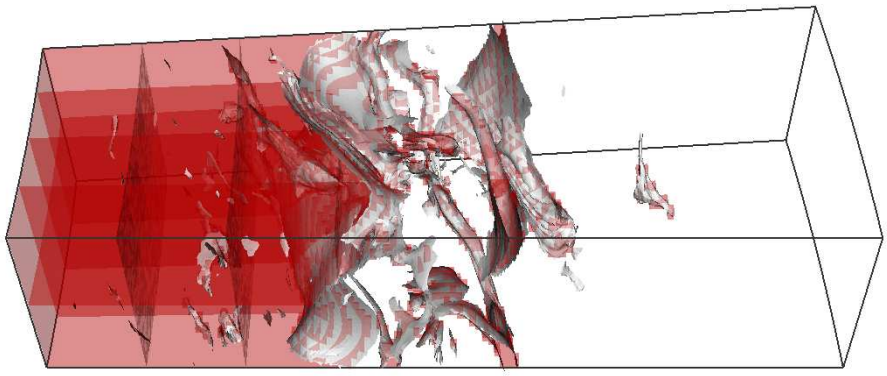

Figure 5.6: Organic phase at the statistically stable flow regime.

ferent time regimes. A large number of small droplets are observed, but on a volume weighted basis, they contribute little. The histogram in Fig. 5.7 shows the total volume of droplets vs. the droplet diameter size. Fig. 5.8 shows the frequency of droplets vs. the droplet diameter size. We found small droplets primarily on a transient basis only, and in a late time statistical steady state we observe a flow segregated into distinct domains, each with a dominant fluid as a continuous phase.

There is a significant difference between the centrifugal mode our simulation reaches and the experimental picture for the late time flow. The experimental picture has a single continuous phase and many small droplets with sizes around 60 microns forming a honeycomb structure with thin lubricating walls of some 10 microns in thickness. We believe the main reason for this difference is the missing disjoining pressure between slightly separated

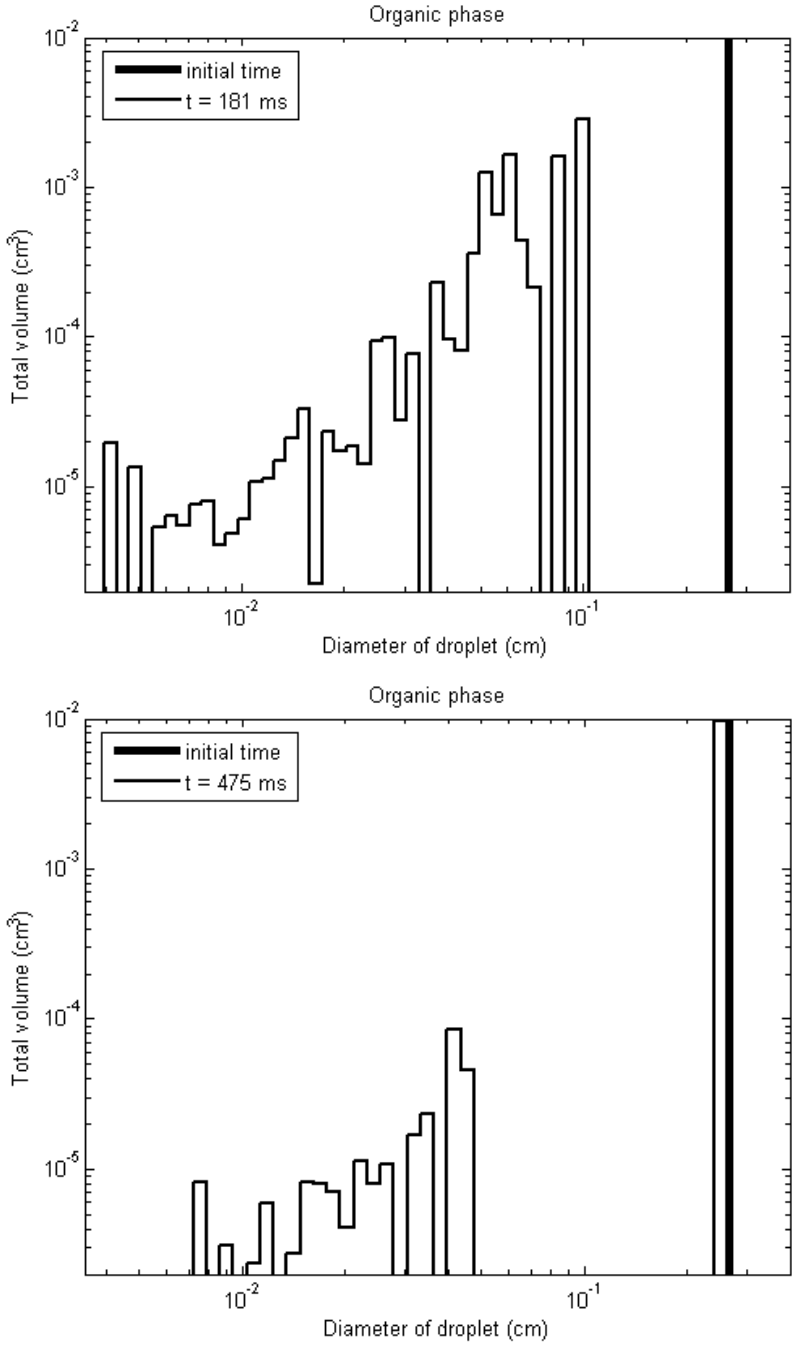

Figure 5.7: Total volume of droplets vs. diameter at a transient chaotic regime (above) and a statistically stable flow regime (below).

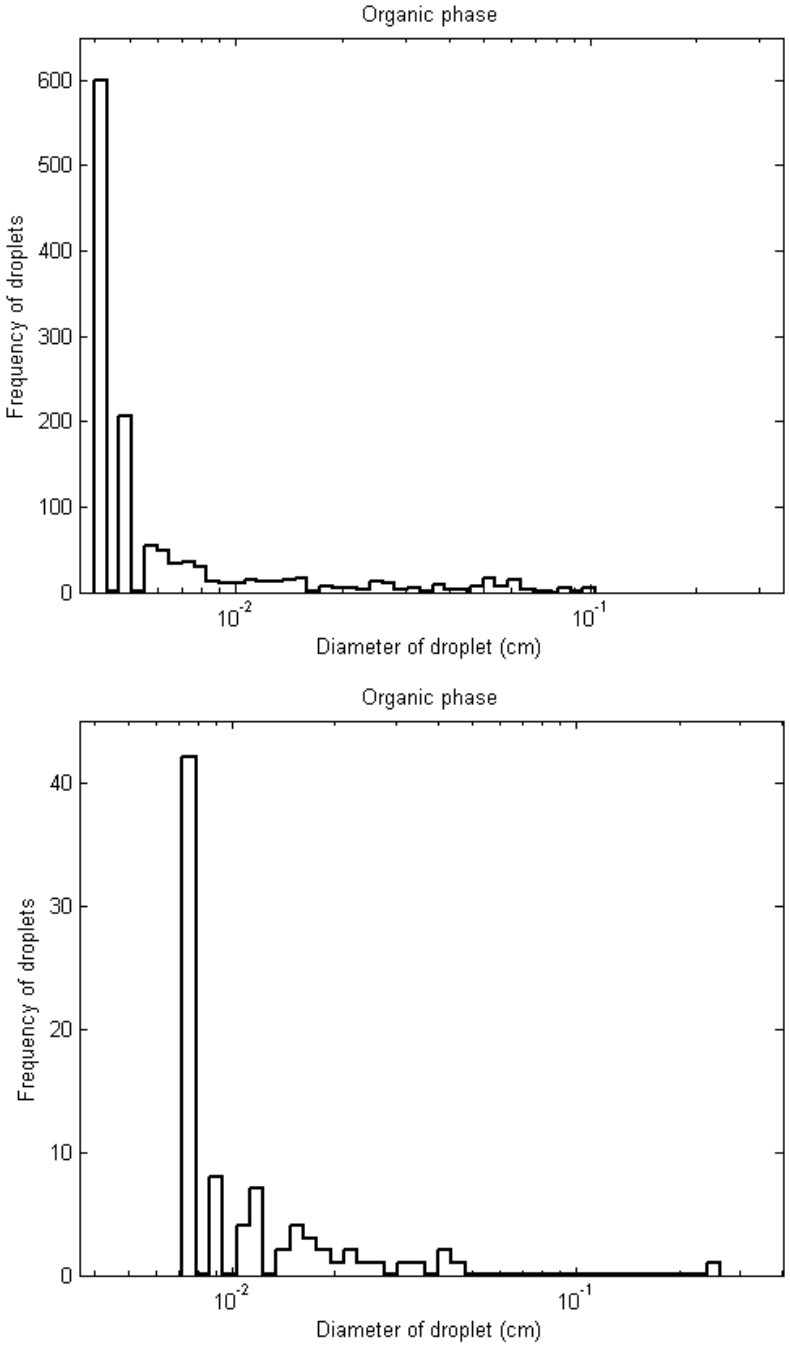

Figure 5.8: Frequency of droplets vs. diameter at a transient chaotic regime (above) and a statistically stable flow regime (below).

droplets, similar to what if found for the physics of lubricating thin films. Here we propose a different subgrid scale picture, motivated by experimental photographs. If small droplets come in close contact, rather than merging, they may form a type of foam, and occupy the majority of the volume while still being a dispersed phase. Direct resolution of this phase will be difficult and perhaps unfeasible, but a subgrid model wherein the hydrodynamic pressure is corrected for fluid confinement via a disjoining pressure of molecular origin could be effective. Also as in the experiment air is added as a third phase, the influence of air could also be a factor. Other possible factors include various domain sizes and different rotating speed of inner cylinder.

With subgrid model added and domain size increased, the geometry of interface would become more complicated than current simulation. Also adding the influence of air would require the interface library to be able to handle non-manifold surfaces. These requirements promote the develop of a more robust interface library. In Chapter 6 we present a new parallel triangular mesh structure which serves as the basis for this requirement.

Chapter 6

HiProp: A Robust Parallel Mesh Library and Its Applications

In this chapter we develop a robust and efficient array based parallel triangular mesh library called *HiProp* which serves as the basis for parallel mesh algorithms. In this new library, any element (point or triangle) shared with other processors maintains a complete parallel information list which has the processor ranks and local IDs for all the processors sharing that element. Each point or triangle thus has a unique global ID which is defined by the processor rank/local ID pair. The attributes on an element could only be updated by its master processor. Compared with the interface communication routine scatter_front() for *FronTier* described in Sec. 2.2.1, this new communication model requires no floating point comparison except in initialization, generates unique global results for parallel mesh algorithms and is more suitable for parallel high-order mesh algorithms. For the convenience of updating attributes defined on the elements, parallel updating lists are built locally based on the parallel information lists. The new library also supports periodic boundary

condition by adding shift flag into the parallel information structure. The details of the new library structure are described in Sec. 6.1.

Most of the high-order mesh algorithms require n-rings of ghost triangles, where n is decided by the number of neighboring points and triangles the algorithms need for each element. Also most parallel simulation codes solving numerical PDEs decompose the spatial domain into N parts, one part solved on each processor. Thus for using the library with numerical PDEs solvers, we need the capability for building ghost triangles based on either the connectivity or spatial decomposition. In Sec. 6.2 we describe the algorithm for building ghost triangles based on the parallel information constructed while maintaining the parallel information during the procedure.

Two examples of parallel mesh algorithms implemented based on this new library, high-order normal and curvature calculation and mesh optimization, are presented in Sec. 6.3. The parallel information discussed in Sec. 6.1 has to be constructed before using the algorithms. The flow chart for using the new library is shown in Fig. 6.1. Under this framework, the parallelization of the high-order functional propagation algorithm is straightforward and the numerical results for two benchmark tests are presented in Sec. 6.3.3 .

As the new HiProp library does not support all the functions in *FronTier* currently, in Sec. 6.4 we present a complete surface propagation package with both high-order smoothing and locally grid based untangling by coupling *HiProp* with *FronTier* through an interface between the two libraries together. We export the triangular mesh from *HiProp* to *FronTier* for locally grid based untangling if a self-intersection is detected through a collision detection algo-

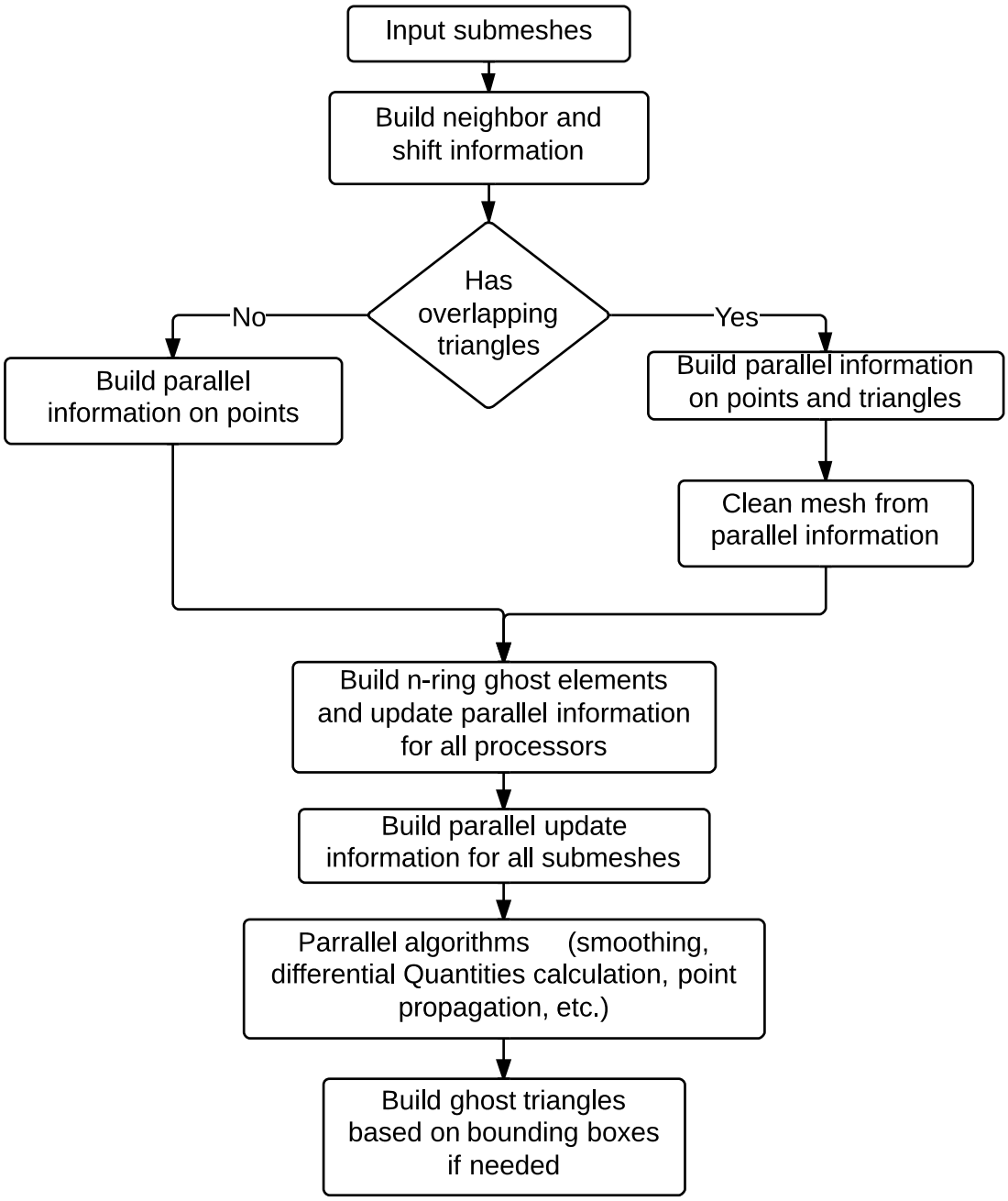

Figure 6.1: HiProp library flow chart

rithm.

6.1 *HiProp* Parallel Mesh Structure

6.1.1 Array-based mesh data structure

Mesh data structures are the foundation of mesh algorithms, mesh-based numerical methods and the parallel mesh structures. Although the mesh data structures are highly application specific, there are certain requirements that are common to all applications. They should be simple, efficient in both space and time, neutral of programming language, convenient for I/O, generalizable to higher dimensional and non-manifold surfaces and easy to support partitioned meshes as well as mesh communications. The mesh structures must also support various queries such as finding the neighboring entities efficiently on the mesh.

Most of the mesh structures are divided into two types, pointer-based and array-based. While the pointer-based structures are more popular as they are relatively easy to manipulate and mesh entities could be recognized as objects for Object Oriented Programming (OOP), they usually suffer from larger storage requirements for the pointers and are inconvenient for parallel mesh communication between processors. The front tracking package *FronTier* has a pointer-based mesh representation. In an array-based mesh structure, the entities are not represented as objects. Instead, an attribute of all the entities of the same type is stored in one or few arrays, and the attributes for a single entity might be distributed in different arrays. An entity may be

referenced through an ID, which can be mapped easily to array indices.

We choose to use the compact array-based half-edge data structure introduced in [1]. This array-based structure provides comprehensive and efficient support for querying incidence, adjacency and boundary classification while requiring substantially less memory than pointer-based mesh representations [1]. It also supports partitioned meshes and interprocessor mesh communication in a natural way. Here we focus on orientable, manifold triangular meshes, the generalization of this data structure could be used also for volume meshes [1] and non-manifold meshes [18].

In this array-based data structure, we assume that the points are assigned consecutive integer IDs ranging from one to the number of points, and similarly for triangles. Each triangle has the same local numbering convention same as CGNS conventions shown in Fig. 6.2. The point positions are stored in a $n \times 3$ double matrix and the triangle connectivities are stored in a $m \times 3$ integer matrix where n is the number of points and m is the number of triangles. For a parallel mesh, each partition has its own numbering system.

As shown in Fig. 6.2, each triangle is bounded by a loop of directed edges in counterclockwise order thus each edge has two directed half-edges with opposite directions belonging to two adjacent triangles except for boundary edges. In the array-based half-edge data structure we use, the half-edge list is represented by a $m \times 3$ integer matrix $\mathbf{OHE}_{m\times 3}$ where m is the number of triangles. Element $\mathbf{OHE}(i, j)$, corresponding to j-th half-edge in i-th triangle on the mesh, is composed of a pair of numbers $\langle f, j\rangle$ where f is the ID of the adjacent triangle and j is the local ID of the opposite half-edge in the

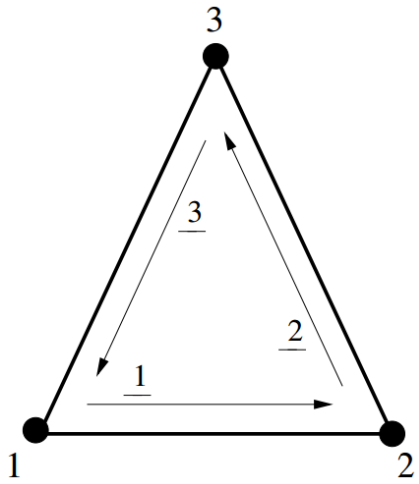

Figure 6.2: Local numbering convention for triangle. Underscored numbers correspond to local edge IDs and the vertex next to an edge is the first vertex of the edge.

adjacent triangle. $\mathbf{OHE}(i, j) = 0$ for boundary edges. We also build a $n \times 1$ integer vector $\mathbf{IHE}_{n\times 1}$ called incident half-edge array where n is the number of points. Element $\mathbf{IHE}(k)$, corresponding to the k-th point, has a half-edge data originated from that point and is composed of a pair of numbers $\langle f, j\rangle$ where f is the triangle containing that half-edge and j is the local index of the half-edge. Both **OHE** and **IHE** could be built on-the-fly in linear time and need to be updated if mesh is modified. Using these two lists, we are able to perform mesh queries such as finding d-rings of neighborhood for a point, which is necessary for our ghost exchange algorithm. Most of our parallel mesh algorithms are built based on this array-based half-edge structure.

6.1.2 Parallel information list

With the array-based triangular mesh on each processor, we describe here the parallel information lists for identifying entities globally and also give the algorithms for building these lists. We assume that the global mesh has been partitioned to submeshes and each processor reads its own portion of the mesh. We call the input submeshes *clean* if each submesh only intersects with other submeshes in a partition boundary curve, i.e., with no overlapping triangles. The input submeshes could be clean or non-clean and if the input type is unknown, it is always recognized as non-clean.

Before we build the parallel information list we first build the neighboring processor list for each processor via an all-to-all communication. The purpose for doing this is that for the following algorithms with message exchange, a point-to-point instead of all-to-all communication could be used to improve the efficiency. Two processors p and q are considered to be neighbors if there exists a common point: $S_p \cap S_q \neq \varnothing$, where S_p and S_q are submeshes on processor p and q respectively. Assume that the total number of processors is N. For a specific processor p_i, we first gather the possibly overlapping point set $\mathcal{P}_{ij} = \{\mathbf{v} \in \mathcal{B}_j\}, j = 0 \ldots, N-1, j \neq i$ with each of the other $N-1$ processors where $\mathcal{B}_j$ is the bounding box of the submesh on processor j . Then we communicate $\mathcal{P}_{ij}$ using a non-blocking send non-blocking receive model for maximally overlapping the communication and computation time, which is critical for good performance of communication models using message passing. On a specific processor p_i, after we receive the point set $\mathcal{P}_{ji}$ from processor p_j, we compare the two sets $\mathcal{P}_{ij}$ and $\mathcal{P}_{ji}$ using floating point comparison with

tolerance ϵ. This is a $\mathcal{O}(n_l^2)$ algorithm where n_l is the number of points in the local overlapping point sets calculated from the bounding boxes. For most applications, especially when the input submeshes are clean, $n_l = \mathcal{O}(\sqrt{n})$, where n is the number of points on each submesh. Thus the overall time complexity of this algorithm is $\mathcal{O}(n)$.

With the neighboring processor information, now we build the parallel information lists for both points and triangles on each processor. First we consider the parallel information lists for points. The parallel information lists for points is an array of n lists $\mathcal{L}$ where n is the number of points and $\mathcal{L}(i)$ is the parallel information list for the i-th point. Each node in the lists has a processor rank ID, a local index ID on the corresponding processor and a pointer to the next node. Each node in the list $\mathcal{L}(i)$ represents a duplication of the i-th point on one processor, including the processor that this point is currently on. The first node (head node) in each list is called the *master node*, which means any attribute defined on that point is uniquely updated by the processor stored in that node. The other nodes are called *slave nodes*. For example, in Fig. 6.3 we see that the local point v_3 on processor 0 also exists on processor 1 and processor 2, with processor 0 as the master processor. Thus on processor 0 for point v_3 we have parallel information list $0/v_3 \to 1/v_1 \to 2/v_4$ which means the point exists on three processors and the local index IDs for the point on three different processors are v_3, v_1, v_4 respectively. Also $0/v_3$ is the head node. Thus we know that processor 0 is the master processor for this point. Similar parallel information list is kept on processor 1 and 2 and we could see that for the same point, although the order of the slave nodes could

differ on various processors, we always keep the master node as the first node of the list. Note that our array of lists is also array-based, which means we use index rather than pointer for representing a list data structure. The benefit for using array-based list representation is that it has better memory locality and is more suitable for communication. The index for head nodes and tail nodes are kept as two arrays with size n for efficient operations on the heads or the tails of the lists, which we could see later, is necessary for building parallel information lists.

The algorithm for building the parallel information list for points is similar to the algorithm for getting neighboring processor list referred above in that floating points comparison with tolerance ϵ is used for two possibly overlapping point sets calculated from the bounding boxes of the submeshes. With the neighboring processor information, we only need point-to-point communications now. In the algorithm for building parallel information lists, assume we have processor p and point set receiving from processor q and the i-th point on p is considered to be the same as the j-th point on q. We then need to insert node q/j to the list $\mathcal{L}(i)$. We insert the node either to the head or to the tail depending on the relationship between p and q. If $rank(p) < rank(q)$, the node is inserted to the tail and if $rank(p) > rank(q)$ the node is inserted to the head which means that in the initialization of parallel information, for the same point on different processors, the processor with a lower rank would be the master. For the points only existing on the current processor, list with a single node is created.

For a clean mesh, we could build the parallel information lists for m

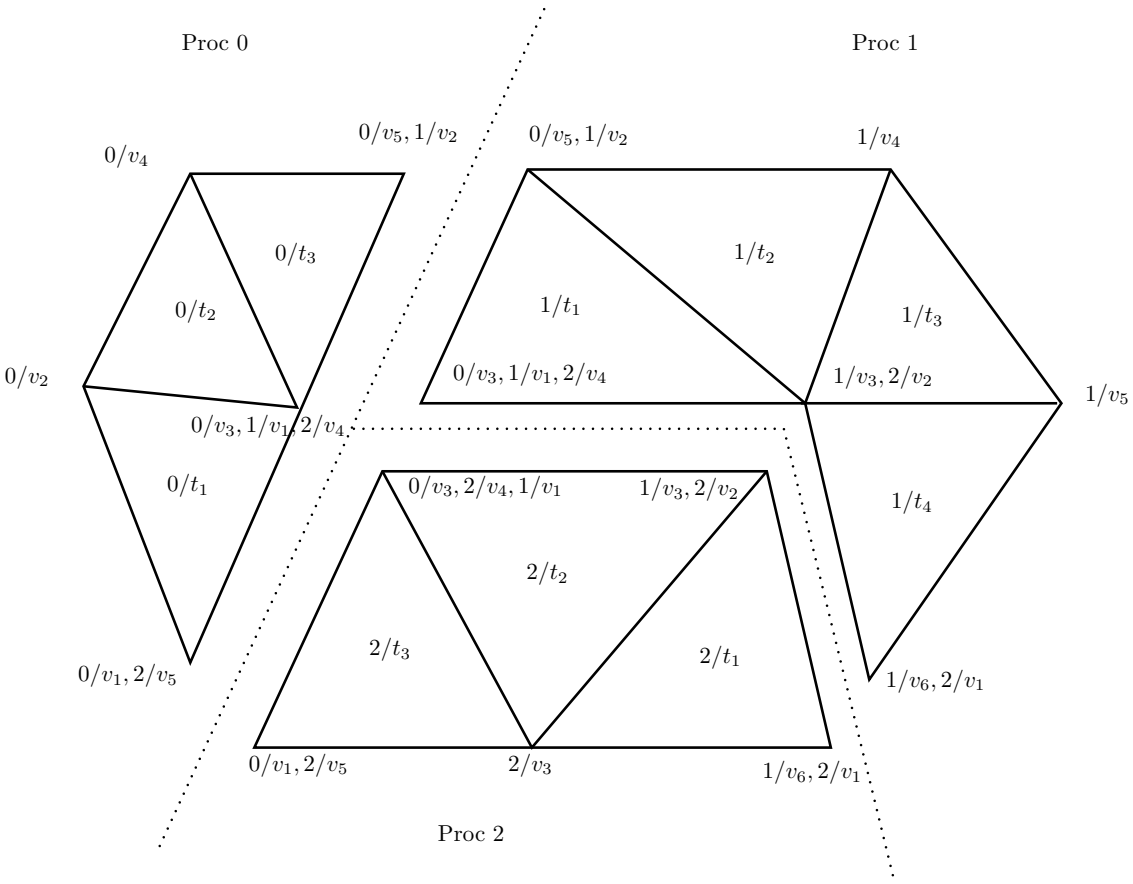

Figure 6.3: Submeshes with parallel information list.

triangles in a trivial way by inserting single nodes for all m lists. However, for non-clean meshes, parallel information lists for triangles also needs to be calculated. This is done by using the connectivity and the parallel information lists of the points. For the triangle t with vertices v_0, v_1, v_2 on processor p, the triangle $\hat{t}$ from processor q is considered to be the same triangle if v_0, v_1, v_2 also exist on q. The parallel information node for $\hat{t}$ is inserted under the same assumption that the processor with lower rank is the master. Similar to the analysis for building neighboring processor list, the time complexity for building parallel information lists for points is $\mathcal{O}(n)$ and $\mathcal{O}(m)$ for triangles. Fig. 6.3 is an example of clean submeshes with complete parallel information lists.

Using the parallel information lists we could classify all entities into three types: GHOST, OVERLAY and INTERIOR. For points and triangles on processor p, GHOST entities have a master processor $\neq p$, OVERLAY entities have master processor p but also exist on another processor and INTERIOR entities only exist on current processor. The attributes on GHOST entities are always updated from other processors while the OVERLAY entities update other processors and INTERIOR entities only update itself. In Fig. 6.4 we show an example of a partitioned mesh and the visualization of the points types for processor 2 after the parallel information lists are built. The yellow points are INTERIOR points, green points are OVERLAY points and blue points are GHOST points.

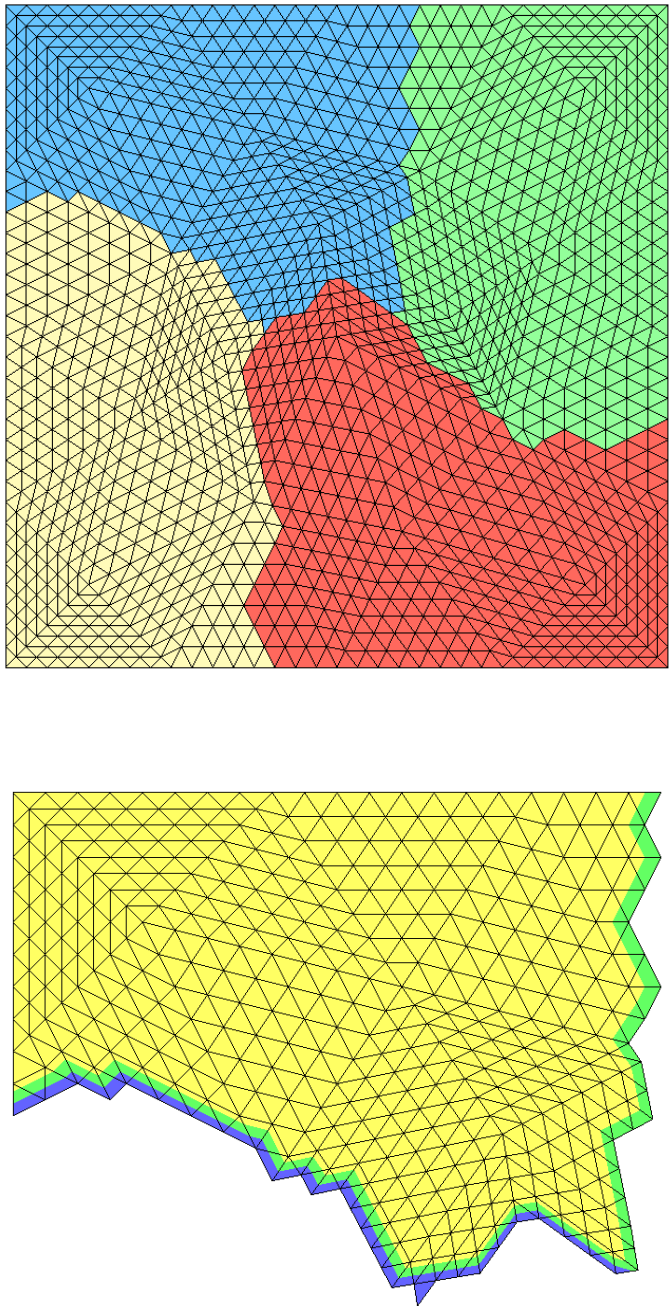

Figure 6.4: A mesh partition (upper) and points types of processor 2 (lower).

6.1.3 Parallel updating list

With a complete parallel information list, we are now able to update any attribute defined on the entities, for example, point positions or velocity field. For any point or triangle, if it is shared by other processors, we receive the attributes if the current processor is not the master processor or send attributes to the processors sharing the same entity. To avoid too many small messages for communication, we construct a parallel updating list between any two neighboring processors for efficiently updating the attributes on entities. First we discuss the parallel updating list for points and the same updating list could also be built for triangles if needed.

For two neighboring processors p and q, we keep a local array on processor p which maintains the indices of the OVERLAY points shared with processor q and on processor q we keep a corresponding local array which has the indices of the GHOST points shared with processor p. We collect these points by simply traversing through the parallel information list of the points. The receiving array on processor q needs to be sorted based on the master index IDs so that the two arrays are in the same order. See Fig. 6.5 for an example of the parallel updating lists. Using these arrays, we could easily build the sending and receiving buffers for any attribute and update the values in a linear time. In the applications where attributes on entities, for example, point positions need to be updated after mesh modification, this function needs to be called before we update the attributes.

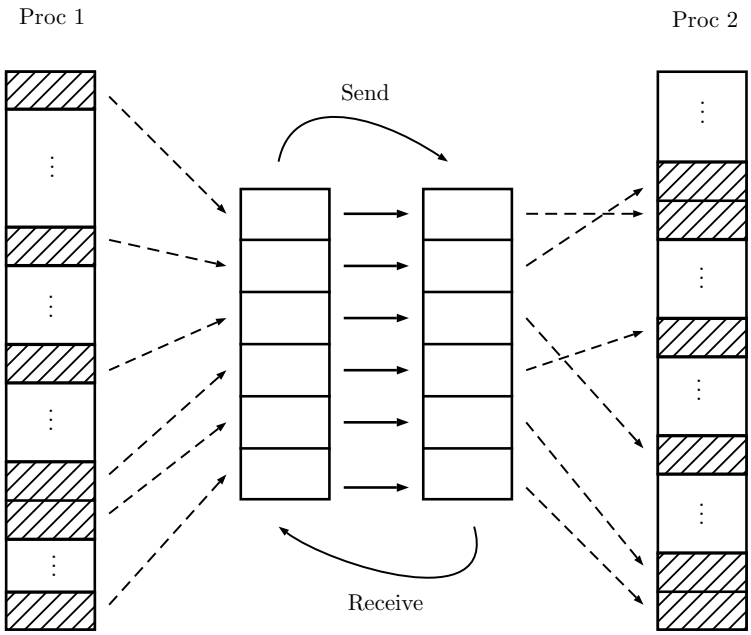

Figure 6.5: Parallel updating list between two neighboring processors.

6.1.4 Boundary conditions

It is common that different scientific applications have different boundary conditions, such as Dirichlet boundary condition, Neumann boundary condition, periodic boundary condition, etc. Thus we need to add various boundary conditions support for *HiProp*. For a triangular mesh library, the physical boundary conditions fall into two classes: periodic and non-periodic. Our previous discussion about the algorithms for building parallel information list and parallel updating list are based on the assumption of non-periodic boundary conditions, which mean, two points are considered to be the same point globally and the parallel information list must be built if and only if they share the same physical position. For a triangular mesh with periodic boundary conditions, two points could be logically the same point if there exists a shift between them in the direction of the periodic boundary. Fig. 6.6 is an example of a 2D triangular mesh on single processor with periodic boundaries in both x and y directions. Although the four corners of the mesh are far away from each other, they are considered to be the same point logically and one of them has to be assigned as the master point.

For supporting periodic boundary conditions, we first generalize the neighboring processor list so that a processor p could be the neighbor of itself. We also add shift tags $\mathbf{s} = (s_0, s_1, s_2)$ for neighboring processors and each parallel information node depending on the boundary conditions, which could be either periodic or regular inter-processor boundary conditions.

Suppose we have two processors p, q and the overall domain length is $\mathbf{D} = (D_0, D_1, D_2)$ in three directions. Point $\mathbf{v} = (v_0, v_1, v_2)$ on p and point

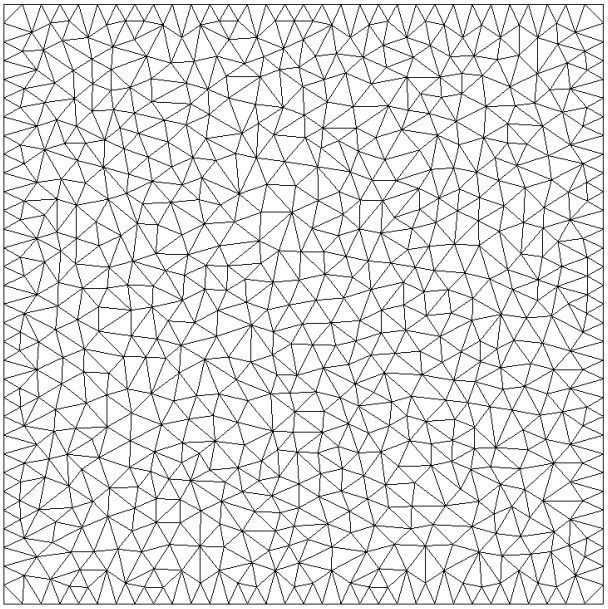

Figure 6.6: A 2D triangular mesh with periodic boundary conditions.

$\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \hat{v}_3)$ on q are considered to be the same point globally if and only if

$$v_i - \hat{v}_i = s_i D_i, \forall i = 0, 1, 2$$

where s_i could only equal to 0 or ± 1. We add this shift tag $\mathbf{s}$ to each parallel information node for points. The shift tags for triangles could be derived from connectivity. We also add the shift tags to the neighboring processor list so that we do not need to go through each possible shift every time. Note that multiple shifts could exist for a single neighboring processor. Taking Fig. 6.6 as an example, the processor 0 has one neighboring processor which is processor 0 itself and 8 different shift tags are associated with this neighboring processor.

The algorithm for building parallel information list also needs to be gen-

eralized for periodic boundary conditions. The simple comparison between the ranks of processors is not valid anymore and local indices have to be used for recognizing the master nodes. Assuming point $\mathbf{v}$ point $\hat{\mathbf{v}}$ on the same processor p are considered to be the same point with shift, we use local indices and the point with the lower local index is the master. Fig. 6.7 shows the visualization of the point types of the triangular mesh with periodic boundary conditions in Fig. 6.6. We use yellow, green and blue to denote INTERIOR, OVERLAY and GHOST points respectively.

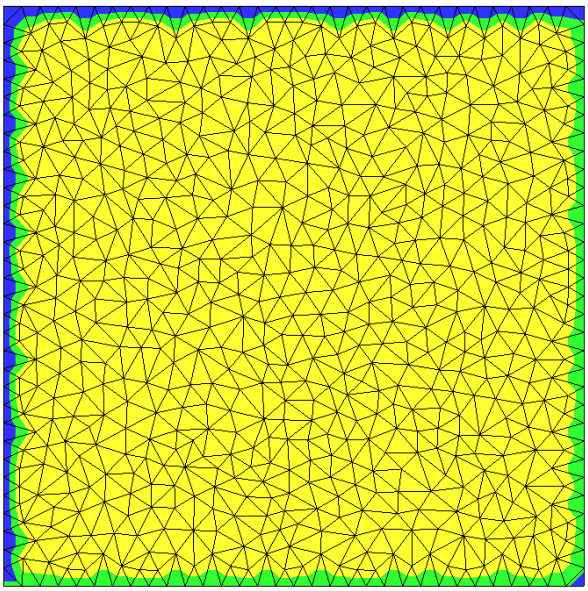

Figure 6.7: Point types of a 2D triangular mesh with periodic boundary conditions.

6.2 Ghost Exchange Algorithm

For using this parallel mesh library on mesh algorithms and parallel numerical PDE solvers based on domain decomposition, it is necessary for each processor to have the ability of gathering ghost triangles from neighboring processors. We describe here two types of ghost exchanging algorithms based on connectivity and domain decomposition. The first is required for parallel high-order mesh algorithms and the second one is necessary for coupling with numerical PDE solvers based on domain decomposition and *FronTier* which is also based on domain decomposition. The parallel information lists for both points and triangles need to be updated while exchanging the ghosts. In the following we first discuss the algorithm for building ghost triangles based on connectivity and the second ghost exchange algorithm shares the major part for communicating triangles and differs in the way of collecting ghost triangles.

First we give a general description of the algorithm for exchanging a n-ring ghost triangles on all processors:

1. For each neighboring processor, collect the vertices shared between the two processors.

2. Collect the n-ring buffer triangles and points based on the shared vertices for send. We only keep triangles not existing on neighboring processors and corresponding points in the buffer.

3. Exchange the ghost entities with parallel information lists.

   3(a) For each point and triangle to be sent, add the parallel information

node for each neighboring processor with an unknown ID.

3(b) Send the points and triangles collected in Step 2 to neighboring processors with the parallel information lists.

3(c) Receive the ghost buffers from neighboring processors and attach to the local mesh. The common points could be recognized using parallel information nodes associated.

3(d) Update the parallel information lists on the master processors.

3(e) Update the parallel information lists for all entities from their master processors to slave processors.

4. Update the neighboring processors list, opposite half edge structure and incident half edge structure.

For the mesh partition given in Fig. 6.3, the result mesh with fully updated parallel information lists after building a 1-ring ghost is given in Fig. 6.8.

The major complication of this algorithm comes from step 3 where parallel information lists are communicated three times for each entity. In the 1-ring ghost exchange example shown in Fig. 6.3 and Fig. 6.8, we need to send triangle t_1 on processor 1 to both processor 0 and 2. In Fig. 6.9 we could see that before processor 1 send the triangle to 0 and 2, it could not get the future local index of triangle t_1 on two other processors in advance and has to keep the initial value as unknown. When the processors 0 and 2 receive the triangle after communication ①, they assign local indices t_4 and t_7 for that triangle separately but still do not have the parallel information for each other. Then in communication ② we use the IDs t_4 and t_7 to update processor 1 and now

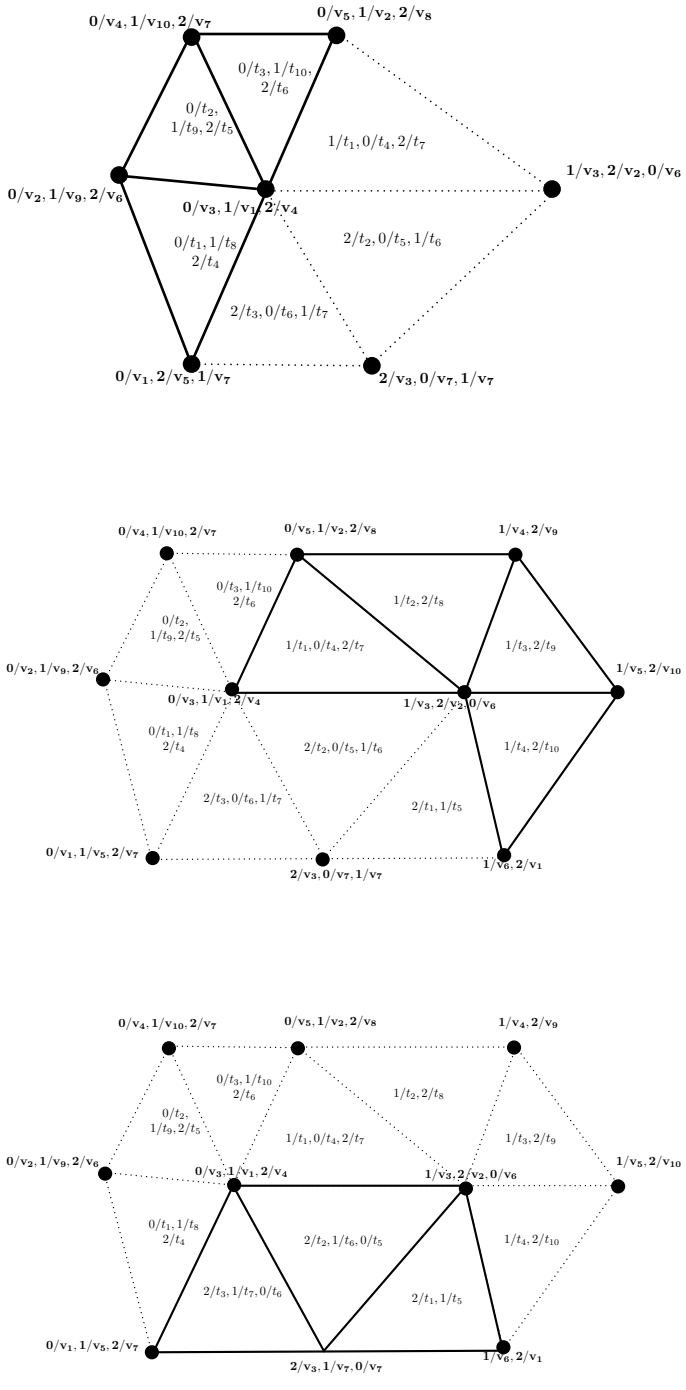

Figure 6.8: 1-ring ghost with fully updated parallel information lists for processor 0 (upper), processor 1 (middle) and processor 2 (lower).

processor 1 has a complete parallel information list for triangle t_1. In the end the whole list is sent back to processors 0 and 2 by communication ③ for keeping consistency of the lists on all processors.

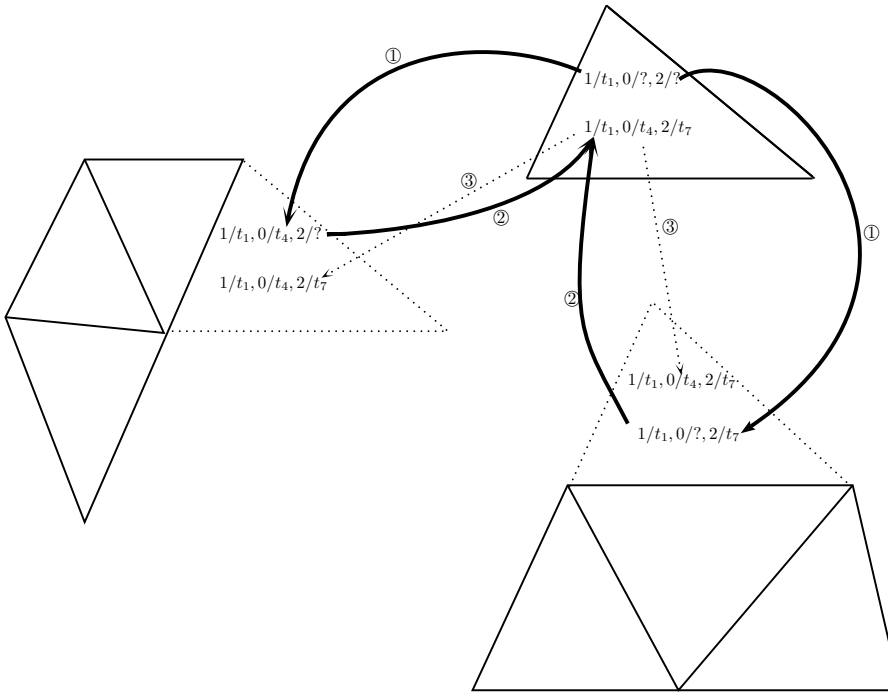

Figure 6.9: Updating the parallel information list for single ghost triangle.

Another complication lies in step 3(c) where ghost buffers are attached to local submeshes. In step 2 we discard the triangles and corresponding points if they already exist on target processor. However it is possible that a same ghost point is gathered from different processors. In Fig. 6.8 the point v_8 on processor 2 is gathered from both point v_5 on processor 0 and v_2 on processor 1. Thus in step 3(c) before adding an point to the local submesh as a new point, we go through all the newly attached points to identify existing points

by their global ID pair which is kept in the master node. For non-clean meshes, same complication exists for triangles and it has to be taken care of. Fig. 6.10 is the example of a submesh with 2-ring ghost (red ones) triangles built. As described in Sec. 6.1.4, we also generalize the ghost exchange algorithm for supporting periodic boundary conditions with shift information. Fig. 6.11 shows an example of the 2-ring ghost triangles for a mesh with periodic boundary conditions in both x and y directions. We use the same convention as before for visualizing GHOST, OVERLAY and INTERIOR entities.

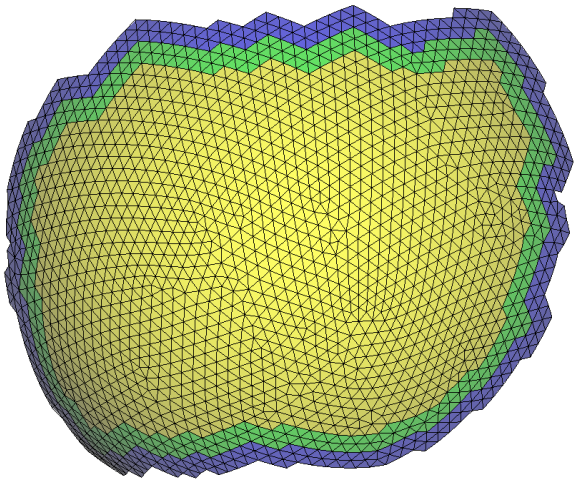

Figure 6.10: A submesh (part of a uniform sphere) with 2-rings of ghost triangles.

For building ghost triangles based on domain decomposition, only step 1 and step 2 in the algorithm needed to be changed. Instead of gathering ghost triangles based on shared vertices and connectivity, we gather the ghost triangles based on domain decomposition. Suppose we have domain controlled

by processor p denoted as $\mathbf{B}_p$, then on any other processor q, the triangles for sending to processor p is $\{t, t \cap \mathbf{B}_p \neq \emptyset\}$. Note that for the most general case we do not have any restrictions on the domains and an all-to-all communication cannot be avoided as it is possible that all processors are collecting from each other. However if we have some prior knowledge of the domain decomposition, point-to-point communications could be used. In Fig. 6.12 we show a comparison of the ghost triangles built from connectivity (left) and domain decomposition (right).

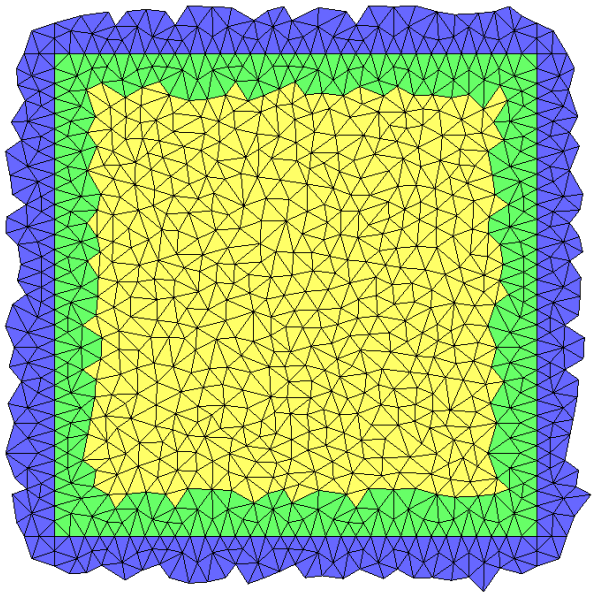

Figure 6.11: 2-rings of ghost triangles for a mesh with periodic boundary conditions.

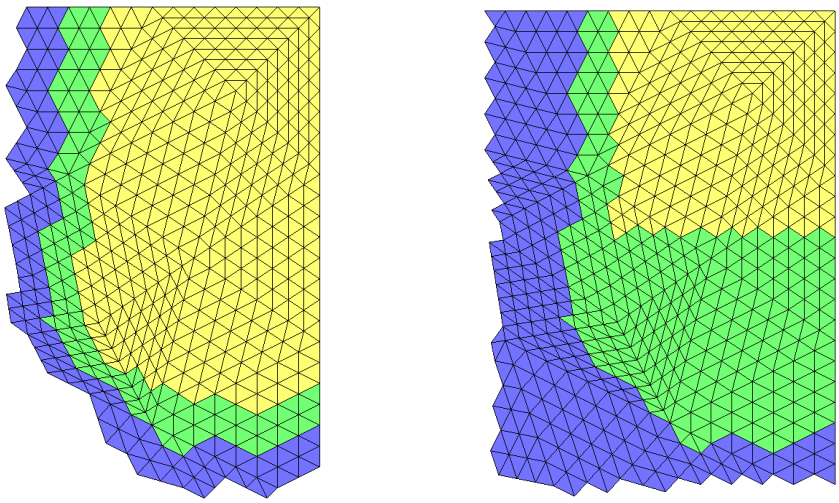

Figure 6.12: Comparison of ghost triangles built from connectivity (left) and domain decomposition (right).

6.3 Applications

In this section we give two applications of the parallel mesh library introduced in this Chapter. First, we present parallel high-order differential quantity calculations using the formula discussed in Sec. 3.1. It is a static mesh algorithm which means that even though ghost triangles are required to be built on each processors, the global mesh is not changed. Second, we use the library for developing a parallel high-order mesh smoothing algorithm. This is a non-static mesh algorithm as points are moved after each iteration. Thus not only we need to build ghost triangles for high-order polynomial fitting, parallel updating for various quantities is also needed. This parallel algorithm is based on the serial version first introduced in [13] and improved in [58] with a geometric limiter.

6.3.1 Parallel high-order normal and curvature calculation

We calculate normals and curvatures on a triangular mesh using the high-order polynomial fitting technique introduced in Sec. 3.1. For polynomial fittings on meshes, neighboring points selection is critical and we select points based on mesh connectivity, which is introduced in [31]. As discussed in [31], for a degree k fitting, a $\frac{k+1}{2}$-rings of neighborhood points should suffices. Fig. 6.13 shows the neighborhood definitions of half-rings up to 2.5 rings [31].

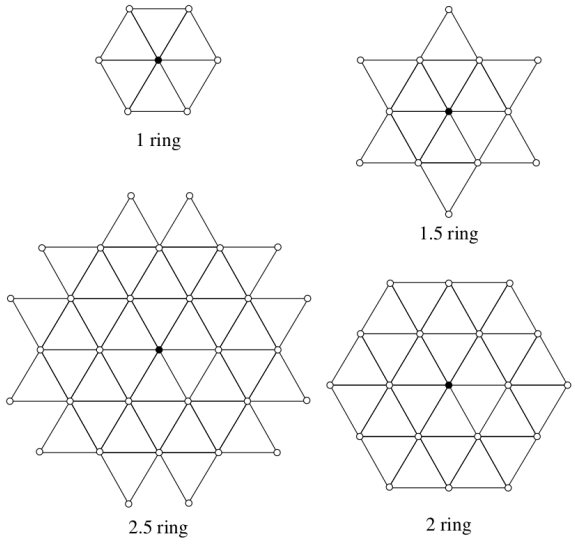

Figure 6.13: Schematics of 1-, 1.5-, 2-, and 2.5-ring neighborhood. Each diagram shows the neighborhood of the center (black) vertex [31].

For clean submeshes, neighborhood information is not sufficient for the points close to the partition boundary. Thus for a degree k fitting, we use the ghost exchange algorithm developed in Sec. 6.2 for building a $\frac{k+1}{2}$-rings of

ghost triangles on each processor. We then calculate normals and curvatures for all INTERIOR and OVERLAY points while the normals and curvatures on GHOST points are updated by other processors. In tables 6.1 and 6.2 we present the L_2 error and convergence rate for normals and mean curvatures of a sphere with radius 0.15 centered at $[0.35, 0.35, 0.35]$ under mesh refinement using both degree 3 and degree 5 fittings. We use single processor for level 1 and 2 meshes, 4 processors for level 3 and 4 meshes and 8 processors for level 5 mesh. The convergence rate is consistent with the results presented in [31], and for the same input mesh, the parallelized high-order normal and curvature calculation algorithm gives identical results to serial algorithms up to machine epsilon, which shows verification of the parallel mesh library.

Table 6.1: L_2 error and convergence rate for normals using degree 3 and degree 5 fittings on sphere.

	$d = 3, L_2$ error	$d = 3$, order	$d = 5, L_2$ error	$d = 5$, order
Level 1 mesh	3.151e-4	NA	6.624e-5	NA
Level 2 mesh	4.065e-5	2.9545	1.940e-6	5.0927
Level 3 mesh	4.211e-6	3.2710	5.010e-8	5.2751
Level 4 mesh	4.300e-7	3.2918	1.312e-9	5.2553
Level 5 mesh	4.869e-8	3.1426	3.675e-11	5.1579

6.3.2 Parallel mesh optimization

Based on the capability for building ghost triangles and updating attributes on ghost entities, we parallelized the high-order mesh smoothing algorithm introduced in [58].

Table 6.2: L_2 error and convergence rate for mean curvatures using degree 3 and degree 5 fittings on sphere.

	$d = 3, L_2$ error	$d = 3$, order	$d = 5, L_2$ error	$d = 5$, order
Level 1 mesh	0.21925	NA	0.02499	NA
Level 2 mesh	0.05487	1.9985	1.54e-3	4.0203
Level 3 mesh	0.01361	2.0114	9.42e-5	4.0311
Level 4 mesh	0.00340	2.0011	5.91e-6	3.9945
Level 5 mesh	0.00086	1.9831	3.73e-7	3.9859

First we need to build ghost triangles on each processor. For the high-order smoothing algorithm using a degree k polynomial fitting, a $\frac{k+1}{2}$ rings of points is needed for local stencils and an additional 1 ring is needed for using CMF or WALF to project the displacement to high-order continuous surface. For an even number k, $\frac{k+1}{2}$-rings + 1-ring is not the same as $\frac{k+1}{2}+1$-rings. Thus we always build a $\lceil\frac{k+1}{2}+1\rceil$-rings of ghost points before we go into the iterations for the mesh optimization algorithm. We also build the parallel updating lists for updating the GHOST point attributes. After this pre-processing for building ghost layers, we go into the iterations for the optimization algorithm and the flowchart is given in Fig. 6.14.

In general, we apply a laplacian smoothing on meshes with many folded triangles or very sharp angles and apply a variational smoothing otherwise. Then the new vertices are scaled to lie within the projection of one ring neighborhood and then projected to high-order surface reconstructed by WALF or CMF. A geometric limiter is then applied to poorly sampled area and a volume preserving smoothing is then applied. For details of each step see [58].

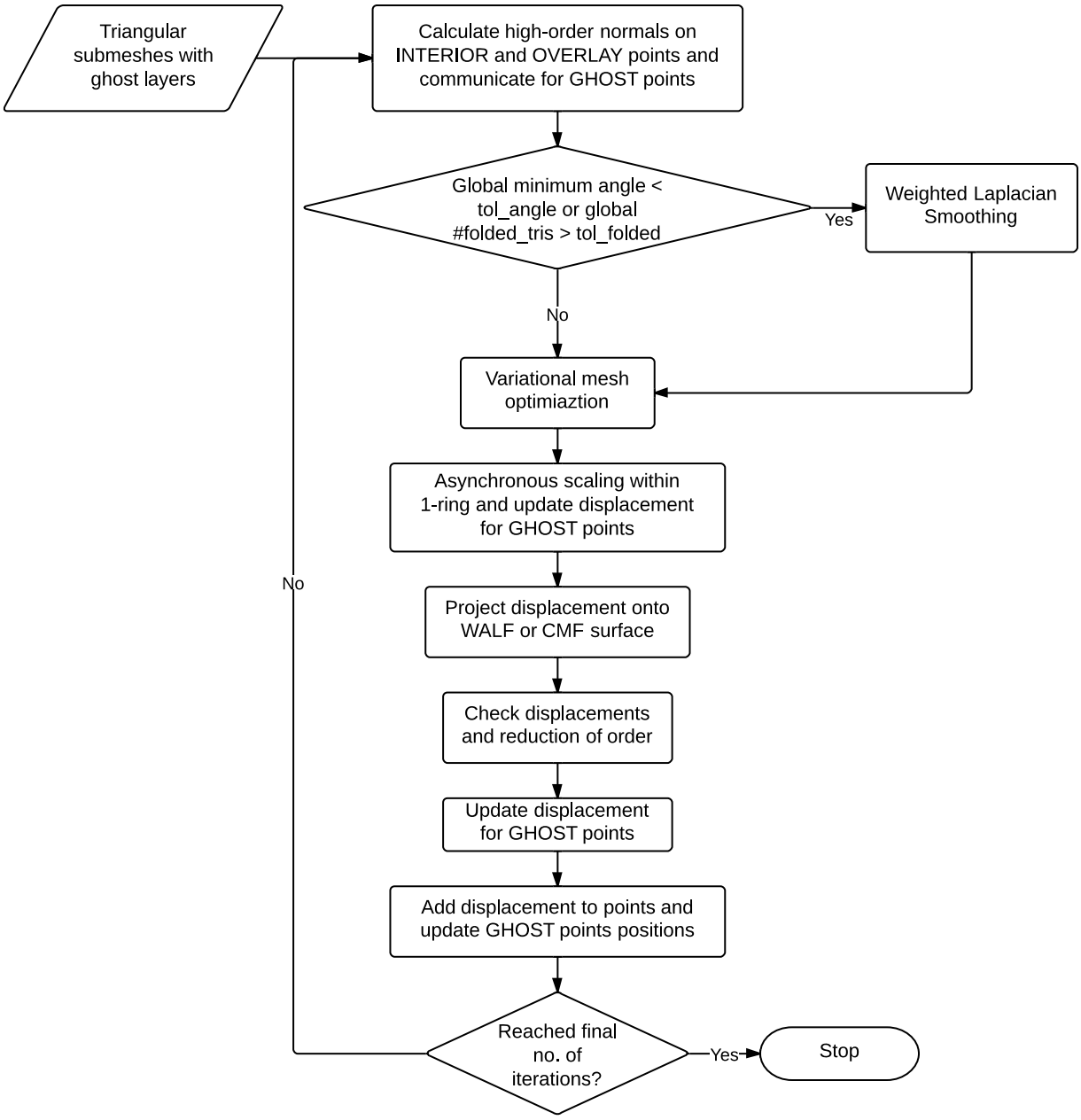

Figure 6.14: Flowchart for parallel optimization algorithm.

We present here two tests for parallel mesh optimization. The input mesh for the first example is a poor quality unit sphere distributed to 4 processors. The mesh is given in Fig. 6.15 and the difference colors on the mesh represent different partitions. We use a degree 2 fitting and CMF for reconstructing the high-order surface. We could see that after 15 iterations, the original mesh on the left is optimized to the mesh on the right. The histogram for triangle angle distributions before and after smoothing is given in Fig. 6.16. We could see from both visualization and angle distribution that the mesh has been improved significantly, which is consistent with the result presented in [58].

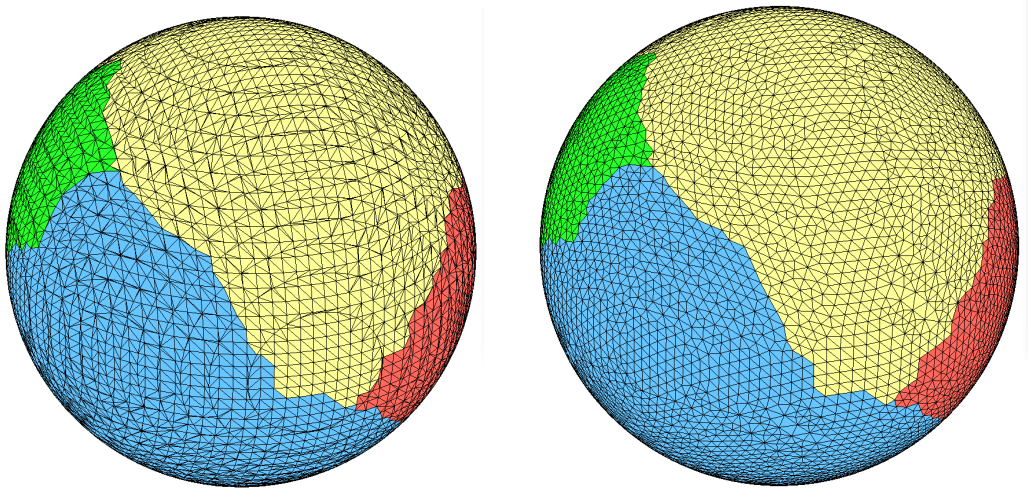

Figure 6.15: Parallel mesh optimization algorithm on unit sphere using 4 processors. The original mesh (left) and the optimized mesh (right).

The input mesh for the second example comes from the contactor simulation on 64 processors. The initial full mesh is given in Fig. 6.17 and the local visualization of the comparison before and after applying the parallel smoothing algorithm is shown in Fig. 6.18. We could see that the triangle quality is

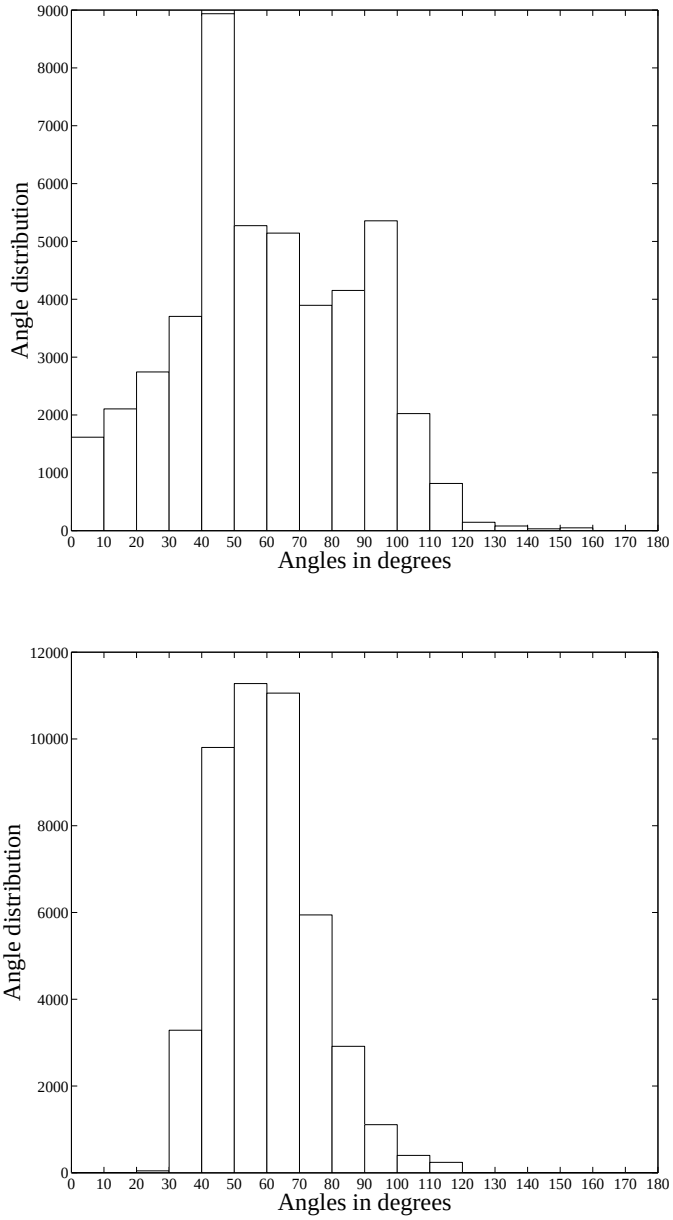

Figure 6.16: Angle distributions of initial mesh (upper) and the optimized mesh (lower).

improved and the mesh is smoothed due to the lapalacian smoothing we used in the algorithm.

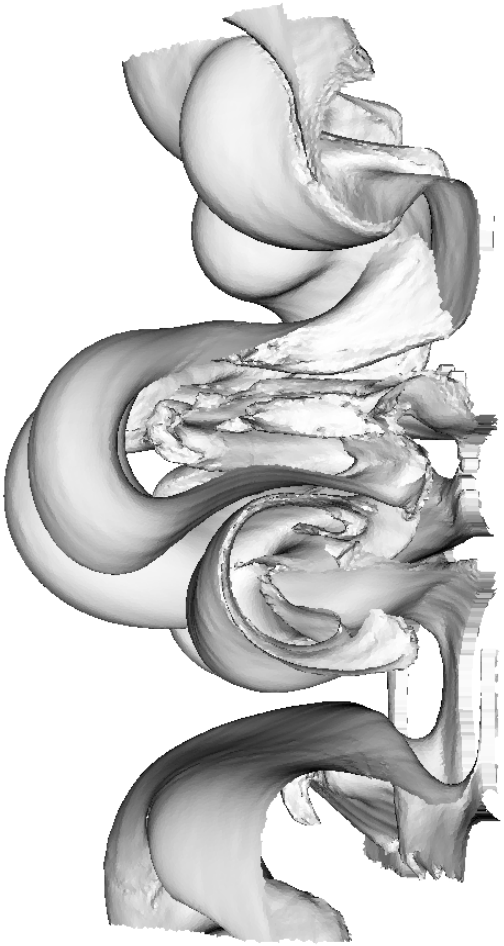

Figure 6.17: Input mesh on 64 processors for parallel smoothing test.

6.3.3 Parallel high-order functional propagation

In this section we present the convergence results of the high-order functional propagation algorithm introduced in Sec. 3.2 for both point positions

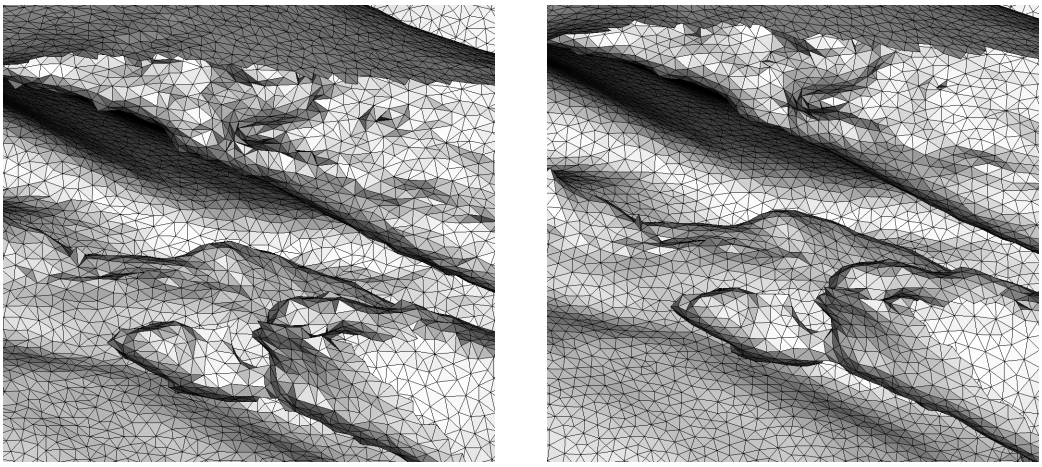

Figure 6.18: Comparison of mesh quality before (left) and after (right) smoothing in a local area.

and normals for two benchmark tests. We use a degree 2 fitting for both tests and the formulation for compact stencil discussed in Sec. 3.2.3 requires a 1-ring neighborhood for each point. With 1-ring ghost points built for each processor, the parallelization of this non-static mesh algorithm is straightforward as we only need to update the positions and normals for GHOST points at the end of each time step. For testing the rate of convergence, five meshes with different refinement level are used in both tests. We use single processor for level 1 and 2 meshes, 4 processors for level 3 and 4 meshes and 8 processors for level 5 mesh.

Sphere Rotation

First we present the numerical result for rotating a sphere centered at $[0.35, 0.35, 0.35]$. The sphere has a radius of 0.15 and the rotation velocity

field is given by

$$u(x, y, z) = -y,$$

$$v(x, y, z) = x,$$

$$w(x, y, z) = 0.$$

Time step Δt is set to $\pi/80$ thus a full revolution takes 160 times steps. Fig. 6.19 shows the error for point positions (left) and normals (right) in L_1, L_2 and L_∞ norms. The order of accuracy for point position is approximately 3.5 and the order of accuracy for normal is approximately 2.5.

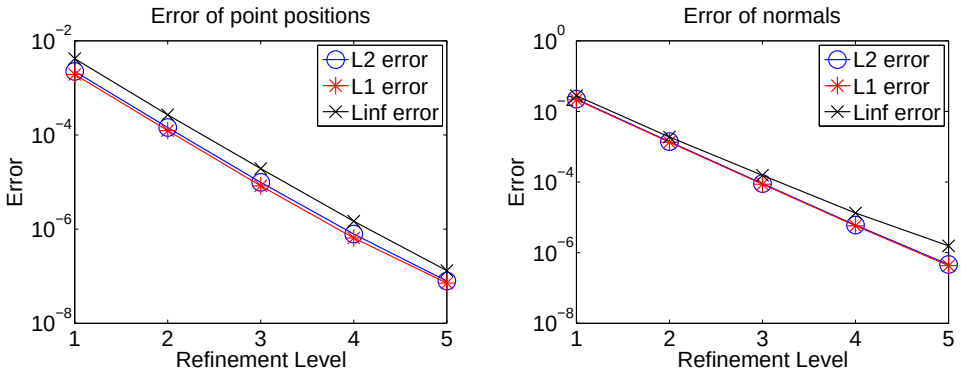

Figure 6.19: Errors of point positions (left) and normals (right) for sphere rotation test.

3D Deformation Velocity Field

We now consider the deformation of a sphere under a reversal vortex flow. As we currently have no mesh optimization and adaptivity combined, we do not stretch the mesh to the point with extremely high curvature where mesh

quality is bad enough to ruin the local polynomial fittings. The deformation velocity field of the vortex is given by:

$$u(x,y,z) = \begin{cases} 2\sin^2(\pi x)\sin(2\pi xy)\sin(2\pi z)\cos(\pi t/2) & \text{if } t < 0.3, \\ 2\sin^2(\pi x)\sin(2\pi xy)\sin(2\pi z)\cos(\pi(t+1.4)/2) & \text{if } t >= 0.3. \end{cases}$$

$$v(x,y,z) = \begin{cases} -\sin^2(\pi y)\sin(2\pi x)\sin(2\pi z)\cos(\pi t/2) & \text{if } t < 0.3, \\ \sin^2(\pi y)\sin(2\pi x)\sin(2\pi z)\cos(\pi(t+1.4)/2) & \text{if } t >= 0.3. \end{cases}$$

$$w(x,y,z) = \begin{cases} -\sin^2(\pi z)\sin(2\pi x)\sin(2\pi y)\cos(\pi t/2) & \text{if } t < 0.3, \\ -\sin^2(\pi z)\sin(2\pi x)\sin(2\pi y)\cos(\pi(t+1.4)/2) & \text{if } t >= 0.3. \end{cases}$$

We start from a sphere with radius 0.15 centered at $[0.35, 0.35, 0.35]$ at $t = 0$ and stop at $t = 0.6$ where the mesh is reversed to the original sphere. The mesh has largest deformation at $t = 0.3$ and the initial mesh versus mesh at $t = 0.3$ is shown in Fig. 6.20. We use a time step $\Delta t = 0.015$ thus it takes 40

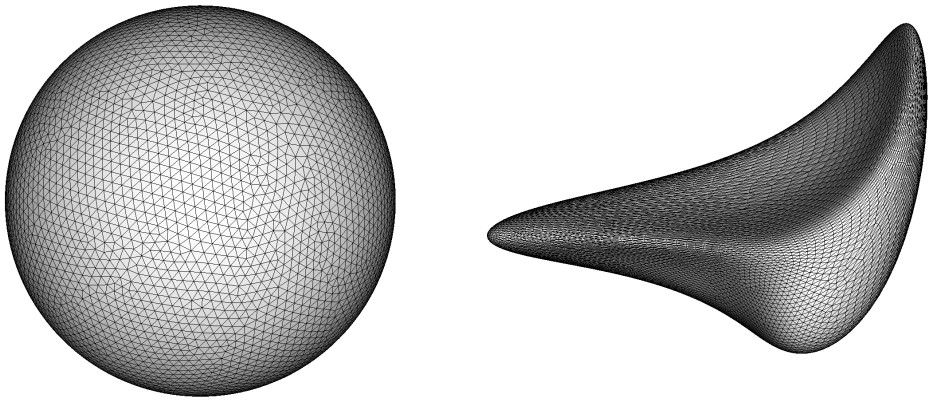

Figure 6.20: Initial sphere (left) and deformation at $t = 0.3$ (right).

time steps for the sphere to be deformed back to the initial position. Fig. 6.21 shows the error for point positions (left) and normals (right) in L_1, L_2 and L_∞ norms. The L_2 errors and convergence rates are given in Table 6.3 and

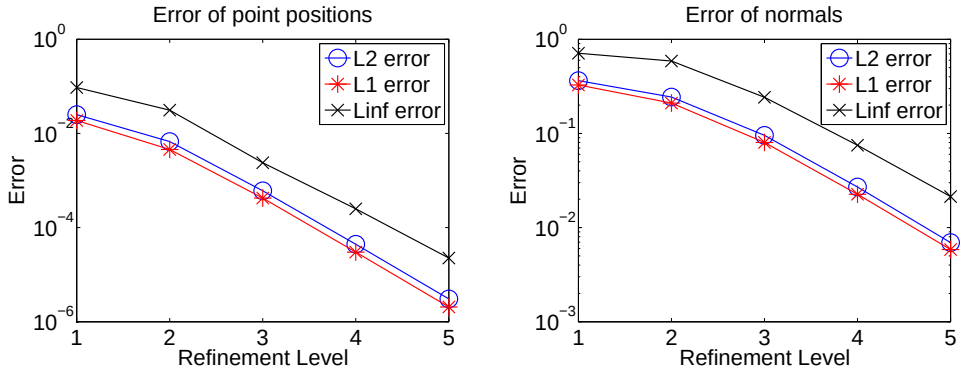

Figure 6.21: Errors of point positions (left) and normals (right) for deformation velocity field test.

the L_∞ errors and convergence rates are given in Table 6.4. We see that the numerical result is consistent with our theoretical rate of convergence for finer meshes. For coarse mesh the number of points in the local stencil is not sufficient for polynomial fitting in high curvature region. Thus a large error and low convergence rate is observed.

6.4 Coupling with *FronTier*

As described in the previous sections, our new interface library could be used as the basis for implementing various parallel triangular mesh algorithms. However it does not have all the functions needed for a complete interface tracking package, which should contains the following components:

Table 6.3: L_2 error for point positions and normal in deformation velocity test.

	Position error	Position order	Normal error	Normal order
Level 1 mesh	0.025022	NA	0.363804	NA
Level 2 mesh	0.006688	1.9036	0.243460	0.5795
Level 3 mesh	6.040e-4	3.4689	0.095352	1.3523
Level 4 mesh	4.419e-5	3.7729	0.026975	1.8216
Level 5 mesh	3.056e-6	3.8537	0.006930	1.9606

Table 6.4: L_∞ error for point positions and normal in deformation velocity test.

	Position error	Position order	Normal error	Normal order
Level 1 mesh	0.094246	NA	0.710556	NA
Level 2 mesh	0.031112	1.5989	0.587540	0.2743
Level 3 mesh	0.002378	3.7096	0.242237	1.2783
Level 4 mesh	2.499e-4	3.2506	0.074971	1.6920
Level 5 mesh	2.248e-5	3.4741	0.021347	1.8123

1. Point propagation with velocity field defined on the mesh or velocity field defined on the cells given by numerical fluid solvers.

2. Triangular mesh quality improvement without lossing accuracy.

   2(a) Mesh optimization which moves the points on a high-order surface without changing the number of points and the connectivity.

   2(b) Mesh adaptivity which includes edge flipping, edge splitting and edge contraction.

3. Untangling the meshes with self-intersections for a topologically correct surface.

Our current *HiProp* library could only handles step 1 and step 2(a) as described in Sec. 6.3.2. Thus for getting a complete interface tracking package we couple *HiProp* and *FronTier* through an interface between two libraries. Here we also replace the locally grid based tangle detection currently used in *FronTier* by applying an exact triangle-triangle intersection detection introduced in [25]. We only transfer data between the two libraries if a self-intersection is detected by the new tangle detection algorithm and the whole parallel information lists have to be rebuilt. The Locally Grid Based algorithm in *FronTier* is then used to resolve the tangle region locally. Fig. 6.22 gives the detail of the local reconstruction of the example of two merging spheres. Note that the small tangle shown in the left of Fig. 6.22 would not be detected using the old grid based intersection detection algorithm. Under the framework based on the interface between the two libraries, we would gradually migrate the

front-tracking package to the new library for a high-order and more robust interface library.

Using the interface between the two libraries, we are currently applying the paralle high-order mesh optimization algorithm introduced in Sec. 6.3.2 to the ongoing simulation of the incompressible Rayleigh-Taylor problem. We optimize the triangular mesh for each time step using iteration number of five and the data transfer between two libraries is called for each time step. A roughly 10 percent overall computation time is reported and a decrease in the computation time is expected in the future when the full front-tracking packaged is migrated to this new library.

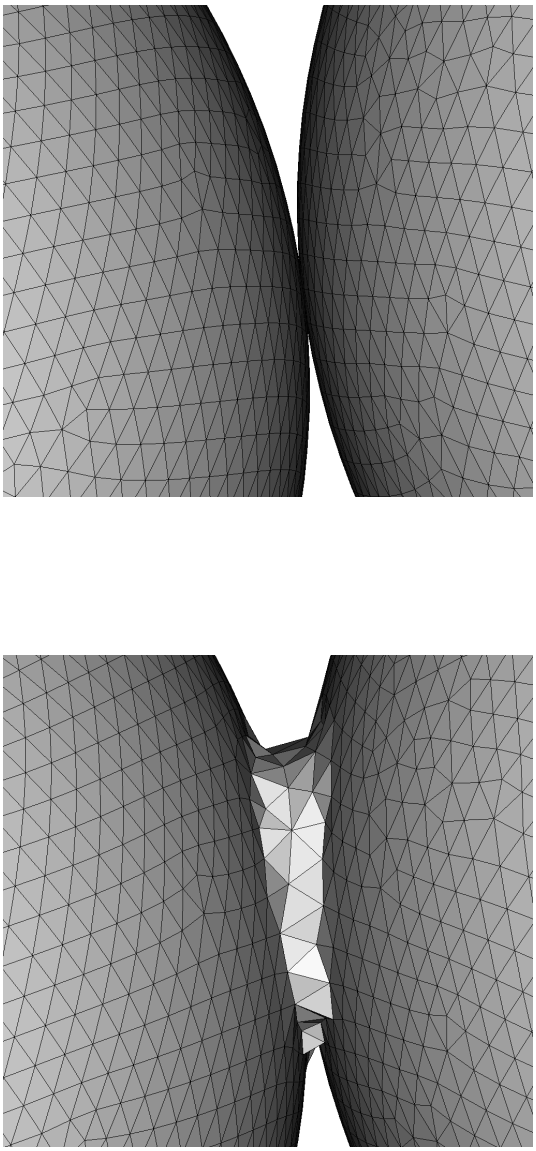

Figure 6.22: Tangle of two merging spheres (upper) and local recontruction result with mesh optimization (lower).

Chapter 7

Conclusion and Future Work

7.1 Conclusion

We use the front tracking method and the Immersed Boundary Method for the solution of the interface problem in two-phase incompressible flow. The fluid solver we developed is second order accurate for one-phase flow and first order accurate near the interface. High-order normal and curvature calculation based on local polynomial fitting was used for accurately coupling the interfacial tension into the fluid solver. We performed verification study for one-phase Taylor-Couette flow and presented numerical results which are consistent with both experiment for Taylor vortices and with linear analysis for the growth rate of the instability. Convergence results on turbulent TC flows were improved by adding a subgrid scale model. We use the solver developed on a complicated, two-phase mixing problem originated from the solvent extraction process in nuclear spent fuel reprocessing. A sector of the annular mixing zone of a high speed centrifugal contactor is used to study both the interfacial area and the droplet distribution for the two fluids. With

an unstable initialization, a stable late time configuration, with heavy fluid outside and light fluid inside, is reached after going through a complicated fully mixed regime. This stable configuration, which we call the centrifuge mode, is inconsistent with the experiment which has microstructure consisting of droplets of some 60 microns in size forming a honeycomb structure and a single connected phase. The discrepancy is mainly caused by the missing of the disjoining pressure existing in the thin lubricating wall between two droplets which are close to each other and by the algorithm which tend to accelerate the merge of separated droplets.

For handling complicated surfaces originated from fluid mixing problems such as the contactor problem, we develop a parallel triangular mesh library called *HiProp* which has a different communication model from the front tracking package *FronTier* and serves as the basis for various parallel mesh algorithms. Full parallel information lists for all entities are maintained in the new library and each entity has unique master processor which is the only processor that updates the attributes defined on that entity. Any parallel mesh operation would have an unique global output as the attributes on any entity could only be updated by its master processor. A ghost exchanging algorithm was developed for building ghost layers on each processors based on either connectivity or domain decomposition. The parallel information lists is updated when building ghosts for building multiple layers of ghosts of different types. The new library also supports periodic boundaries by adding a shifting tag to each parallel information node. The library is tested by implementing two high-order mesh algorithms based on the capability of building ghost trian-

gles. For both the parallelized static mesh algorithm for calculating differential quantities and the parallelized non-static mesh algorithm for optimizing the mesh for better quality, we present results consistent with the original serial algorithm, which verifies the uses of the new library. We also get an interface tracking package by combining *HiProp* with *FronTier*. The new library and the full interface tracking package could be used on many other problems and serves as the framework for gradually improving the front-tracking package.

Based on the same technique we used for high-order normal and curvature calculation and parallel library *HiProp*, we present the computational framework of a functional propagation which propagates local polynomial patches instead of isolated points. We solve the problem via the high-order local polynomial fitting and weighted least square formulation that generates not only high-order results for point positions, but also for differential quantities such as normals. We present numerical results for a rotation and a deformation velocity field. For a degree 2 fitting, higher than third order convergence is observed for point positions and second order is observed for normals. The benchmark tests do not have very large extortion as that would require high-order mesh optimization and high-order adaptivity for mesh quality control.

7.2 Suggestions for Future Work

As mentioned in Sec. 5.2.2, we are currently modeling disjoining pressure in the front tracking method to obtain experimentally consistent simulation results for multi-phase incompressible flows.

For getting a more accurate and robust interface tracking library, the

following steps needed to be done based on *HiProp*:

1. High-order parallel adaptivity algorithm for better mesh quality.

2. A robust and accurate topology operation for meshes with self-intersection.

3. Component calculation for any point in the space.

For task 1, we need to use parallel information lists on points and triangles to defined parallel information lists on edges as the edges are the main object for the adaptivity algorithm to work on. Task 2 could be tackled partially by first finding the exact point where the first topological change happens inside a time step using continuous collision detection (CCD). Then by changing the connectivity at that moment, untangle algorithm could be simplified by considering a single point of contact. Either the old locally grid based untangling could be used on the region surrounding the contact point or a more accurate grid free untangling algorithm which does topological operations directly on the triangles neighboring the contact point could be developed. Task 3 could either be resolved by a grid-based algorithm for checking components on grid lines or by a domain decomposition by tetrahedralizing each cell to find the connected components.

The high-order functional propagation could be improved by coupling with high-order mesh optimization and high-order mesh adaptivity of the algorithm to handle large deformations. For meshes with sharp features, the face offsetting method could be used for the points close to the edges or corners. Parameterizations other than local orthogonal projection parameterization such as spherical or cylindrical parameterizations could be used for the

regions with large curvature on the mesh. Lastly an implicit or semi-implicit time discretization could also be used on the PDE for propagating the local polynomial patches.

Bibliography

[1] T. Alumbaugh and X. Jiao. Compact array-based mesh data structures. In *Proceedings of 14th International Meshing Roundtable*, pages 485–504, 2005.

[2] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[3] M. Barad, P. Colella, D. T. Graves, P. Schwartz, B. van Straalen, and D. Trebotich. A cartesian grid embedded boundary method for the incompressible navier-stokes equations. *preprint submitted to Elsevier Science*, 2009.

[4] J. B. Bell, P. Colella, and H. M. Glaz. A second order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257, 1989.

[5] J. B. Bell, P. Colella, and H. M. Glaz. An efficient sceond-order projection method for viscous incompressible flow. *Proceedings of the Tenth AIAA Computational Fluid Dynamics Conference*, AIAA:360, 1991.

[6] W. Bo. *Applications of 3D Front Tracking to Multi Phase Fluid*. PhD thesis, State Univ. of New York at Stony Brook, 2009.

[7] W. Bo, X. Liu, J. Glimm, and X. Li. A robust front tracking method: Verification and application to simulation of the primary breakup of a liquid jet. *SIAM J. Sci. Comput.*, Accepted for publication, 2011.

[8] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100:335–354, 1992.

[9] D. Brown, R. Cortez, and M. Minion. Accurate projection method for the incompressible Navier Stokes equations. *J. Comp. Phys.*, 168:464–499, 2001.

[10] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability.* Oxford University Press, Oxford, 1961.

[11] A. J. Chorin. Numerical solution of the Navier Stokes equations. *Math. Comp*, 22:745–762, 1968.

[12] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comp*, 23:341, 1969.

[13] B. Clark, N. Ray, and X. Jiao. Surface mesh optimization, adaption and untangling with high-order accuracy. In *Proceedings of 21st International Meshing Roundtable*, 2012.

[14] P. Colella, D. T. Graves, B.J. Keen, and D. Modiano. A cartesian grid embedded boundary method for hyperbolic conservasion laws. *J. Comput. Phys.*, 211:347–366, 2006.

[15] A. Davey. The growth of Taylor vortices in flow between rotating cylinders. *J. Fluid Mech.*, 14:336–368, 1962.

[16] S. Dong. Direct numercial simulation of turbulent Taylor-Couette flow. *J. Fluid Mech.*, 587:373–393, 2007.

[17] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, and L. Wu. A simple package for front tracking. *J. Comput. Phys.*, 213:613–628, 2006.

[18] V. Dyedov, N. Ray, D. Einstein, X. Jiao, and T. J. Tautges. Ahf: Array-based half-facet data structure for mixed-dimensional and non-manifold meshes. In *Proceedings of 22nd International Meshing Roundtable*, pages 445–464, 2014.

[19] C. R. Ethier and D. A. Steinman. Exact fully 3d Navier-Stokes solutions for benchmarking. *Int. J. Numer. Meth. Fl.*, 19:369–375, 1994.

[20] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[21] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot. A dynamic subgrid scale eddy viscosity model. *Phys Fluids A*, 3:1760–1765, 1991.

[22] J. Glimm, M. J. Graham, J. W. Grove, X.-L. Li, T. M. Smith, D. Tan, F. Tangerman, and Q. Zhang. Front tracking in two and three dimensions. *Comput. Math. Appl.*, 35(7):1–11, 1998.

[23] J. Glimm, J. W. Grove, X.-L. Li, K.-M. Shyue, Q. Zhang, and Y. Zeng. Three dimensional front tracking. *SIAM J. Sci. Comp.*, 19:703–727, 1998.

[24] J. Glimm, J. W. Grove, X.-L. Li, and D. C. Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM J. Sci. Comp.*, 21:2240–2256, 2000.

[25] P. Guigue and O. Devillers. Fast and robust triangle-triangle overlap test using orientation predicates. *Journal of Graphics*, 8(1):25–42, 2003.

[26] C. Hirt and B. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[27] K. Ito and Z. Li. Interface conditions for stokes equations with a discontinuous viscosity and surface sources. *Applied Mathematics Letters*, 19:229–234, 2006.

[28] D. Jamet, D. Torres, and J. U. Brackbill. On the theory and computation of surface tension: the elimination of parasitic currents through energy conservation in the second-gradient method. *J. Comput. Phys.*, 182:262–76, 2002.

[29] X. Jiao. Face offsetting: a unified framework for explicit moving interfaces. *J. Comput. Phys.*, 220:612–625, 2007.

[30] X. Jiao and D. Wang. Reconstructing high-order surfaces for meshing. In *Proceedings of 19th International Meshing Roundtable*, 2010.

[31] X. Jiao and H. Zha. Consistent computation of first- and second-order differential quantities for surface meshes. In *Proceedings of the ACM Solid and Physical Modeling Symposium*, pages 159–170, 2008.

[32] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput Phys*, 147:60–85, 1998.

[33] J. Van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM J. Sci. Comput.*, 7:870, 1986.

[34] M. Kang, R. P. Fedkiw, and X.-D. Liu. A boundary condition capturing method for multiphase incompressible flow. *J. Sci. Comput.*, 15:323–360, 2000.

[35] J. Kim and P. Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *J. Comput. Phys.*, 59:308, 1985.

[36] Y. Kim and C. S. Peskin. Penalty immersed boundary method for an elastic boundary with mass. *Phys. Fluids*, 19:5], 2007.

[37] M. Kupiainen and B. Sjogreen. A Cartesian embedded boundary method for the compressible Navier-Stokes equations. *J. Sci. Comput.*, 41:94–117, 2009.

[38] M.-C. Lai and Z. Li. A remark on jump conditions for the three-dimensional Navier-Stokes equations involving an immersed moving membrane. *Applied Mathematics Letters*, 14:149–154, 2001.

[39] R. J. LeVeque and Z. L. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Num. Anal.*, 31:1019–1044, 1994.

[40] Z. Li. An overview of the immersed interface method and its applications. *Taiwanese J. Mathematics*, 7:1–49, 2003.

[41] Z. Li and K. Ito. *The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains.* Frontiers Appl. Math. 33, SIAM, Piladelphia, 2006.

[42] H. Lim, J. Iwerks, J. Glimm, and D. H. Sharp. Nonideal Rayleigh-Taylor mixing. *PNAS*, 107(29):12786–12792, 2010. Stony Brook Preprint SUNYSB-AMS-09-05 and Los Alamos National Laboratory preprint number LA-UR 09-06333.

[43] H. Lim, J. Iwerks, Y. Yu, J. Glimm, and D. H. Sharp. Verification and validation of a method for the simulation of turbulent mixing. *Physica Scripta*, T142:014014, 2010. Stony Brook Preprint SUNYSB-AMS-09-07 and Los Alamos National Laboratory preprint number LA-UR 09-07240.

[44] X.-D. Liu, R. P. Fedkiw, and M. Kang. A boundary condition capturing method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 160:151–178, 2000.

[45] X. F. Liu, E. George, W. Bo, and J. Glimm. Turbulent mixing with physical mass diffusion. *Phys. Rev. E*, 73:056301, 2006.

[46] X. F. Liu, Y. H. Li, J. Glimm, and X. L. Li. A front tracking algorithm for limited mass diffusion. *J. of Comp. Phys.*, 222:644–653, 2007. Stony Brook University preprint number SUNYSB-AMS-06-01.

[47] R. Lueptow. Taylor-couette flow. *Scholarpedia*, 4(11):6389, 2009.

[48] T. Ma. *Large eddy simulation of variable density flows.* PhD thesis, University of Maryland, 2006.

[49] P. McCorquodale, P. Colella, and H. Johansen. A Cartesian grid embedded boundary method for the heat equation on irregular domains. *J. Comput. Phys.*, 273:620–635, 2001.

[50] D. S. Meek and D. J. Walton. On surface normal and gaussian curvature approximations given data sampled from a smooth surface. *Comput. Aid. Geom. Des.*, 17:521–543, 2000.

[51] J. Melvin, P. Rao, R. Kaufman, H. Lim, Y. Yu, J. Glimm, and D. H. Sharp. Atomic scale mixing for inertial confinement fusion associated hydro instabilities. *High Energy Density Physics*, 2012. Submitted. Stony Brook University Preprint SUNYSB-AMS-12-01 and Los Alamos National Laboratory Preprint LA-UR 12-21555.

[52] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[53] J. Nave, R. R. Rosales, and B. Seibold. A gradient-augmented level set method with and optimally local, coherent advection scheme. *J. Comput. Phys.*, 229:3802–3827, 2010.

[54] C. S. Peskin. Numerical analysis of blood-flow in heart. *J. Comput. Phys.*, 25:220–252, 1977.

[55] C. S. Peskin. The immersed boundary method. *Acta Numer.*, 11:479–517, 2002.

[56] S. Popinet and S. Zaleski. A front-tracking algorithm for accurate representation of surface tension. *Int. J. Numer. Meth. Fluids*, 30:775–93, 1999.

[57] R. C. Di Prima and H. L. Swinney. Instabilities and transition in flow between concentric rotating cylinders. *Topics in Applied Physics*, 45:139–180, 1985.

[58] N. Ray, T. Delaney, D. Einstein, and X. Jiao. Surface remeshing with robust high order reconstruction. *Engineering with Computers*, 2014. Accepted.

[59] N. Ray, D. Wang, X. Jiao, and J. Glimm. High-order numerical integration over discrete surfaces. 50(6):3061–3083, 2012.

[60] P. H. Roberts. The solution of the characteristic value problems. *Proc.R.Soc.*, 283:550–556, 1965.

[61] R. Scardovelli and S. Zaleski. Direct numerical simulaion of free-surface and interfacial flow. *Ann. Rev. Fluid Mech.*, 31:567–603, 1999.

[62] P. Schwartz, M. Barad, P. Colella, and T. Ligocki. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *J. Comput. Phys.*, 211:531–550, 2006.

[63] L. E. Scriven. Dynamics of a fluid interface: Equation of motion for newtonian surface fluids. *Chem. Eng. Sci.*, 19:98–108, 1960.

[64] J. A. Sethian. *Level Set Methods*. Cambridge University Press, 1996.

[65] V. S. Smeeton and D. L. Youngs. Experimental investigation of turbulent mixing by Rayleigh-Taylor instability (part 3). AWE Report Number 0 35/87, 1987.

[66] G. P. Smith and A. A. Townsend. Turbulent couette flow between concentric cylinders at large taylor numbers. *J.Fluid Mech*, 123:187–217, 1983.

[67] E. M. Sparrow, W. D. Munro, and V. K. Jonsson. Instability of the flow between rotating cylinders: the wide gap problem. *J.Fluid Mech.*, 20:35–46, 1974.

[68] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 77:146–154, 1994.

[69] Z. Tan, D. V. Le, Z. Li, K. M. Lim, and B. C. Khoo. An immersed interface method for solving incompressible viscous flows with piecewise constant viscosity across a moving elastic membrane. *J. Comput. Phys.*, 227:9955–9983, 2008.

[70] Z. Tan, D. V. Le, K. M. Lim, and B. C. Khoo. An immersed interface method for the incompressible Navier-Stokes equations with discontinuous viscosity across the interface. *SIAM J. Sci. Comput.*, 31:1798–1819, 2009.

[71] G. I. Taylor. Stability of a viscous liquid contained between two rotating cylinders. *Philos. Trans. R. Soc. London, Ser. A*, 223:289–343, 1923.

[72] H. Terashima and G. Tryggvason. A front-tracking/ghost-fluid method for fluid interface in compressible flows. *J. Comput. Phys.*, 228:4012–4037, 2009.

[73] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169:708–759, 2001.

[74] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, imcompressible, multi-fluid flows. *J. Comput. Phys.*, 100(1):25–37, 1992.

[75] J. Walowit, S. Tsao, and R. C. DiPrima. Stability of flow between arbitrarily spaced concentric cylindrical sur faces including the effect of a radial temperature gradient. *J. Appl. Mech.*, 31:585–593, 1964.

[76] D. Wang, B. L. Clark, and X. Jiao. An analysis and comparison of parameterization-based computation of differential quantities for discrete surfaces. *Comput. Aid. Geom. Des.*, 26:510–527, 2009.

[77] S. Wang, R. V Samulyak, and T. Guo. An embedded boundary method for elliptic and parabolic problems with interfaces and application to multi-material systems with phase transitions. *Acta Mathematica Scientia*, 30B(2):499–521, 2010.

[78] S. T. Wereley and R. M. Lueptow. Azimuthal velocity in supercritical circular couette. *Experiments in Fluids*, 18:1–9, 1994.