Parallel processing of Eulerian-Lagrangian, cell-based adaptive method for moving boundary problems

by

Chih-Kuang Kuan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2013

Doctoral Committee:

Professor Ken Powell, Co-Chair
Professor Wei Shyy, Co-Chair
Assistant Professor Eric Johnsen
Assistant Professor Krzysztof Fidkowski
Associate Professor Yin Lu Young

© Chih-Kuang Kuan, 2013
All Rights Reserved

To my parents

Acknowledgements

To Professor Shyy, I would like to thank you, for your guidance and support in the past five years. You always give me a lot of freedom regarding the direction I wanted to take with my researches. When I was in the mud of endless debugging, you reminded me of the research goals and helped me structure my studies. I also want to thank you for the professional training you gave me, and thoughtful attitudes I learned from you. I still remember you saying in a group meeting years ago that the relationship between an advisor and a PhD student is unique. It is much more than a simply mentoring in professions. It is a life-long connection.

I would like to thank my co-chair Professor Powell for his guidance in the final year of my study. I am also grateful for Professor Johnsen, Professor Young, and Professor Fidkowski for serving on my defense committee and helping me finalize the last part of my study.

Table of Contents

List of Figures

List of Tables

List of Symbols

α	Thermal diffusivity
β	Thermal expansion coefficient
c	Heat capacity of liquid material
C	Color function of Volume of fluid method
D	Characteristic size of cylinder/sphere
D_0	Diameter of the initial droplet/bubble
F	Distance function of level-set method
g	Gravitational acceleration
h	Thickness
$I_{p,i}$	Overlapping zone of partition *p* with neighbor partition *i*
k	Wavenumber
K	Thermal conductivity
κ	Curvature
L	Lagrangian domain
L_p	Lagrangian partition of processor *p*
Ls_p	Lagrangian subdomain of processor *p*
N	Size of a Eulerian grid of a problem
n	Number of processor involved in a computation
$\mathbf{n}$	Normal of Lagrangian interface
Ω	Eulerian domain
Ω_p	Eulerian partition of processor *p*
Ωs_p	Eulerian subdomain of processor *p*
σ	Surface tension coefficient
θ	Separation angle
μ	Viscosity of liquid
ρ, ρ_l, ρ_g	Density of certain fluid, liquid, or gas
T	Temperature
τ	Characteristic time scale
$\mathbf{V}$	Velocity vector
U_0	Characteristic velocity of a problem
Δx	Cartesian grid space at the finest level
$\mathbf{x}$	Coordinate of an Eulerian point
$\mathbf{X}$	Coordinate of a Lagrangian point

List of Abbreviation

AMR:	Adaptive Mesh Refinement
VOF:	Volume of Fluid method
LS:	Level-set method
TC:	Taylor-Culick
RP:	Rayleigh-Plateau
RT:	Rayleigh-Taylor

Abstract

In this study, issues and techniques related to the parallel processing of the Eulerian-Lagrangian method for multi-scale moving boundary computation are investigated. The scope of the study consists of the Eulerian approach for field equations, explicit interface-tracking, Lagrangian interface modification and reconstruction algorithms, and a cell-based unstructured adaptive mesh refinement (AMR) in a distributed-memory computation framework. We decomposed the Eulerian domain spatially along with AMR to balance the computational load of solving field equations, which is a primary cost of the entire solver. The Lagrangian domain is partitioned based on marker vicinities with respect to the Eulerian partitions to minimize inter-processor communication. Overall, the performance of an Eulerian task peaks at 10,000-20,000 cells per processor, and it is the upper bound of the performance of the Eulerian-Lagrangian method. Moreover, the load imbalance of the Lagrangian task is not as influential as the communication overhead of the Eulerian-Lagrangian tasks on the overall performance. To assess the parallel processing capabilities, a high Weber number drop collision is simulated. The high convective to viscous length scale ratios result in disparate length scale distributions; together with the moving and topologically irregular interfaces, the computational tasks require temporally and spatially resolved treatment adaptively. The techniques presented enable us to perform original studies to meet such computational requirements. Coalescence, stretch, and break-up of satellite droplets due

to the interfacial instability are observed in current study, and the history of interface evolution is in good agreement with the experimental data. The competing mechanisms of the primary and secondary droplet break up, along with the gas-liquid interfacial dynamics are systematically investigated. This study shows that Rayleigh-Taylor instability on the edge of an extruding sheet can be profound at the initial stage of collision, and Rayleigh-Plateau instability dominates the longitudinal disturbance on the fringe of the liquid sheet at a long time, which eventually results in primary breakups.

Chapter 1.

Introduction

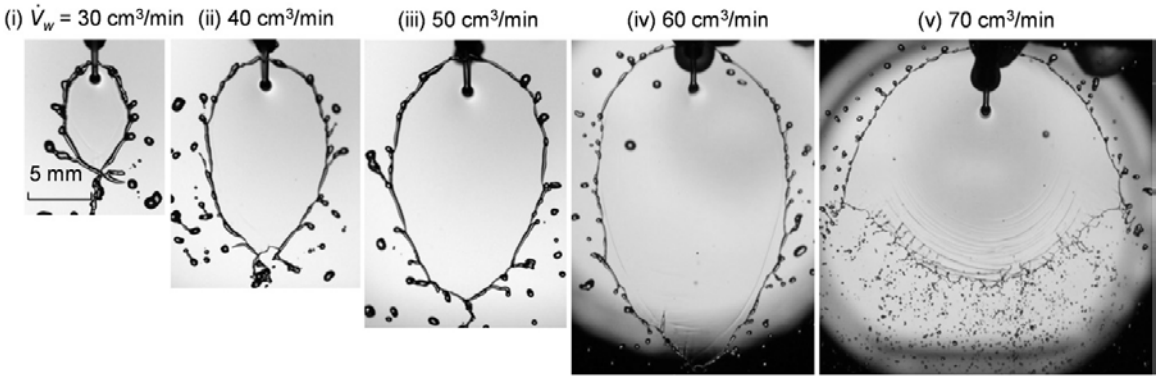
1.1 Background and motivation

Multiphase flows involving moving gas-liquid interfaces are abundant in nature, such as ocean waves, rain, melting ice. They are also ubiquitous in various industrial applications where working fluids are subjected to free surfaces. In many cases, non-linearity, instabilities, multiple-length-scale features, or vaporization and condensation accompany the transport of fluids. These processes are some of the most concerned mechanisms that engineers use to control fluid dynamics, facilitate functionalities of specific fluid devices, or improve efficiency of reacting flows. For example, from the fuel storage, delivering, injection to combustion phase, designs of the propulsion system of an astronautic vehicle cannot do without the consideration of the multiphase flow phenomena. Under microgravity condition, surface tension forces significantly affects the transportation of fuels in the pipes, storage, and delivering rate of fuel [1]. The boiling of cryogenic fluid due to heat absorption from the tank sidewall increases the pressure in a tank which results in the loss of available fuel [2]. Furthermore, the cavitation effects of cryogenic fluids at the transportation process is another issue for designing fluid machines [3]. The fuel injector of rocket engine combustors is one of the most critical components affected by multiphase phenomena. The size distribution of atomized droplets produced by the impingement of liquid jets determines the speed of vaporization,

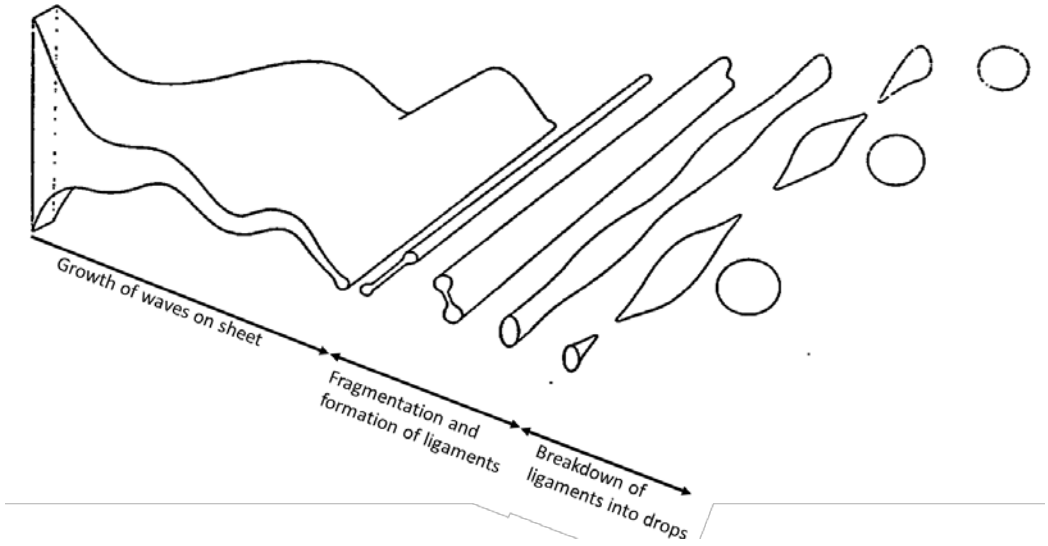
and thus the chemical reaction rate and combustion stability [4]. Understanding the interfacial dynamics of jet atomization is crucial for the performance improvement of the reacting flows. However, it is challenging to analyze the sophisticated morphologies of gas-liquid boundaries by experimental approaches since the interfaces are usually wiggled, corrugated, and unstable, as shown in Figure 1-1(a). The spreading interfaces generated by liquid jet impingement under high Weber and Reynolds number develop considerable characteristics in multiple length scales. The instabilities induced by the combined viscous, capillary and inertia forces on a moving sheet are highly complicated (Figure 1-1(b) [5]). These delicate structures result in shape and topological changes of the interface, with the latter accompanied by a cascading process of kinetic energy that drives an initially smooth interface to disintegration. The characteristic sizes of disturbance on the interfaces vary greatly. A full-scale assessment of all features of the fluid boundaries and flow field is challenging.

For moving boundaries under such conditions, direct numerical simulation can provide detail morphologies. An abundance of information is accessible through a series of simulations on a range of non-dimensional variables to help the verification of empirical scaling. However, numerical simulations of multiphase flows remain tenacious challenges in terms of computational cost and accuracy [6]. In addition to the computation of governing equations of continuous phases, more efforts are required to capture and identify fluid boundaries. Furthermore, the modeling task requires the treatment of distinct physical properties across the fluid boundaries, and interfacial dynamics, including surface tension and phase change effects. In order to perform satisfactory numerical simulation, interface evolution in time and space needs to be

resolved, which places high demand on the computational algorithm as well as on computational resources.



(a)



(b)

Figure 1-1. The structures of gas-liquid fronts (a) Paired liquid jet impingement at increasing jet velocity. Reprinted from Yamamura et al. [7] with permission from Elsevier. (b) Disintegration of planar liquid jet. Reproduced from Dombrowski [5].

The goal of the current study is to develop a computational platform to handle multi-scale simulation along with moving fluid boundaries with satisfactory parallel processing capabilities. Specifically, the previously developed techniques based on the

Eulerian-Lagrangian interface tracking method, presented by Singh and Shyy [8], Uzgoren, Sim, and Shyy [9], and Sim and Shyy [1], are further developed in a distributed-computation framework with a parallel adaptive mesh refinement technique to reduce the computational requirement. We document the challenges associated with the parallel implementation of the Eulerian-Lagrangian method and detail the cell-based unstructured adaptive mesh refinement techniques. We address the key components of the computational algorithm in the distributed computation context, including interface tracking based on the Lagrangian framework, incompressible Navier-Stokes computation in the Eulerian approach utilizing the Cartesian grid, communication between Eulerian and Lagrangian approaches, interface reconstruction techniques, and parallel cell-based unstructured adaptive mesh refinement (AMR).

1.2 Literature overview

1.2.1 Computational modeling of moving fluid boundaries

Numerical representations of fluid boundaries can be a cell-averaged scalar function to denote the volume fraction of a specific fluid, a distance function to describe boundary contour, or an explicit mesh as fluid boundaries. We use advection schemes to update the spatial distribution of the numerical representation of fluid boundaries. Nowadays, popular advection schemes to model multi-fluid boundaries are in three main families: the volume of fluid methods (VOF), the level-set methods (LS), and the front-tracking methods. For the purpose of distinguishing the challenges of parallel implementation and comparing these advection schemes with the Eulerian-Lagrangian interface tracking method used in present work, the review of these advection schemes focuses on issues related to parallel implementation. A comprehensive review of moving

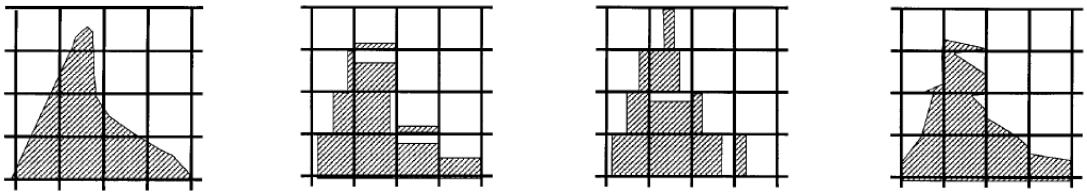
boundary modeling for fluid interfaces can be found in the book of Tryggvason et al. [6].

For VOF methods first introduced by Hirt and Nichols [10], one reconstructs the interface based on the cell-averaged volume fraction, named color function *C*, and then the color function is advected by the underlying, prescribed velocity field in a conservation form.

$$\frac{\partial C}{\partial t} + \frac{\partial (UC)}{\partial x} = 0 \tag{1}$$

Here *U* is the prescribed velocity field. The VOF scheme can naturally preserve the mass of a specific fluid if the velocity field is treated with a conservative formulation. Reconstruction is an operation to adjust the orientation of the line segments representing the interface to physically represent the fluid volume. An interface reconstruction method influences the accuracy of interface geometries. A VOF method without an appropriate reconstruction algorithm may produce unphysical interface distortion or breakup. Promising reconstruction algorithms such as the modified piecewise linear interface calculation (PLIC) [11, 12] originated by Youngs [13], least squares volume-of-fluid reconstruction algorithm of Pilliod and Puckett [14], cubic spline interpolation of Lopez et al. [15], and others extensions [16] have been proposed to improve the quality of fluid boundaries. However, the balance between the accurate reconstruction of interface shapes and the conservation of the key physical quantities such as volume and mass can be a challenge. Figure 1-2 shows reconstructed interfaces based on an initial volume fraction field with the Hirt-Nichols method [10], simple line interface calculation method (SLIC) of Noh and Woodward [17], and piecewise linear and interface calculation method (PLIC) of Youngs [13]. Although the advection of volume fraction is straightforward,

reconstruction methods for accurate representation of fluid interfaces dramatically increases the complexity of implementation, especially in 3D spaces. Topology changes of VOF method are handled in the advection scheme implicitly. The breakup and merge of interfaces need no additional treatment.



Actual fluid shape Hirt-Nichols [10] SLIC in X axis [17] PLIC [13]

Figure 1-2. Two-dimensional interface reconstruction based on the scalar field of volume fraction by Hirt-Nichols, SLIC, and PLIC methods. Reproduced from Rudman [11] with permission of John Wiley and Sons.

Level-set methods originally proposed by Osher and Sethian [18] define the interfacial boundaries on the zero level of a smooth distance function *F*, which evolves according to the velocity field by the advection equation.

$$\frac{\partial F}{\partial t} + U\frac{\partial F}{\partial x} = 0 \qquad (2)$$

Level set methods are relatively simple in terms of implementation. One can obtain accurate results when the interface is parallel to one of the coordinate axes. However, in cases where the interface is largely deformed, level set methods suffer from a loss of volume. Therefore, the interface is not simply propagated by fluids. A process to reinitialize interfaces at every time step is necessary to conform the gradient of distance function *F* with the phase boundaries [19]. The original level set method is attractive for its simplicity of interface-capturing scheme. Furthermore, its latter extensions add more complicity for improving the mass conservation, such as using the couple level-set VOF

method [20], and reducing spurious velocity due to surface tension computation [20-22].

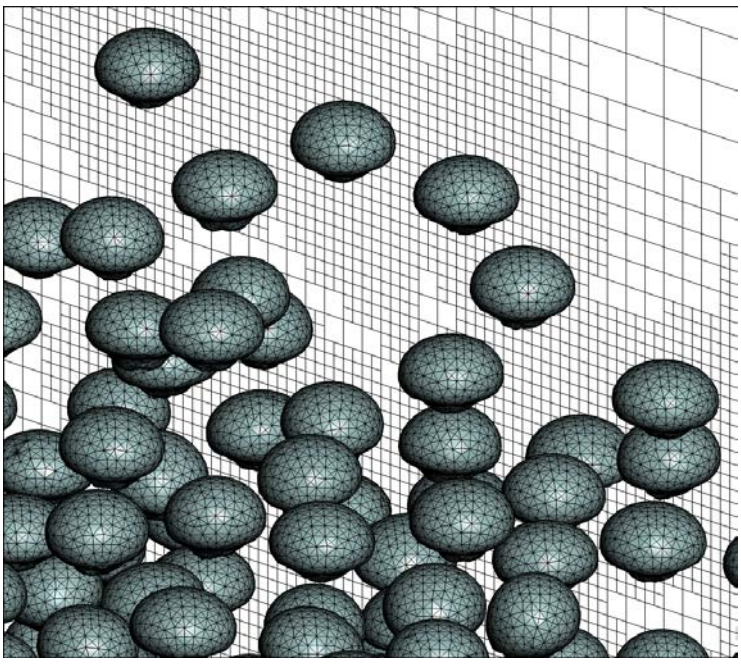


Figure 1-3. An illustration of a front-tracking method for a bubble swarming problem. Gas-liquid boundaries are represented by Lagrangian markers and triangular elements, while field equations are solved on the Eulerian grid (Cartesian grid). Computed by using the present adaptive Eulerian-Lagrangian interface tracking method.

Both VOF and level-set methods are the front-capturing method, which identifies interfaces by the procedure “reconstruction” according to the cell-averaged functions, and belong to the class of Eulerian methods [23]. Instead of using scalar function to locate interfaces, front-tracking methods use markers that move with the velocity field as a surface mesh to represent moving boundaries [24-28]. The set of markers are on a Lagrangian frame, and field equations are solved on the underlying Eulerian grid (Figure 1-3). Therefore, a front-tracking method stores variables on Eulerian and Lagrangian coordinates with a different data structure. Variables on the Lagrangian interfaces and Eulerian grid are coupled with cross-domain interpolation. The size and frequency of information exchanged between Lagrangian markers and the Eulerian grid depend on the

modeling formulation of fluid boundaries and procedures of solving field equations. Continuous interface methods model the fluid boundaries as smoothing transition of fluid properties at the boundaries [29, 30]. The spreading of material properties and force terms for the enforcement of boundaries condition demands more information of communication than sharp interface methods [26, 29, 31, 32]. In addition, the Lagrangian interface requires a remeshing technique to maintain the quality of the triangular surface mesh. Moreover, direct advection of interfaces does not change the interface topology. That is, the breakup and merging of interfaces requires special treatment on the connectivity of Lagrangian markers. Bo et al. presented a front-tracking technique with a local grid-based method to treat topological changes [33], but generally, complex interface deformation remains challenging for the front-tracking family. In summary, front-tracking methods have more algorithm complexities in three aspects: cross-domain data interpolation, Lagrangian remeshing, and topology change algorithm. However, the explicit representation of interfaces provides excellent accuracy of boundary geometries, such as curvature computation, which is one of its most appealing features. Such advantage is influential on the study of interfacial instability. Table 1-1 is a comparison between VOF, LS, and front-tracking methods in terms of numerical methods, algorithm, and recent advancements. We have to emphasize that recent algorithms have improved some of the disadvantages of the LS or VOF methods, such as the mass loss of the LS method and the numerical diffusion of the VOF methods. These additional techniques add much complexity to their conceptually-simple origins in 3D domain such that the gaps of algorithm complexity between these three types are not as large as before.

Table 1-1. A comparison of VOF method, level-set method, and front-tracking method on accuracy, algorithm complexity, data structure, recent advancement in multiphase flow applications, and parallel implementation.

	VOF	Level-set	Front-tracking
Development	Hirt and Nicols [10] Youngs (PLIC) [13] Rudman [11]	Osher and Sethian [18] Sussman et al. [19] Osher and Fedkiw [21]	Peskin and McQueen [24] Unverdi and Tryggvason [25] Udaykumar et al. [26] Ye et al. [27] Glimm et al. [28]
Features	Eulerian computation (Advection of volume fraction)	Eulerian computation (Advection of distance function)	Eulerian-Lagrangian computation (A mesh moves based on equation of motion.)
Advantage	Mass conservation	Robust computation for interface deformation and topological change	Accurate curvature computation
Concerns	Numerical diffusion for early versions such as SLIC method Merging and breakup of interfaces occur automatically.	Mass loss (non-CLSVOF type) Merging and breakup of interfaces occur automatically.	Implementation difficulties
Algorithm complexity	Medium: Reconstruction of interface in 3D	Medium: Re-initialization of interface	High: 1. Cross-domain interpolation 2. Lagrangian remeshing algorithm 3. Topology change algorithm
Data structure	As single phase solver	As single phase solver	Two sets of data structure
Recent advances	Popinet [34] - Balanced-force /continuum-surface-force surface-tension formulation and height-function curvature estimation Agbaglah et al. [35] - Octree adaptivity with VOF	Sussman [20] - Couple Level-set VOF (CLSVOF) method Yokoi [22] - Density-based CSF model Wang et al. [36] - A second order hybrid level set-volume constraint method	Tryggvason et al. [32] - A 3D front-tracking by solving one-set of governing equations Singh and Shyy [8] - A 3D volume-conservative interface tracking method Bo et al. [33] - Ghost fluid method with locally grid based reconstruction method

Parallel implementation	Agbaglah et al. [35] Espostiongaro et al. [37]	Sussman [38] Rodriguez et al. [39]	Kuan et al. [40, 41]

1.2.2 The Eulerian-Lagrangian interface tracking method

In the current study, the Eulerian-Lagrangian interface tracking method is a front-tracking method that uses a set of connected Lagrangian markers representing moving phase boundaries. The Lagrangian markers primarily move with the underlying velocity field on an Eulerian grid where field equations are solved. The Eulerian-Lagrangian method tracks interface explicitly, and provides outstanding in-cell interface resolution when compared with pure Eulerian methods such as the volume of fluid method and level-set method [8, 9, 32]. Due to explicit interface representation, it does not reconstruct interfaces from cell-averaged functions for which it provides accurate computation on morphology-related variables such as surface tension. The Eulerian-Lagrangian method presented in this work uses a continuous interface method to model a material jump around fluid boundaries and a sharp interface method to accommodate no-slip boundary conditions of solid boundaries [42]. Furthermore, it has a dynamic contact-line-force algorithm to model moving fluid-fluid interfaces on arbitrary solid boundaries. This Eulerian-Lagrangian method has been successfully applied to many practical engineering problems, such as binary droplet collision at low Weber number, cryogenic fuel sloshing in spacecraft fuel tanks, and free surface instability [1, 8, 42]. Moreover, accurate surface tension representation avoids spurious velocity as computations are carried out at the high Laplace number regime. It is an effective approach for moving boundary computations, but inherently it incurs significant challenges toward a parallel implementation. There are two necessary features to consider on the perspective of

parallelization. First, the Eulerian-Lagrangian method is a dual-domain technique. The Lagrangian data is stored in dynamic arrays on a triangular surface mesh, as opposed to flow data on an Eulerian frame. Partition strategies must consider the distribution of workload on both domains. Second, data on an Eulerian grid cell and a Lagrangian marker interact frequently in a computational time step. Strong data dependency in time between Eulerian and Lagrangian domains imposes restrictions on the choice of parallelisms for scalability.

Since the data structure of Eulerian methods is similar to traditional single-phase flow solvers, the parallel implementation of Eulerian methods is straightforward and usually accomplished by a spatial domain decomposition. Several large-scale geological simulators using VOF methods have been successfully applied on distributed memory machines [35, 37, 38]. Sussman presented a parallelized Cartesian grid solver using a coupled level-set/VOF method for flows in general geometries [38]. In contrast, only certain problems such as conditions with the uniform distribution of Lagrangian elements have applied to parallel Eulerian-Lagrangian frameworks. No studies have systematically discussed arbitrary moving boundaries with the consideration of domain decomposition due to case-dependent performance and the difficulty of parallel implementation. Capecelatro and Desjardins reported a particle-laden flow solver with a parallelism under the Eulerian-Lagrangian framework [43]. Darmana et al. presented an Euler-Lagrange model to track swarming bubbles in columns. They introduced a mirror domain parallelism to decompose Lagrangian particles while the Eulerian domain was uniformly partitioned along the z-axis [44]. Herrmann used the Eulerian method to capture interfaces and described the under-resolved liquid volume with Lagrangian particles [45].

These studies showed favorable parallel performance, which is likely due to the Lagrangian data being loosely coupled with the Eulerian data, and the Lagrangian objects being individual particles (no connectivity) and uniformly distributed in the Eulerian domain. However, for many practical problems, the Lagrangian objects' frequent interaction with the Eulerian grid and their spatial distribution is non-uniform with respect to the Eulerian domain. Nkonga and Charrier reported a parcel method for a dispersed spray in a turbulent flow of piston engines [46]. The Lagrangian parcels are used to describe dispersed spray particles moving on a two-dimensional Eulerian unstructured mesh. They discussed the load imbalance due to the non-uniform distribution of parcels in a domain and the result showed that the parallel efficiency is problem dependent. In summary, the challenges of the Eulerian-Lagrangian methods in terms of parallelization are load balance for both Eulerian and Lagrangian domains and communication strategies. A load-balanced parallelism for both Eulerian and Lagrangian computations may be possible only with a compromise in communication overhead. The parallelisms adopted determine the frequency and data size of cross-processor communication. Minimizing communication overhead may be achievable only by using a parallelism having an imbalanced computation load on one of the two domains. We discuss possible parallelisms for the Eulerian-Lagrangian method, and have chosen a spatial domain decomposition for the Lagrangian domain because of the concern with data locality.

1.2.3 Adaptive mesh refinement

This work incorporates the Eulerian-Lagrangian method with a parallel AMR technique to maintain grid efficiency and solution accuracy for multi-scale problems.

Dynamic grid adaptation methods provide the efficient management of grid resolution, but incur considerable challenges in parallel implementation, especially in the aspects of load balance and remeshing. The spatial range of adaptation and data structures generally categorize them as: block-AMR, cell-based tree AMR, and cell-based unstructured AMR. The inherent data structure and the scale of problems determine the performance of these AMR approaches. Moreover, the data structure adopted also affects the complexity of the algorithm and domain decomposition strategies.

Of the many parallel AMR methods proposed, the most popular one is the block-based AMR family. The block-based AMR applies finer grid blocks upon regions demanding higher resolution. Each individual block is typically a uniform Cartesian grid, which is highly structured. The entire domain is composed of overlapping multiple blocks and domain decomposition is based on block units, as shown in Figure 1-4. Domain decomposition is usually based on the block units. At runtime, the refinement blocks are continually applied over old coarser blocks [47-49]. *SAMRAI* [47] based on the Berger-Rigoutsos algorithm [50], which is one of the most popular block-based AMR libraries. An immersed boundary solver, *IBAMR* developed by Griffith et al. utilizes SAMRAI for multiphase flow simulations of blood vessels [30]. Zuzio and Estivalezes [51] used *PARAMESH* developed by MacNeice et al. [16] for the simulation of a two-phase interfacial flow. In general, adaptive blocks could be non-overlapping. The block-tree type AMR, reported by Jablonowski et al. [52] refines a set of grid block by bisecting a parent block in each coordinate direction to generate children blocks with higher level refinement, and demonstrates high scalability on thousands of CPU cores for solving MHD equations [53]. The block-based AMR has an adequate cache hit rate due to the

structured data format and the more regular partition boundaries when compared with a cell-based AMR, which may reduce the data size of cross-boundary communications. Moreover, a block-based AMR tends to over-refine the grid, which results in an exceeding demand of computation and memory. Data interpolation between overlapping blocks requires additional cross-processor communication in the flow solver phase.

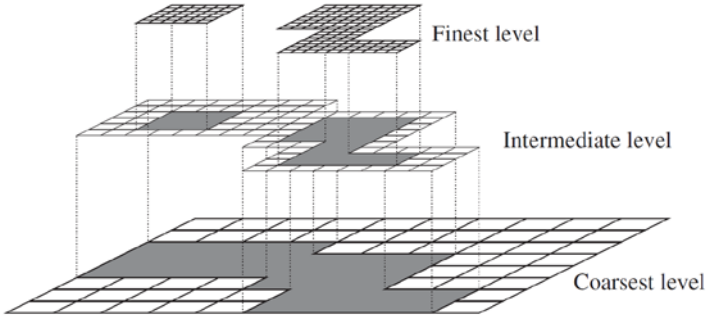


Figure 1-4. The patch block AMR. Reprinted from Gunney et al. [47] with permission from Elsevier.

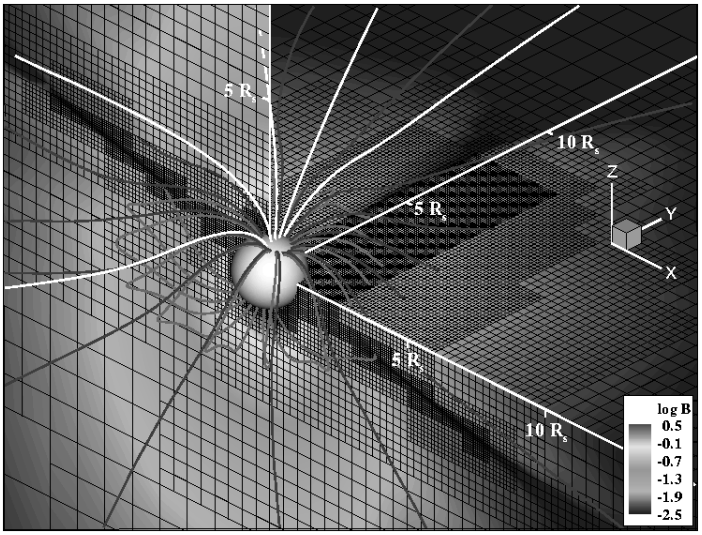


Figure 1-5. Simulation of corona on the magnetosphere of the earth by the block AMR method. Adopted from the book section of *Adaptive Mesh Refinement – Theory and Applications* [53] with permission from Springer.

Figure 1-5 is an example of a numerical simulation with a block-based adaptive

grid for a steady corona on the magnetosphere [53]. The grid along the sun to earth is adapted gradually by various levels of blocks and leads to 14 million cells in total.

In contrast, a cell-based tree AMR utilizes quad-tree or octree data structures to accommodate a hierarchy of sub-grids [35, 54, 55]. The tree structure forms a cell-based object containing information of the coarsest to finest levels of cells for a desired spatial resolution and parent-children relationship. Figure 1-6 is an example of a tree-based grid. Parent cell 1 in the tree structure directly accesses family members 5-8 which are unavailable for the other parent cells 2-4. The parallelism of tree-based AMR is usually the coarsest-level grid decomposition for simplicity [34, 35]. Decomposition is executed on the coarsest-level cells by balancing the total children cell count in each partition. A parallel sharp interface fix-grid method using a tree-based local refinement for moving boundary problems was developed by Udaykumar et al. [55]. Agbaglah et al. decomposed the computational domain based on the graph of the coarsest-level cells with weighting function to adjust load balance for dynamic AMR operation at run time [56]. Burstedde et al. presented library *p4est*, a scalable algorithm for parallel adaptive mesh refinement based on the concepts of forest octrees and the *z*-shaped space-filling curve between octrees [54]. Although the cell-based tree AMR provides flexible refinement regions and organized data structure, the computational speed is degraded due to frequent pointers traveling in-and-out of the tree hierarchy when searching for data.

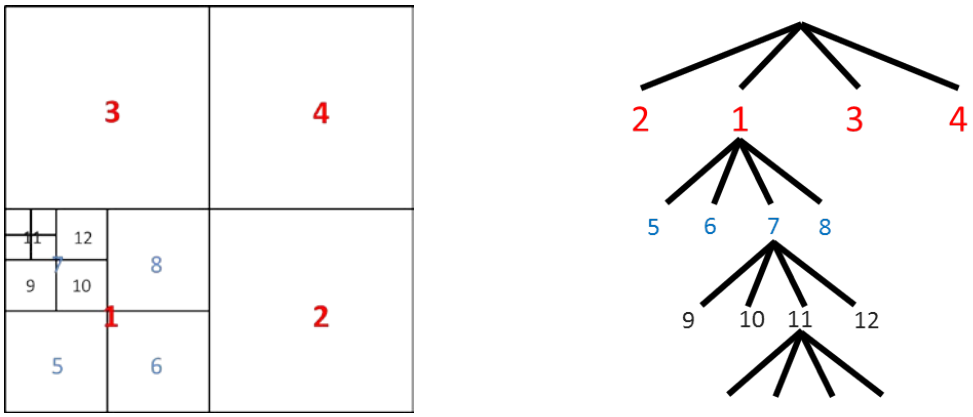


Figure 1-6. The quad-tree AMR and its corresponding hierarchy.

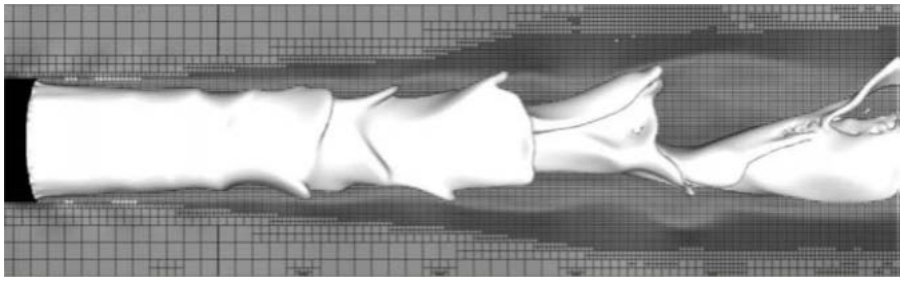


Figure 1-7. Simulation of primary atomization by VOF method and oct-tree AMR. Reprinted from Fuster et al. [57] with permission from Elsevier.

Various studies have proved the effectiveness of tree-based AMR. For example, Fuster et al. modeled primary atomization by VOF method with octree AMR. The surface instabilities on liquid-gas boundaries induced by aerodynamic shear stress were captured, as shown in Figure 1-7 [57]. The equivalent grid size of a uniform grid with the same resolution of this computation requires 512x128x128 points in 3D.

While both block-based and cell-based tree AMRs may tend to waste data storage and computing power by over-refining the grid or maintaining hierarchy information from the coarsest to the finest cells. A cell-based unstructured AMR, which has no hierarchy information, can be an alternative in merits of spatial flexibility and compact data storage. It shows outstanding performance in serial computations [1, 8, 42].

ParFUM is a typical example in this category [58]. The data is unstructured, which means the grid is an unstructured graph as shown in Figure 1-8. An explicit list of neighbor vertices is required to maintain graph connectivity. The development of a scalable parallel AMR for these methodologies is challenging, especially for a large number of CPU computation. This is due to the explicit storage of the global graph. Choices of domain decomposition approach may be limited due to the absence of a systematic grid structure. Using heuristics partition libraries is a relatively popular approach for accomplishing the partitioning of the unstructured AMR [58, 59]. A cell-based unstructured AMR has flexible refinement regions and better grid efficiency in terms of performance, especially for problems with complex moving boundaries. It has a consistent data-fetching rate and no level-level interpolations such that the performance of field equation solvers is independent of the number of refinement level. Using heuristic partition libraries for load balance is favorable in complex domains. The current work adopts the cell-based unstructured AMR method. We believe the cell-based unstructured AMR has the benefits on the aspect of solver performance as listed below.

(1) There is no grid hierarchy, nesting, overlapping, and no level-level interpolation in a time step, which means no cross-processor communication is needed due to inter-level interpolation.

(2) Data fetching rate is constant for all levels of grid.

(3) Load balance is independent of the number of refinement level.

(4) Flexible refinement provides better grid efficiency for complex interface.

A comparison of different AMR approaches is summarized in Table 1-2.

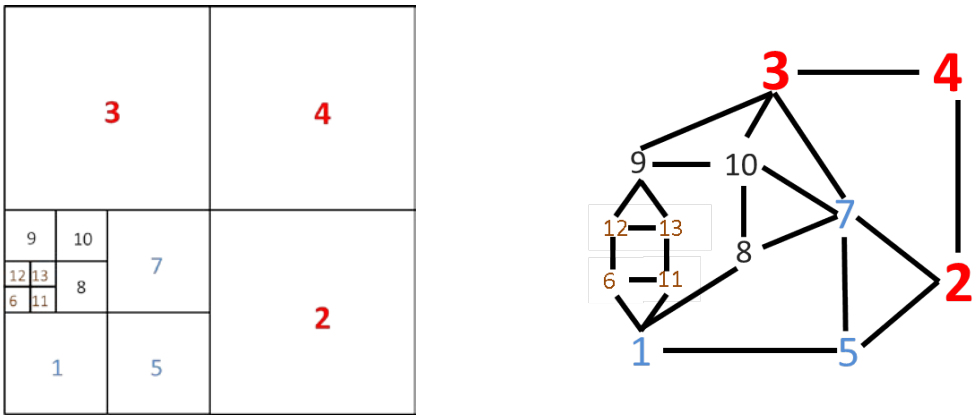


Figure 1-8. The cell-based unstructured AMR and its corresponding graph.

Table 1-2. A comparison of block-based, cell-based tree, and cell-based unstructured AMR.

	Block-based	Cell-based tree	Cell-based unstructured (present)
Domain decomposition	Block decomposition - an algorithm to distribute block units to processors	Coarse-grain based decomposition with space-filing curves	Heuristic domain decomposition algorithm
Data structure	Structured Cartesian grid	Tree- hierarchy	Unstructured grid
Leading memory cost	Data in distributed blocks	Data of leaves cells for all distributed trees	Global unstructured graph
Interpolation in a time step	Inter-level interpolation - required inter-processor communication	N/A	N/A
Caching rate	Good	Low	Fair, and constant for all levels
Refinement flexibility	Over-refined	Flexible	Flexible
Example	Guenny et al. (SAMARI) [47] MacNeice et al. (PARAMESH) [48]	Popinet et al. (Gerris) [35] Burstedde et al. (p4est) [54]	Lawlor et al. (ParFum) [58] Kuan et al. [41]

1.2.4 Droplet collision at high weber number regimes

Atomization is a process in various engineering applications that defragments liquid sheet/thread and ends up with finely dispersed droplets. This process usually starts with an ejection of liquid jets that drives the gas-liquid interface to be unstable and finally breaks up into smaller liquid fractions, ligaments, and micro-droplets. For decades, the design of atomizers and spray nozzles mostly relied on empirical estimations, or the knowledge of breaking mechanisms based on observations and physical reasoning. Detailed gas-liquid interface instabilities are hard to access, by either numerical or experimental approaches, since instabilities usually evolve rapidly in space from a meter to micrometer scale.

For this reason, to improve the understanding of gas-liquid instabilities, we apply the current numerical framework to model the classic binary droplet collision cases, which is a fundamental process in spray environments. In such a case, the primary dimensionless parameter is the Weber number, a ratio of inertia force to surface tension forces (Table 1-3). For a Weber number below 200, head-on, off-axis, and equal/unequal-size droplet collisions have been addressed numerically in extensive research [8, 60, 61]. Computation at high Weber number regimes (from 200 to thousands) is inevitably costly because gas-liquid interfaces appear multiple order of length scale after collision. Numerical simulation for such cases is absent now, and even experimental work is limited. Pan et al. reported head-on droplet collision at Weber number 200 to 5000 in 2009, and summarized the collision regimes as fingering, fingering and separation, breakup, and prompt splattering [62]. This study follows the operation conditions in the work of Pan et al. to demonstrate the computational capability of present parallel adaptive

Eulerian-Lagrangian method. Rayleigh-Plateau [63], Rayleigh-Taylor instabilities [64] and end-pinching at the end of a jet sheet (as known as Taylor-Culick rim [65]) are observed in current simulations. The interface evolution is summarized with a focus on surface instabilities, and the phenomena observed are compared with theoretical and experimental work. Physical insights based on numerical results are provided in the discussion.

Relevant non-dimensional parameters of multiphase flow problems used in current case studies are listed in Table 1-3. In the droplet collision problems, Reynolds, Weber, and Ohnesorge are key parameters used in determining the results. The Weber number is used broadly in characterizing thin film, droplets, and sprays. For a free surface with small a Weber number, gas-liquid boundaries tend to be stable in spherical shapes. Gas-liquid boundaries are unstable in case of a large Weber number. They easily form ligaments and break up [66].

The Ohnesorge number was originally introduced to describe the modes of droplet breakup due to aerodynamic force by Hinze [67]. The breakup regimes of droplets were determined based on the Weber and Ohnesorge numbers by the extensive experimental work of Hsiang and Faeth [68]. For a small Ohnesorge number ($Oh < 0.1$), observations show the breakup modes of droplets are almost independent of the Ohnesorge number [66]. The inverse square of the Ohnesorge number, the Laplace number, describes the ratio of two competing mechanisms: surface tension forces and damping effect of viscosity. For numerical computation of surface tension-dominant flows, the Laplace number is a scale of instability. Inaccurate computation of surface tension forces could cause serious spurious velocity and breaks down an entire flow

solver at a large Laplace number condition. Here, the Laplace number in numerical exercises of Chapter 5 is $O(10^5\text{-}10^6)$, which demonstrates accuracy and robustness of surface tension force computation in the current framework. The Eötvös number and Motron number are combinations of viscosity, density, surface tension of carrier fluids, and gravitational acceleration. They are two dimensionless parameters that determine the results of rising bubbles [69] and used in the validation cases of section 4.1.3. The Rayleigh number is a ratio of the buoyancy force induced by the thermal expansion of fluids to viscous force, and is adopted to define the condition of the natural convection flow in section 4.1.1.

Table 1-3. Relevant non-dimensional parameters used in the analysis of multi-fluid, moving boundary problems.

Dimensionless parameters	Definition
Reynolds number	$Re = \frac{\rho U_0 D_0}{\mu}$
Weber number	$We = \frac{\rho {U_0}^2 D_0}{\sigma}$
Ohnesorge number Viscous forces/surface tension force	$Oh = \frac{\mu}{(\rho D_0 \sigma)^{1/2}}$
Laplace number Surface tension force/momentum transport	$La = \frac{\rho D_0 \sigma}{\mu^2} = Oh^{-2}$
Eötvös number	$Eo = \frac{\Delta\rho g {D_0}^2}{\sigma}$
Motron number	$M = \frac{g\mu_l^4 \Delta\rho}{\rho_l^2 \sigma^2}$
Rayleigh number	$Ra = \frac{g\beta(T_s - T_\infty)l^3}{\nu\alpha}$
Density ratio (liquid/gas)	ρ_l/ρ_g

1.3 Objectives

Three objectives of the present study are summarized.

a. **A parallel Eulerian-Lagrangian method**: This dissertation presents a parallel implementation of the Eulerian-Lagrangian method for multi-scale moving boundary computations. The study incorporates the Eulerian-Lagrangian interface tracking method, Lagrangian interface modification (smoothing/coarsening/refining), reconstruction algorithm, and the cell-base unstructured adaptive mesh refinement technique in a distributed-memory computation paradigm.

b. **Cell-based unstructured adaptive mesh refinement**: This parallel cell-based unstructured adaptive mesh refinement technique brings substantial

computational power to the dynamic computation of moving boundary problems. Current work describes the parallel remeshing, data redistribution, and domain decomposition. The performance of the Eulerian computation, Lagrangian interface tracking method, and cell-based unstructured AMR is presented separately.

c. **High Weber number droplet collision simulation**: The parallel adaptive Eulerian-Lagrangian method developed here is applied to simulations of binary droplet collision at a high-density ratio $O(10^3)$, the Weber number $O(10^2\text{-}10^3)$, and Laplace number $O(10^5)$. The effectiveness of the current parallel adaptive framework is shown for multi-scale moving boundary computations.

1.4 Outline

The present study is outlined as follows.

Chapter 2 addresses the numerical framework of the three-dimensional Eulerian-Lagrangian interface tracking method. Chapter 3 describes the parallelisms used for the Eulerian and Lagrangian domains. Challenges of achieving high performance are addressed. Detail algorithms of domain decomposition, communication strategies, parallel Lagrangian surface modification, and reconstruction algorithms are discussed. Implementations of the cell-based unstructured adaptive mesh refinement including parallel remeshing, load balancing, data redistribution are illustrated comprehensively.

In chapter 4, code validations are presented for the parallel implementation of the sharp interface method for solid boundaries, and continuous interface method of moving fluid boundaries. The performance studies cover the scalability of the field equation solver, adaptive mesh refinement, and overall performance of the Eulerian-Lagrangian method.

In chapter 5, droplet collisions at a higher Weber number regime are demonstrated. Numerical results are compared with experimental work, and a grid dependency study is presented. Dynamics of surface instabilities on gas-liquid boundaries are addressed, and the mechanisms of droplet defragmentation are discussed.

Chapter 2.

Governing equations and numerical methods

2.1 Introduction

The Eulerian-Lagrangian method resolves velocity field on the Eulerian grid, and treats the interfaces separating different phases by Lagrangian surface meshes. Figure 2-1 shows an illustration of the present method. A basic geometrical unit of interfaces is a line-segment in two-dimensional and a triangular element in three-dimensional domains. Both fluid and solid interfaces are in the Lagrangian frame. The no-slip boundary condition of solid interface is enforced by the ghost cell method as shown in Figure 2-1(c). Velocity, pressure, and temperature reconstruction in a ghost cell is extrapolated from the fluid side along the local norm of the solid boundary (Figure 2-3).

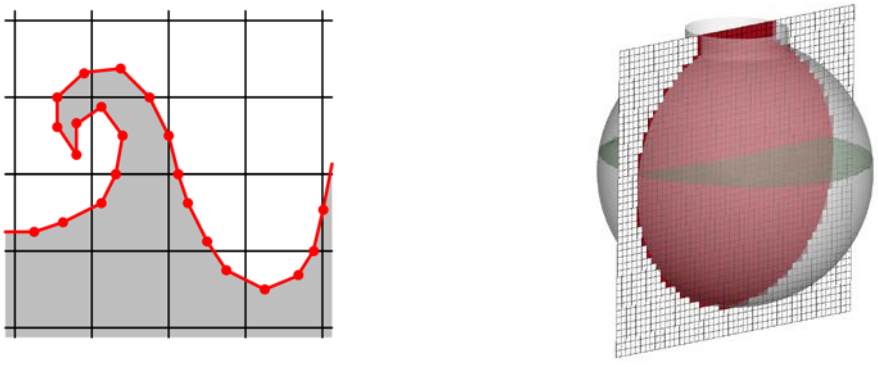


Figure 2-1. The Eulerian-Lagrangian method. (a) Two-dimensional Lagrangian markers (red) on the stationary Cartesian grid. (b) Three-dimensional illustration: a liquid interface in a spherical container. Field equations are solved on the Eulerian Cartesian grid (red). Fluid interfaces are represented by moving the Lagrangian mesh (green), and solid boundary (gray) is embedded with the ghost cell

method for boundary condition enforcement.

The following sections will focus on the fluid interface since its dynamic behavior induces issues on parallelism. More detailed numerical methods of the Eulerian-Lagrangian method can be found in literature [1, 8, 41].

2.2 Governing equations

On the Eulerian domain, the governing equations are mass, momentum and energy conservation equations for incompressible Newtonian fluids. We account interfacial dynamics as source terms at the right hand side of the momentum and energy equation. The momentum forcing term, $\mathbf{F}_f$ accounts for the effect of the surface tension of fluid interfaces. The forcing function $\mathbf{F}_s$ represents no-slip condition on the solid interfaces. The energy source term, Q_f is latent heat effects around fluid interface. Here, $\mathbf{V}$ is the velocity vector, and ρ , μ and p are the density, viscosity, and pressure, respectively. Non-dimensional zed parameters are Reynolds (Re), Froude (Fr), Weber (We), and Prandtl (Pr) number.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0 \tag{3}$$

$$\frac{\partial (\rho \mathbf{V})}{\partial t} + \nabla \cdot (\rho \mathbf{V}\mathbf{V}) = -\nabla p + \frac{1}{Re} \nabla \cdot \mu (\nabla \mathbf{V} + \nabla^{\mathrm{T}} \mathbf{V}) + \frac{1}{Fr} \rho \mathbf{g} + \frac{1}{We} \mathbf{F}_f + \mathbf{F}_s \tag{4}$$

$$\frac{\partial (\rho c T)}{\partial t} + \nabla \cdot (\rho c T \mathbf{V}) = \frac{1}{RePr} \nabla \cdot (K \nabla T) + Q_f \tag{5}$$

We solve Eqs. (3)-(5) by a finite-volume projection method on a staggered grid. The intermediate velocity field is computed first, and then, projected onto a divergence-

free space to satisfy the mass conservation equation. The convection term is discretized by the third order ENO scheme in space and second order Runge-Kutta integration in time. The viscous term uses the central difference discretization in space and Crank-Nicholson integration in time. With the phase change model, the divergence-free condition is modified by accounting the amount of mass transfer across the interface.

Across fluid interface, the pressure and viscous stresses show discontinuities due to surface tension force. The jump condition of the flow properties in the normal direction ($\mathbf{n}$) can be computed from the surface tension (σ) and the curvature (κ) in Eq. (6). On the solid phase boundary, the no-slip condition is implemented with the prescribed velocity of the moving solid geometries.

$$(p_2 - p_1) - \mathbf{n} \cdot (\tau_2 - \tau_1) \cdot \mathbf{n} = \sigma\kappa \tag{6}$$

2.3 Indicator function

At the vicinity of an interface, we calculate an indicator function *I*, a discrete form of the Heaviside step function, obtained by integrating the 1-D form of the approximate Dirac delta function (δ_h) from liquid to gas phase as shown in Eq. (7).

$$\mathrm{I}(\mathbf{x}) = \mathrm{H}\left(r = \mathbf{n} \cdot \left(\mathbf{x} - \mathbf{X}\right)\right) = \int_{-\infty}^{r} \delta_h(h)dh \tag{7}$$

The approximate Dirac delta function, originally proposed by Peskin [29], is calculated over finite thickness of four-cell width as in Eq. (8).

$$\delta_k = \begin{cases} \frac{1}{8}(5 - 2|r| - \sqrt{-7 + 12|r| - 4|r|^2}) & 1 \le |\mathrm{r}| \le 2 \\ \frac{1}{8}(3 - 2|r| + \sqrt{1 + 4|r| - 4|r|^2}) & 0 \le |\mathrm{r}| \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The $|r|$ is the distance from an Eulerian cell center to the nearest Lagrangian element or vertex. Figure 2-2 is a 1-D illustration of the discrete Dirac delta function and indicator function across a fluid boundary at x = 0.

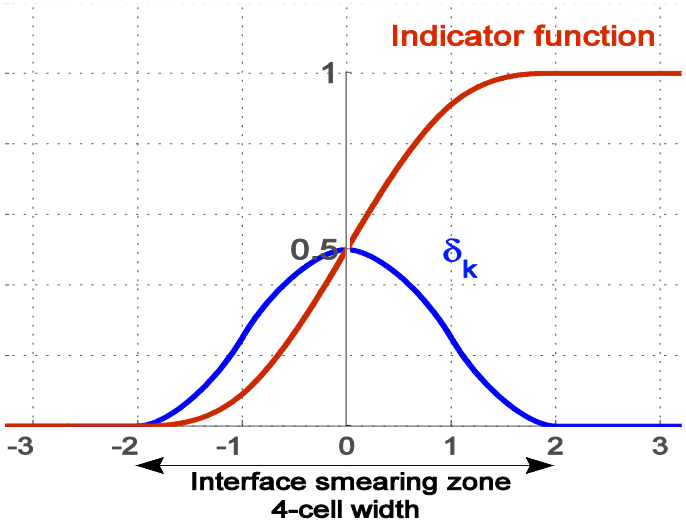


Figure 2-2. Indicator function is integrated from the discrete Dirac delta function across the interface. Reproduced from Sim [70].

The 3D extension of the Dirac delta function can be obtained by the multiplication of the Dirac delta function of each coordinate by Eq. (9) [29].

$$\delta_h = \frac{1}{\Delta x \Delta y \Delta z} \delta(r_x)\delta(r_y)\delta(r_z) \tag{9}$$

The indicator function at the interface is 0.5, and varies from 0 to 1 smoothly across the interface. This approach is known as being more general than the Poisson equation solver method since it requires only distance information from the interface, and thus gives accurate values even at the boundaries [42]. This indicator function maps discontinuous material properties into a continuous form so that we can use a single set of equations to describe all fluid phases in the domain with smoothed-out material

properties across the interface.

The smoothed fluid properties are computed by the indicator function in Eq. (10). Here, φ can be material properties such as density ρ, kinematic viscosity ρ/μ, heat capacity ρC, and thermal diffusivity $k/\rho C$.

$$\varphi = \varphi_2 + (\varphi_1 - \varphi_2)\mathrm{I} \tag{10}$$

2.4 Interface tracking

Lagrangian interfaces are a set of triangular elements formed by Lagrangian markers with connectivity. The location of a marker, denoted by $\mathbf{X}$ in a Lagrangian frame is updated based on the marker velocity $\mathbf{V}(\mathbf{X})$, given by the equation of motion as

$$\frac{\partial \mathbf{X}}{\partial t} = \mathbf{V}(\mathbf{X}) \tag{11}$$

The forward Euler method is used to update marker location. The velocity of a marker is a projection from the resolved velocity field on the underlying Eulerian grid to the Lagrangian frame, as shown in Eq. (12). The approximate Dirac delta function, δ_h, is employed for transforming the Eulerian velocity field, $\mathbf{v(x)}$, to Lagrangian marker velocities $\mathbf{V}(\mathbf{X})$. The interface velocity is a function of the Eulerian velocity field and mass transfer rate $\dot{m}$ on the Lagrangian domain in case of phase change.

$$\mathbf{V}(\mathbf{X}) = \int_V \mathbf{v}(\mathbf{x})\delta_h(\mathbf{x} - \mathbf{X})dV - \frac{\dot{m}}{\rho_f} \tag{12}$$

2.5 Contact line force

When a fluid-fluid interface intersects a solid surface, the treatment of the tri-

junction locations, called contact lines, is required to account for the presence and interactions of the gas, liquid, and solid phases. Modeling contact line force and imposing no-slip condition for velocity in Navier-Stokes equations lead to a non-integrable singularity in stress. Here, the contact line force is imposed with a local slip condition to overcome this singularity issue. The contact line force, which is computed from the present contact angle and the given equilibrium contact angle, is implemented as surface tension forces and makes the interface approach the static contact angle asymptotically. In the local slip model, a perfect slip is applied exactly on the contact line by assuming infinite slip length, and a partial slip is applied within two-cell distances with the help of the approximated Dirac delta function as continuous fluid interface is diffused. This local slip condition is implemented with the previously-mentioned sharp solid interface method by interpolating velocities of solid points (SP) on the solid surface (including contact line velocity) from the known fluid velocity fields and weighing them with the approximate Dirac delta function. The details can be found in our previous work [1].

2.6 Interfacial dynamics modeling

On the Eulerian grid, the discontinuities of surface tension and heat transfer from a Lagrangian marker are treated diffusively in finite-width variations. At the vicinities of the Lagrangian interfaces, we use the approximate Dirac delta function to convert Lagrangian quantity $F(\mathbf{X})$ to Eulerian quantity $f(\mathbf{x})$, as shown in Eq. (13).

$$f(\mathbf{x}) = \int_{\Gamma(t)} F(\mathbf{X})\delta_h(\mathbf{x}-\mathbf{X})\,d\Gamma \tag{13}$$

For surface tension forces exerting on a Lagrangian element, the Lagrangian data is $\sigma\kappa$, where κ is the local curvature of the Lagrangian element. By a line integral form

shown in eq. (14) with normal and tangent vectors instead of a direct curvature computation, the surface force on a discrete Lagrangian element is computed. Curvature computation using interpolation-based methods is numerically sensitive and often requires some form of data smoothing, and does not enforce surface tension conservation on a closed surface.

$$\delta f = \int_{\delta A} \sigma(\mathbf{n}\times\nabla)\times\mathbf{n}d\mathrm{A} = \int_{S} \sigma(\mathbf{t}\times\mathbf{n})dS \tag{14}$$

When phase change is considered, the latent heat is computed by Eq. (15), where $\dot{m}_f$ is mass transfer per unit area across the interface due to phase change and L is the latent heat at the saturation temperature.

$$Q_f = \int_{\Gamma(t)} \dot{m}_f L \delta_h(\mathbf{x}-\mathbf{X})d\Gamma \tag{15}$$

The mass transfer rate across the interface is computed from heat transfer relations in Eq. (18) based on the Stefan condition using the temperature gradient with discontinuous material properties. Here, L is the latent heat, k is thermal conductivity, and q_1 and q_2 are heat flux vectors transporting across interfaces. The interface temperature is assumed to be equal to the saturation temperature which is an adequate assumption in macroscopic problems.

$$\dot{m}_f L = (\mathbf{q_1}-\mathbf{q_2})\cdot\mathbf{n} = -k_1 \left.\frac{\partial T}{\partial n}\right|_1 + k_2 \left.\frac{\partial T}{\partial n}\right|_2 \tag{16}$$

For more detail implementation and formulations of phase change models, please refer to the work of Sim and Kuan [71]. Here, we address phase change modeling briefly for comprehensively documenting capabilities of this multiphase flow code, and phase

change is not in the scope of the current dissertation.

The accuracy of the current implementation was shown in the work of Singh [8] for a static bubble problem. The non-dimensional spurious velocity (Capillary number) due to numerical error on interface curvature computation is in the order of 10^{-4} and insensitive to Laplace number up to 10^{5}, while VOF and LS methods usually produce spurious velocity in order of 10^{-2} and cannot sustain high Laplace number computation without sophisticated interface reconstruction algorithms.

2.7 The ghost cell method for solid boundary condition

We use the sharp interface method to model the solid boundaries and enforce the boundary conditions by reconstructing solution fields with the given boundary condition on the solid phase. Velocity is reconstructed in ghost cells, which are defined as solid cells adjacent to any fluid cell. In this sharp interface method, the accuracy of a solid boundary condition depends on the reconstruction scheme of solution fields at ghost cells, and thus, the accuracy of interpolations at imaginary points. An imaginary point IP is built by extending the normal of the local solid boundary from the center of a ghost cell toward the fluid phase with a constant interval, while the solid point (SP) is the intersection point of the normal vector and solid interface. An appropriate reference length is the minimum cell width as the interval for SP to IP. Figure 2-3 is an illustration of reconstruction stencil.

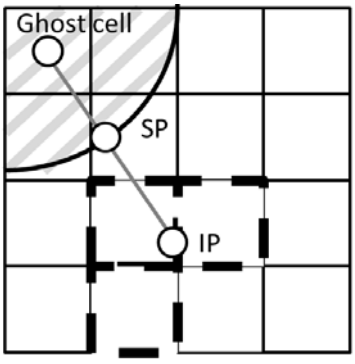


Figure 2-3. Schematics of reconstruction stencil of the two-dimensional linear reconstruction scheme with imaginary point (IP) and solid point (SP).

A linear interpolation is applied to determine the values at the imaginary point as shown in Eq. (17) which requires four fluid cells in three-dimensional domains.

$$\phi_{IP} = c_0 + c_1 x + c_2 y + c_3 z \tag{17}$$

A quadratic interpolation as Eq. (18) can be used such that seven fluid cells are needed for the interpolation in a three-dimensional domain. Other types of quadratic formulation can be found in [27].

$$\phi_{IP} = c_0 + c_1 x + c_2 y + c_3 z + c_4 xy + c_5 yz + c_6 xz \tag{18}$$

For a second-order accurate finite volume flow solvers, quadratic interpolation of flow variables at the wall may retain second-order accuracy of the scheme. A comparison of interpolation schemes for ghost cell immersed boundary methods by Tseng and Ferziger showed improvement on L_∞ error norm by using a quadratic interpolation, but overall performance of the solver is not affected greatly by using a linear interpolation scheme [72], and a similar observation was reported by Kao et al. [73]. Iaccarino and Verzicco concluded that a linear reconstruction is applicable for laminar flows or for high

Reynolds number flows when the image point is in the viscous sub-layer [74]. In our case studies, whenever adaptive mesh refinement is applied around the solid interfaces, a grid size is adopted that ensures the boundary layer is well resolved such that image points always stay in the laminar regions. A concern of quadratic interpolations is that more than seven points in the fluid phases are required to obtain the reconstruction coefficient. This requirement is sometimes difficult for ghost cells at large-curvature solid boundaries such that the stencil may be in a skew shape, which results in a stiff linear system of equation. As a result, using linear interpolation scheme for the reconstruction of variables on image point is a reliable and robust approach for complex geometry cases.

Another issue mostly discussed for modeling a sharp interface is mass/energy conservation on the boundary cells. A solid boundary that does not conform with the computational grid causes confliction between the no-slip condition on the surface and the mass conservation of boundary cells. Many researchers implemented a finite difference method instead of a finite volume method in order to detour such an issue [75]. A mass/energy source/sink approach was proposed to compensate the non-divergence free condition of boundary cells, but it just guarantees the convergence of the pressure Poisson solver, and the mass is still not conserved with computed no-slip conditions [76]. Local correction method, which is our previous approach, is also a kind of mass source/sink method, and has same issues on the accuracy [42]. When the Reynolds number is small and grid is fine, the velocity fields have small values around a solid surface, and the error can be negligible. However, high Reynolds or Rayleigh number flow exacerbates the accuracy.

Present approach imposes Neumann pressure boundary condition implicitly in

Poisson equation. When solving the pressure Poisson equation, the pressure of ghost cells is used to enforce zero normal pressure gradients at the immersed solid wall. The zero normal pressure gradients is equivalent to a non-permeable wall, and thus mass conservation should be satisfied for fluid cells if ghost cell pressure is updated consistently. For each ghost cell, a Neumann condition implies ghost cell pressure is equal to pressure at its respective imaginary point. Hence, the ghost cell pressure of an imaginary point could be determined by a weighting combination of multiple fluid cells adjunct to its respective image point. In a three-dimensional domain, the linear relationship of pressure at an imaginary point and its surrounding fluid cells can be represented by a coordinate vector [1 x y z] and the interpolation coefficient $[c_0\ c_1\ c_2\ c_3]^t$.

$$P_I = \begin{bmatrix} 1 & x_I & y_I & z_I \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}, \tag{19}$$

where

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{X} & \mathbf{Y} & \mathbf{Z} \end{bmatrix}^{-1} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \tag{20}$$

In the iterative solving procedure of pressure Poisson equation, the pressure P_1 to P_4 are objects at the linear system equation of discretized Poisson equation. With the inverse of the coordinate matrix [**1 X Y Z**], a ghost cell pressure is associated linearly to its respective fluid cells with zero wall normal derivative enforced.

$$P_G = P_I = [1 \quad x_I \quad y_I \quad z_I]\left[\mathbf{1} \quad \mathbf{X} \quad \mathbf{Y} \quad \mathbf{Z}\right]^{-1}\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = \sum_{k=1}^{4} a_k P_{F,k} \tag{21}$$

Then the ghost cells are embedded in the linear system of discretized pressure Poisson equation. Since the coefficient matrix of the linear system with embedded Neumann BC is not symmetric and positive, the entire linear system is solved by a bi-stabilized conjugate gradient solver or GMRES solver of PETSc [77] with Jacobi, BoomerAMG or ParaSAILS preconditioner of *hypre* [78]. Sim [70] showed that the current field equation solver with the ghost cell reconstruction for solid boundary conditions has grid convergence between first and second order accuracy for a natural convection problem (Figure 4-2) with an adaptive grid.

2.8 Summary of the Eulerian-Lagrangian method

This numerical framework utilizes the stationary (Eulerian) frame to resolve the flow field, and the marker-based triangulated moving (Lagrangian) surface meshes to treat the phase boundary interfaces. Both continuous and sharp interface methods are implemented in a unified framework. The source terms in the governing equations computed at the vicinity of the interface are transferred between Eulerian and Lagrangian frame through the discrete Dirac delta function. For moving contact line treatment, local slip condition is applied around the contact line. The large-deformable fluid boundaries are modeled using a continuous interface method, and the surface tension between fluid interfaces is smeared within finite distance. The solid boundaries are treated by a sharp interface method along with the ghost cell method by reconstructing the solution on the ghost cell based on the known solid boundary condition. The pressure boundary

condition of a sharp interface method at solid boundaries is embedded in the linear system of discretized pressure Poisson equation.

Chapter 3.

Parallelism

Figure 3-1 describes the tasks of the Eulerian-Lagrangian interface tracking method in a single time step. Tasks are placed in the Eulerian or Lagrangian frame to show the space of computation with arrows to indicate data dependency between tasks. In a single time step, we have four times of cross-domain communication through the Dirac delta function. Strong data dependency between the Eulerian and Lagrangian domains implies data locality is a primary factor determining the parallel performance. For a serial computation, solving field equations on the Eulerian domain usually costs more than 60% of wall-clock time. Therefore, the load balance on the Eulerian computation is a priority, but appropriate decompositions of the Lagrangian or Eulerian-Lagrangian computation are also required for overall efficiency. This is non-trivial to achieve because the size and location of the Lagrangian mesh vary in time and space.

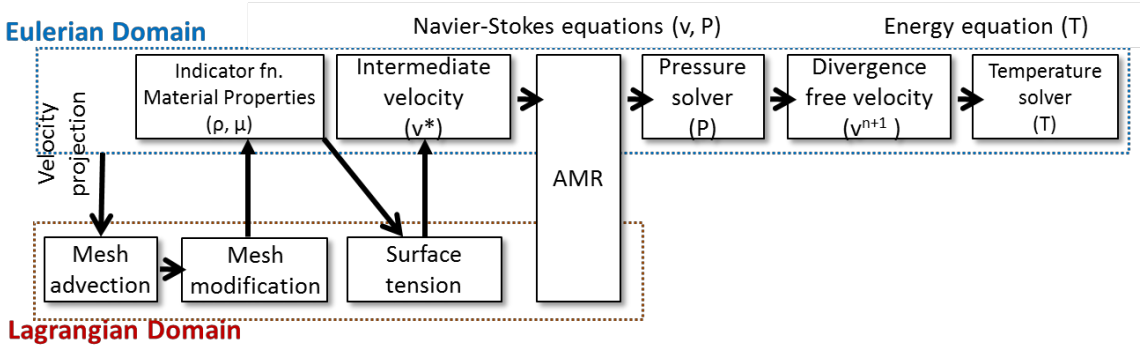


Figure 3-1. Procedures of the Eulerian-Lagrangian interface tracking method.

3.1 Eulerian domain decomposition including sharp interface method

A partitioning methodology of Eulerian domain needs to consider the following issues; load balance, communication cost, and data dependency arising from ghost cell methodology. Among many approaches for the spatial decomposition of a graph, the most popular two methods are space-filling curve methods and heuristic methods such as partitioning libraries ParMETIS[79]. A space-filling curve traverses an N-dimensional graph and maps it bijectively to a one-dimensional array. The ordered array is then broken into p parts, where p is the total number of distributed computational elements. Consequently, the load balance is satisfied, but the total edges cut, that is, the size of partition boundary requiring information exchange, is not an objective of a space-filling curve method. On the other hand, graph partitioning packages such as ParMETIS using k-way multilevel heuristic approach as partitioning algorithm usually provide satisfactory load balance and fewer edges cut compared with the space-filling curves methods [80]. Moreover, it provides flexible weighting function to manipulate a preferred partition scenario.

The graph-partitioning package ParMETIS is used to handle the Eulerian domain decomposition in the present study. Figure 3-2(a) is an illustration of an Eulerian domain decomposition. This experiment models the liquid sloshing of the cryogenic fuel in a liquid hydrogen tank of a Saturn V/S-IVB rocket. We use dynamic material tags to distinguish gas, liquid, and solid phase as shown in Figure 3-2 (b). Because the ghost cells in the solid boundary are not evenly distributed among partitions, the workload of ghost cell boundary condition method is not balanced. However, since the computational time of this operation contributes very little to the total runtime, we do not observe overhead due to load imbalance in the ghost cell method. Between adjacent partitions, we

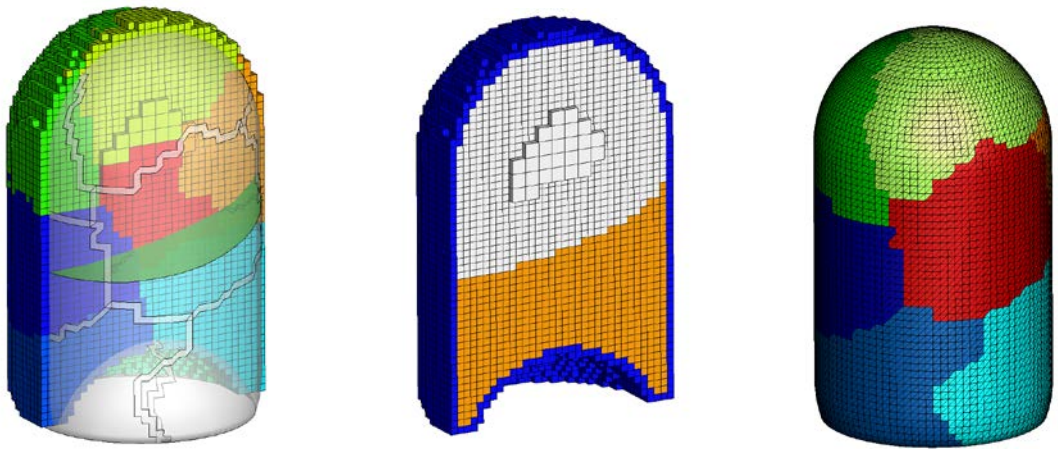
build overlapping layers. The overlapping zone serves as a buffer region for the synchronization of information on the boundaries of two adjacent partitions. Suppose that an Eulerian domain Ω is already decomposed into n non-overlapping partitions Ω_p, i.e.,

$$\Omega = \bigcup_{p=1}^{n} \Omega_p \tag{22}.$$

Here p is the processor ID and n is number of processors. We define a sub-domain, Ωs_p,

$$\Omega s_p = \Omega_p \bigcup (\bigcup_{i=1}^{adj} I_{p,i}) \tag{23}.$$

It is a union of an individual partition Ω_p and regions of a finite width away from the partition Ω_p overlapping with its adjacent partitions.



(a) (b) (c)

Figure 3-2. The Eulerian domain decomposition of a liquid fuel tank with a free boundary of the gas-liquid interface. (a) Partitions are represented in different colors. (b) The material tags of gas phase (white), liquid phase (yellow), and solid phase (blue). (c) Decomposition of the solid boundary according to the Eulerian partitions.

The term, $I_{p,i}$ represents the overlap of the sub-domain Ωs_p with its neighbor partition, Ω_i. A sub-domain Ωs_p has its own local grid connectivity. The overlapping

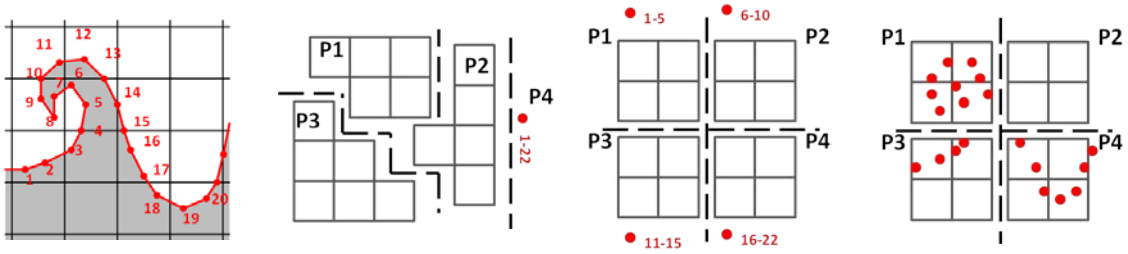
zones, $I_{p,i}$ purely serve as ghost cells for the storage of data from processor *i* through communication routines. They are also the buffer zones to perceive influences from moving Lagrangian markers, which is addressed in the next section.

3.2 Lagrangian domain decomposition

The information on the Eulerian and the Lagrangian domain is highly coupled in a single time step. From the step of Lagrangian mesh advection to intermediate velocity field computation on the Eulerian grid, procedures on both domains have prerequisite procedures on their domain counterparts, as shown in Figure 3-1. For the parallelisms of Lagrangian interface, three approaches can be considered: task parallelisms, atomic decomposition, and spatial decomposition. An example in Figure 3-3 illustrates how four processors could decompose a problem with Lagrangian markers denoted in circle dots superposing on an Eulerian Cartesian grid.

Task parallelisms reserve some specific processors to exclusively deal with Lagrangian computations and use other processors for the calculation of Eulerian data. This approach takes advantage of the concurrency between Eulerian and Lagrangian tasks to accelerate computation. However, there is very little concurrency between Eulerian and Lagrangian procedures. Because the Lagrangian and Eulerian procedures need information on their domain counterparts beforehand, it is difficult to overlap the Lagrangian and Eulerian computation in an efficient manner, which excludes the effectiveness of task parallelisms. We consider data parallelism to balance the Lagrangian computation. We may distribute Lagrangian markers evenly over all processors regardless of their location in the Eulerian domain as the atomic decomposition approach of Figure 3-3(c). This straightforward approach achieves the load balance of Lagrangian

computation automatically and may perform well in small problems. For larger grid size problems, there is an increasing amount of data needed to transfer across processors because the information required for computation is unavailable in the local memory of a processor. The communication overhead will be the dominant bottleneck and the worst case is deadlock when data to be sent is much larger than the system buffer. Furthermore, the communication pattern will be very dynamic and complicated. Therefore, we have to consider the data locality to minimize the communication overhead.



(a) (b) (c) (d)

Figure 3-3. Strategies of decomposition of Lagrangian interface (a) A 2D Lagrangian mesh on Eulerian domain (b) task parallelism (c) atomic decomposition (d) spatial decomposition

A spatial decomposition assigns the Lagrangian markers to a processor based on the spatial inhabitation of markers. In Figure 3-3(d), the Lagrangian markers encompassed by an Eulerian sub-domain owned by a particular processor should be assigned to this processor. This approach takes the advantage of data locality that prerequisite information for computation is available in the local memory of each processor such that the frequency and size of communication can be minimized. The load balance of Lagrangian computation may not be satisfied but reasonable scalability is expected in general cases since the cost of pure calculation of Lagrangian data is much smaller than the cost of the flow solver.

We define a Lagrangian partition of a processor by including all Lagrangian markers inhabiting in its Eulerian partition. Markers in a Lagrangian partition are defined as “active” because the host processor will manage the advection and computation of these active markers. Note that Eq. (13) transfers Lagrangian quantity to the Eulerian grid through the approximate Delta function. In other words, a Lagrangian element disperses its impact on the Eulerian grid in a finite-diameter spherical space. The sphere centers at the Lagrangian marker with the cut-off length of the approximate Dirac delta function as radius. As a result, to ensure the Eulerian partition receives correct influences from the Lagrangian interface, a processor includes extra Lagrangian markers out of its Eulerian domain, which may have influence on any cells of the Eulerian partition. They are defined as “passive” because neighbor processors host their advection.

Locating Lagrangian markers on a local Eulerian partition can be expensive. We employ the following procedures to accomplish this work. After defining the Eulerian partition at the procedure of “Load Balance” in the AMR method, a processor defines the Eulerian overlapping zone in a range that can accommodate possible passive markers. Defining the overlapping zone requires searching reachable vertexes (cells) in a finite distance from a root vertex on the global Eulerian graph. The root vertex is a cell on partition boundaries, and the searching direction points to neighbor partitions. We explore cells in neighbor partitions by the breadth-first search (BFS) technique to determine overlapping zones. The BFS function defines the cells of a neighbor partition that are in a designated distance from a starting point. The distance of searching is the cut-off length of the approximate Dirac delta function such that any marker that enters the overlapping zones will exert influence on certain cells in a partition. Figure 3-4 illustrates

this algorithm.

1 Given Eulerian partition Ω_p and a global Eulerian graph *G(V,E)*, where *V* is the vertex and *E* is the edge.

2 $l_{cut\text{-}off}$: the cut-off length of the Dirac delta function

3 u[ ] : a returned list of the breadth-first function

4 DO neighbor partition ID = i

5 DO boundary cell next to *i* partition

6 CALL breadth-first search function BSF($l_{cut\text{-}off}$, *G(V,E)*, u[ ])

7 Add u to overlapping zone $I_{p,i}$, $I_{p,i} \leftarrow$ u[ ]

8 END DO

9 Remove duplication cells in $I_{p,i}$

10 END DO neighbor partition

11 define the Eulerian sub-domain of processor *p*, $\Omega s_p = \Omega_p \cup (\bigcup_{i=1}^{adj} I_{p,i})$

Figure 3-4. Eulerian sub-domain algorithm

With the overlapping zones and the Eulerian partition, each processor can safely determine passive and active markers at every time step. The set of active markers form a Lagrangian partition, $L_p(t^n)$, and the union of active and passive markers comprise a Lagrangian sub-domain $Ls_p(t^n)$ at time step t^n. Figure 3-5 shows the Lagrangian sub-domain algorithm. This approach has several merits. First, it reduces required Eulerian-Lagrangian messages because of data locality. Second, we store only the mesh connectivity of the global Lagrangian interface, and all other Lagrangian data are distributed into processors having interface dwelling in their Eulerian partitions. Though

the Lagrangian computation is not perfectly balanced, we at least maintain scalability for large problems. Figure 3-6 shows an example of Lagrangian sub-domain with respect to the Eulerian partition.

---

1 Project velocity from Eulerian partition Ω_p to Lagrangian marker M_p^i, $M_p^i \in L_p(t^n)$

2 advect markers for all M_p^i, $M_p^i \in L_p(t^n)$

3 conduct surface modification for all M_p^i, $M_p^i \in L_p(t^n)$

4 smoothing/refining/coarsening

5 locality_tag [ ] : locality tag of markers

6 DO active Lagrangian marker M_p^i

7 searching the location of M_p^i

8 IF(M_p^i is in partition E_k) locality_tag [i] ← k

9 END DO

10 communicate locality_tag with neighborhood partitions for $M_p^i \in \partial L_p(t^n)$

11 re-define active and passive markers on $M_p^i \in \partial Ls_p(t^n)$

12 re-define Lagrangian partition $L_p(t^{n+1})$ and subdomain $Ls_p(t^{n+1})$

Figure 3-5. Lagrangian sub-domain algorithm.

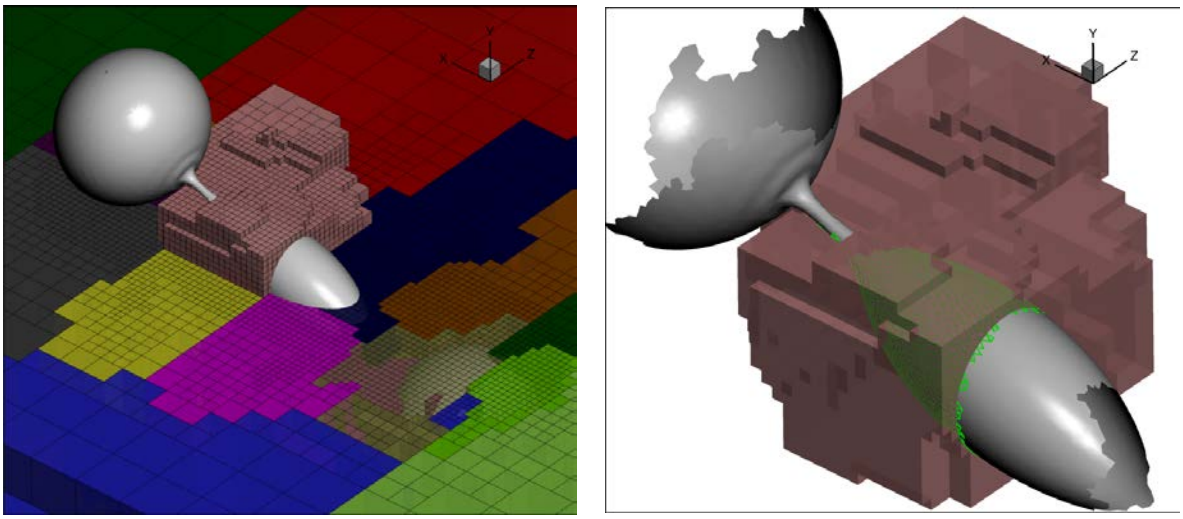


Figure 3-6. Illustration of Lagrangian domain decomposition. (a) An elongated droplet stays in several Eulerian partitions. (b) Markers dwelling in an Eulerian partition are flagged as active (green mesh). Passive markers (shaded in grey) are markers that possibly scatter source terms on the grid cells in the current Eulerian partition. Surface tension forces and other source terms are computed both on active and passive markers by the host processor. The active and passive markers constitute a Lagrangian sub-domain.

3.3 Lagrangian interface modification

3.3.1 Smoothing, refining, and coarsening

Lagrangian markers move with carrier fluids. However, with the direct propagation of interfaces according to the equation of motion, markers on the Lagrangian interfaces may form unfavorable distribution. To maintain the consistent accuracy of the interface-related computations, the spacing between the markers is rearranged by adding or removing markers whenever markers are too close or too distant from others. We employ three techniques—smoothing, refining, and coarsening—to organize marker distribution. These operations are based on the volume-conserved surface mesh restruction of Singh and Shyy [1]. The decision to activate a mesh modification is evaluated by comparing local space between markers with the resolution of the

underlying Eulerian grid. For a 2-D Eulerian cell with inhabiting Lagrangian interfaces, it should contain at least one and at most three markers within its volume in order to maintain continuous representation for transferred quantities like surface tension force. This constraint is used to determine criterion distance between two adjacent markers *i* and *j* with respect to Eulerian grid spacing, Δx , as shown in Eq. (24).

$$\begin{aligned} &\text{if} \quad \left|\mathbf{X}_i - \mathbf{X}_j \le \frac{\Delta x}{3}\right| \quad \text{, coarsening} \\ &\text{if} \quad \left|\mathbf{X}_i - \mathbf{X}_j \ge \Delta x\right| \quad \text{, refining} \end{aligned} \tag{24}$$

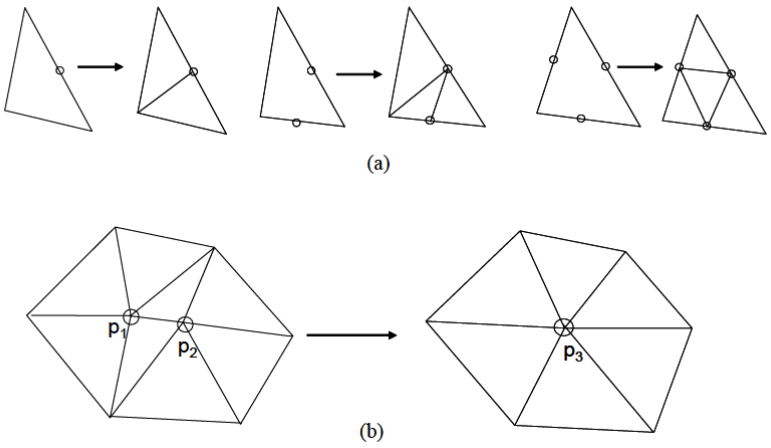


Figure 3-7. Refine and coarsen operation on Lagrangian mesh. (a) Refine: long edges are divided by adding a marker in the middle of edges; new elements are constructed in the original element. (b) Coarsen: two very close makers p_1 and p_2 collapse into p_3. Adopted from Singh [81].

Figure 3-7 is an illustration of refine and coarsen based on the method of Singh [81].

Parallelizing the smoothing operation is relatively straightforward because the connectivity of a Lagrangian mesh is not changed. A processor applies smoothing on the markers of its Lagrangian partition and then exchanges new marker coordinates with its neighbor partitions. In contrast, the refining and coarsening operations change the connectivity of a Lagrangian mesh. Without constrains, we have to remesh the interior of

a Lagrangian partition and partition boundaries separately with communications to coordinate modification on partition boundaries. Here, we limit the coarsening and refining operation to the interior of Lagrangian partition such that communication is avoided during these two operations. After local refining and coarsening operations, the global connectivity of the Lagrangian mesh is updated by a collective communication.

3.3.2 Reconstruction for topology change

Unlike VOF and level-set methods, the Eulerian-Lagrangian interface tracking method needs special treatment for topology change. The topology reconstruction algorithm has two steps: topology change detection and surface mesh reconstruction. We use a norm-based probe method to identify the necessity of interface merge and breakup. Along the interface norm of a marker, we examine the material tags at probes on each side of the interface. If the material tags at two probes refer to the same material, topology reconstruction is necessary. The detection process is handled in parallel. Each processor executes topology change detection on its Lagrangian partition, and flags Lagrangian bodies that need topology reconstruction. Once the reconstruction flags of all Lagrangian bodies are determined, we temporarily reserve the information of the Lagrangian bodies that are free from merge and breakup at this moment, and then execute surface mesh reconstruction algorithm on the flagged bodies.

The level-contour-based interface reconstruction technique is implemented to create new surface mesh of reconstructed bodies. Indicator function is recomputed for flagged bodies by integrating the Dirac delta function in parallel, and then new interface is created on indicator function $I = 0.5$ contour in serial. The overhead of creating new Lagrangian mesh is not a primary concern because usually the topology reconstruction is

invoked at tens of time steps if required. After new Lagrangian meshes are generated and the body tags are defined, reserved Lagrangian bodies are combined with new Lagrangian bodies. Finally, the Lagrangian sub-domain algorithm is invoked to re-determine the Lagrangian partitions.

3.4 Communication strategy

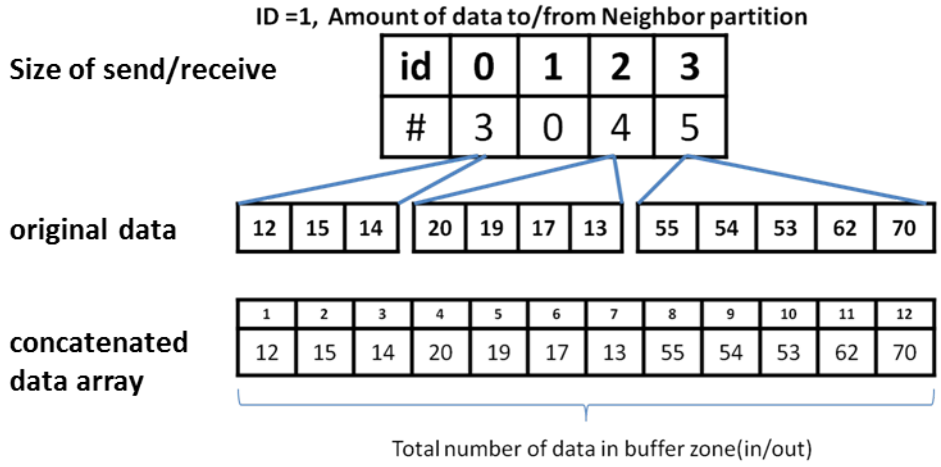


Figure 3-8. An example of a concatenated array of data in overlapping zones of processor ID 1 for a four-processor computation.

Data in overlapping zones of a sub-domain are frequently exchanged with neighbor sub-domains when solving field equations and propagating the Lagrangian markers. For a point-to-point communication, i.e., a sub-domain synchronizing data in overlapping zones with neighbor partitions, the non-blocking send and receive are used to avoid blocking delay. The send-out data of a sub-domain is collected and concatenated into a 1D array on an add-up manner. Figure 3-8 illustrates how the data in overlapping zones are grouped in a chunk of continuous memory. Assume a problem is executed with four processors, and partition 1 is adjacent to partition 0, 2, and 3. There are 3, 4, and 5 entries of data in the overlapping zones that need to send to neighbor partitions 0, 2, and

3, respectively. A temporary array is then created to accommodate the concatenation of prepared data. The pointer of this array is passed to a subroutine for exchanging data with neighbor partitions, which uses the non-blocking send and receive to post the communication requests. The procedure of receiving data from neighbor partitions is preallocating 1D array beforehand for arriving data. An *MPI_WAIT* is placed at the end of the initialization of send and receive subroutines to check if procedures are completed correctly. The pseudo code of the communication procedure is shown in Figure 3-9.

The performance of our methodology is evaluated with a one-million-cell problem. The problem is executed on the NYX machine in the Center of Advance Computing of the University of Michigan. The nodes used are comprised of Intel Nehalem Xeon E5540 CPU and InfiniBand networking whose latency is about 10^{-6} second. Each Intel Xeon E5540 CPU has four cores with 8M bytes L3 shared cache and total memory available per CPU is 12G bytes. The averaged data size exchanged in a process of communication is from 108K to 41K bytes on 4 to 96 processors. The start-up time represents the preparation and calling of MPI_ISEND and MPI_IRECV procedures, and the time to finish synchronization stands for time staying on MPI_WAITALL. The cost of one data synchronization is in order of 2×10^{-5} second and nearly independent of number of processers used. This test suggests the communication strategy used in present work has low overhead for these cases. This communication design is applied to data exchange across partition-boundaries for both Eulerian domain and Lagrangian domain in the present work.

```
1    accumulate_send = 0; accumulate_recv = 0
2    number_of_recv[k] : number of entries received from partition k
3    number_of_send[k] : number of entries sent to partition k
4    DO index0 = 0, total number of processors - 1
5        IF NOT neighbor partition, CYCLE
6        accumulate_recv = accumulate_recv + number_of_recv[index0]
7        recv ← recv_data[accumulate_recv]
8        source = index0
9        tag1 = processor_id
10       CALL MPI_IRECV(recv, # of receive, type, source , tag1, comm., reqs)
11       accumulate_send = accumulate_send + number_of_send[index0]
12       send ← send_data[accumulate_send]
13       tag2 = index0;  dest. = index0
14       CALL MPI_ISEND(send, # of send    , type, dest.   , tag2, comm, reqs)
15   END DO
16   Any computation work that does not require coming data can be placed here.
17   CALL MPI_WAITALL(reqs, stat, comm)  or MPI_TEST(reqs, stat, comm)
```

Figure 3-9. A non-blocking send/receive communication algorithm for exchanging data with neighbor partitions.

Similar efficient communications are achieved for most of point-to-point, cross-partition-boundary communications. In case of the size of send-out data larger than 10M bytes a time, a significant delay of the process completion is observed. This is primarily due to the saturation of communication buffer on a computational node. Using a buffered send *MPI_BSEND* can alleviate this problem, and insert more computation work in

between the request of send/receive and *MPI_WAITWALL* can hide this communication overhead. A more proactive strategy is using non-blocking call *MPI_TEST* to check whether data arrive for certain requests, and proceeding possible computation that is not dependent on incoming data.

Table 3-1. Communication cost for velocity field in a one-million-cell problem.

# of processors	4	8	16	64	96
Start-up time [sec]	3.0×10^{-5}	2.4×10^{-6}	2.0×10^{-5}	2.1×10^{-5}	2.1×10^{-5}
Time to finish synchronization (sec)	9.5×10^{-6}	9.7×10^{-6}	9.5×10^{-6}	1.0×10^{-5}	1.1×10^{-5}
Averaged data size exchanged (K Bytes)	108	105	84	50	40.5

3.5 Cell-based unstructured AMR

The AMR method utilizes a cell-by-cell isotropic adaptation. The Eulerian domain consists of Cartesian cells, while the connectivity is in unstructured data format. Figure 3-10 shows an example of Cartesian grid with cell indices and its corresponding graph *G(V,E)*, where *V* is the vertex (cell) and *E* is the edge (face) of a graph element. The graph represents the connectivity of the global Eulerian grid. Due to lack of a logical structure for a vertex to refer neighborhood by indexing, an unstructured domain usually preserves neighborhoods' connectivity for the load balance procedure to simplify the implementation complexity. Since a global Eulerian graph is maintained in each processor, memory requirement may be a concern. The information of global Eulerian graph contains information of a cell's surrounding faces and a face's adjacent cells. Figure 3-10 gives an estimation of the memory cost of our design. N is the grid size and *p* is the number of processors used in a computation. Storing global Eulerian graph is the leading cost of the system memory. The connectivity of a global Eulerian graph costs 85N bytes, and 905 N/p bytes of memory goes to variables and data belonging to a local Eulerian sub-domain. For example, a distributed memory system equipped with 3G bytes RAM per processor can spend 1700M bytes storing global connectivity for a 20 million grid points problem and the rest of memory storing local, distributed variable arrays. Although memory requirement constrains the applicability of a current design to very large problems, the cell-based unstructured AMR has advantages of no cross-processor communication for inter-level interpolation, constant data-fetching rate for all levels of grid, and flexible refinement area. These properties make the field equation solver favorably efficient regardless of the number of refinement levels applied on the complex Lagrangian interfaces.

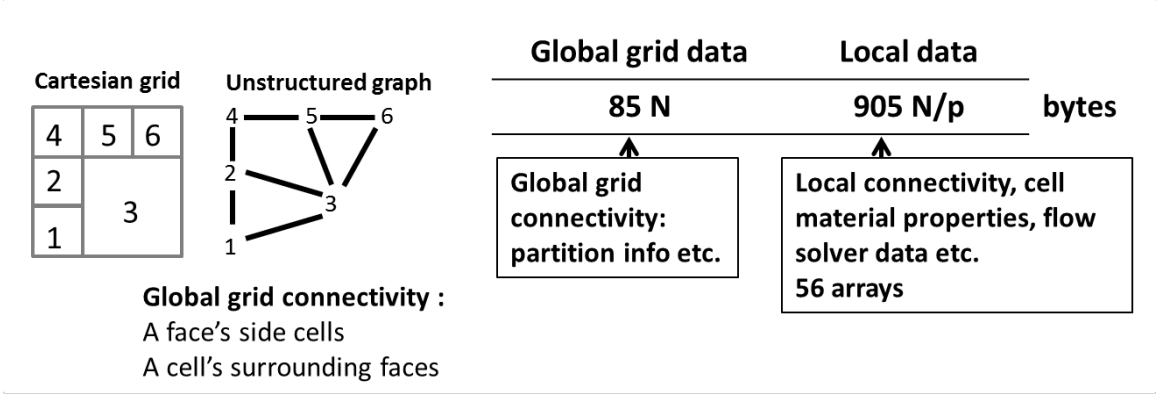


Figure 3-10. Memory cost of the Eulerian-Lagrangian method with cell-based unstructured AMR method.

3.5.1 Adaptation criteria

Grid adaptation is based on the location of interfaces and solution gradient on the Eulerian partitions. The geometry-based adaptation promotes grid around the neighborhood of interfaces to a designate level of resolution. The range of refinement around interface determines frequency of AMR required. In our design, the highest-level Cartesian grid must encompass interfaces all the time. With a wider range of refinement, interface stays in the highest-level grid longer such that the execution frequency of AMR can be reduced. However, larger refinement area may introduce inefficient mesh usage. In current study, the width of the highest refinement region is five times that of the finest-cell width on each side of interfaces. Cells far away from the interface are adapted based on the solution of the flow field. The decision to refine or coarsen a cell is determined by comparing the vorticity of a cell to standard deviations of vorticity of the entire Eulerian domain, as show in Eq. (25).

$$\xi_i = |\omega_i| = |\nabla \times \mathbf{V}_i|,\ i \in \Omega_p$$
$$\sigma_s = \sqrt{\left(\sum_{p=1}^{n} \sum_{i=1}^{m} (\xi_{i,p} - \bar{\xi})^2 \right) / N} \qquad (25)$$
if $\xi_i > \sigma_s$, adaptation_flag[i]=1
if $\xi_i < 0.2\sigma_s$, adaptation_flag[i]=-1

If the energy equation is solved, temperature gradient is another adaptation criterion. Adaptation flag "-1" is assigned to "coarsening" candidates and "1" to "refining" candidates; otherwise it is "0".

---

1 Create an temporary sub-domain connectivity $\Omega s_p^{'} = \Omega s_p$

2 adaptation_flag [ ] = adaptation flag that has been assigned by previous adaptation check subroutines

3 DO WHILE any of the adaptation_flag of C_p is equal to 1, $\forall C_p \in \Omega s_p$

4 DO i = 1, number of cells in Ωs_p

5 IF adaptation_flag[i] = 1 THEN

6 IF any neighbor of C_p^i is on lower grid level THEN

7 adaptation_flag[neighbor] ← 1

8 CYCLE

9 ELSE

10 Split cell C_p^i into 8 children; modify grid connectivity on $\Omega s_p^{'}$

11 adaptation_flag [i] ← 0

12 END IF

13 END DO

14 END DO WHILE

15 DO i = 1, number of cell in Ω_p

16 IF adaptation_flag [i] = -1 AND not at partition boundary THEN

17 IF all neighbors are on coarser or the same grid level THEN

18 Merge cell C_p^i with corresponding siblings and modify grid connectivity on $\Omega s_p^{'}$

19 adaptation_flag [i] ← 0

20 END IF

21 END IF

22 END DO

Figure 3-11. Grid generation algorithm.

3.5.2 Parallel grid generation

Figure 3-11 illustrates the local grid generation algorithm. The grid adaptation is applied on the Eulerian sub-domain Ωs_p, but only the grid generated in a partition Ω_p is adopted as a new grid. The reason for applying refinement on the overlapping zones is for the recursive refinement procedure as shown in Figure 3-12. Any refinement-flagged cell has to be refined for encompassing the Lagrangian interfaces with the finest resolution. Since we do not refine a target cell with a coarser neighbor, the coarser neighbor has to be refined beforehand and then the target cell can be refined. This rule results in the scheduling of splitting cells on consecutive cells. Extending the refinement to the overlapping region can guarantee consistent refinement results on partition boundaries. The coarsening process is applied on a partition and avoids two scenarios: a cell having

neighbors on higher grid level and at partition boundaries. At the beginning of grid refinement, a set of a local grid Ω_p' is created, including cells tagged with the refinement flag and their neighboring cells. This local grid contains partial grid connectivity of the sub-domain Ωs_p which the tentatively modified connectivity due to refining and coarsening operation can be placed on. The refinement routine recursively operates over this local grid to generate new cells and faces on new graph of Ω_p'. Whenever the rules are followed, the coarsening operation merges cells and faces that originally exist in the finer resolution level. The indices of the cells and faces designated to merge are deactivated. Then the modified connectivity of the merged cells is placed on Ω_p'.

For a refining operation, the cell-centered pressure and temperature and face-centered velocity of new cells and faces are constructed linearly by the information of surrounding cells. The variables of merging cells due to coarsen operation are averaged using the corresponding cell-centered or face-centered values of original cells. The discretization of convection term and Poisson equation on hanging nodes are treated by the approach of Singh and Shyy [1].

The grid generation algorithm generates a new Eulerian grid independently on each processor. In case of an even distribution of refinement-tagged cells among all processors, this procedure is load balance. For Lagrangian interfaces staying at certain Eulerian sub-domains, the refinement is applied to those sub-domains alone, which induces load imbalance. However, at this worst scenario it is not helpful to adopt other parallelisms such as task (procedural) decomposition because procedure dependency of AMR avoids possibilities of task overlapping. Although the load imbalance of grid

generation is inevitable for extreme cases, moderate scalability is still expected.

This size of the pre-refinement region allows user-defined and could be treated as an accommodation buffer for evolving interfaces that may reduce the frequency of required AMR operations.

3.5.3 Global indices and connectivity construction

The new graph information is localized before this step. A new graph of an Eulerian sub-domain after local mesh generation is independent and is not known by its neighbors. For example, a shared face on partition boundaries is broken due to the cell refinement in processor 1 but is intact in processor 2. However, the new local graph of the sub-domain Ωs_1 and Ωs_2 are legislative individuals, it is required to coordinate the modified information of this face for updating the global Eulerian graph and domain decomposition. This process of connectivity synchronization is a typical challenge of parallel grid generation in terms of efficiency since synchronization and index ordering relies on either intensive communication or serialization. For instance, some approaches remesh the partition boundaries faces before or after remeshing the interior regions, which increases the ratio of serial operation in an AMR implementation [82]. The construction of the global indices and connectivity for new graph in parallel requires all-to-all data exchange. In current study, we accomplish this task by three steps: an add-up indices function to give global order of the created cells and faces in each Eulerian partition, an all-to-all communication to collect connectivity of modified cell-face relationships, and then an update of global graph by the collected information.

We separate the indices generation and connectivity process by facilitating two

data classes containing the modified cells and faces information to accomplish the construction of a new global graph $\Omega^{"}$. The data classes store modified cells and new generated faces during the refining and coarsening processes. The data structure of modified cells Mc_p possesses the original global indices of the broken and merged cell c_k and its sibling cells. Eq. (26) is the data structure Mc_p on two-dimensional Eulerian domains.

$$
\begin{aligned}
&Mc_p = \{c_1, c_2, ..., c_k\}_p, \\
&c_k = [g_i, s_{3k+1}, s_{3k+2}, s_{3k+3}]^T, \text{ for spliting operation} \\
&c_k = [-g_i, g_{i2}, g_{i3}, g_{i4}]^T, \quad \text{for merging operation} \\
&k \text{ is the number of modified cell group in processor } p \\
&g_i \text{ is the global index of parent cell i} \\
&s \text{ is the index of new children cells due to refinement}
\end{aligned}
\tag{26}
$$

Because sibling cells' global indices are unknown beforehand, temporal, sub-domain based indices as denoted by s_k are assigned to them. In case of cell merging operation, negative signs are symbolically used to the global index of parent cell g_i, which will be the resulting index representation after the merge operation. The other three or seven original (on 2D and 3D domain) cells denoted in c_k will be deactivated. The other data structure called new face information Nf_p contains all faces that are generated during the refining process in the partition Ω_p.

$$Nf_p = \{f_1, f_2, ..., f_m\}_p, \text{ m is the number of new faces in processor } p \tag{27}$$

It possesses a new face's orientation, side cells index, and parent face index if generated through splitting a face. In case new faces are bounded by new cells just

generated, we refer the cell index to the coordinate of Mc_p. New faces generated at partition boundary are pre-processed by a founction reconstructing broken faces pairs. The routines go through all boundary face ${f^i}_p \in \partial\Omega_p$ and check the "broken" status of their corresponding face in adjacent partitions. For a face pair, both of which are marked in new face data array Nf_p, only the primary processor (lower rank ID) creates new global index that avoids the duplication of the face index.

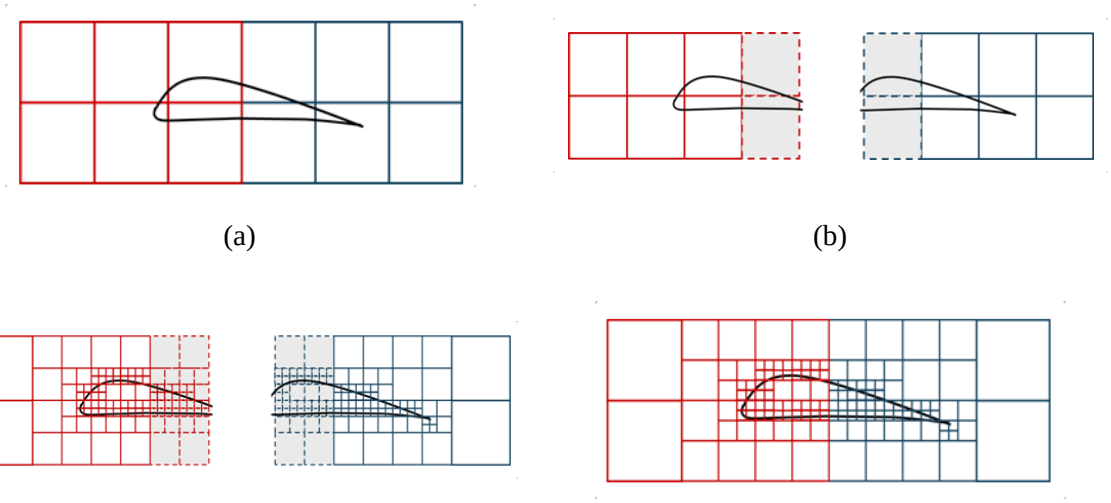
An all-to-all collective communication is used to broadcast Mc_p and Nf_p to all other processors so that every processor has the collection of Mc_p and Nf_p from all processors. The collection of Mc_p and Nf_p is in the concatenated arrays as show below.

$$\begin{aligned} Mc &= (Mc_1, Mc_2, .., Mc_n) \\ Nf &= (Nf_1, Nf_2, .., Nf_n) \end{aligned} \tag{28}$$

Once each processor has a full knowledge of the number of added and merged cells, we assign the global indices to new cells due to refinement operation in an add-up sequence by looping through *Mc* array. For example, the number of added cells on a three-processor computation is [10 11 9], and the original total size of cells is 50, the new global indices of new cells are [51:60 61:71 72:80]. Thus, the assembled contiguous indices can be used to update the cell connectivity by renumbering the obtained local arrays. In case of merged cells, inactive tags are assigned to them. Inactive cells and faces do not participate in the following load balance operation and the solution marching process. The face indices are assigned by the similar add-up method. The final procedure to grid up a new global graph is collecting those faces composing a global cell to form a cell-face list.

After the creation of *Mc* and *Nf* by all-to-all collective communication, every processor operates on an identical set of new grid data. As a result, the updating of connectivity of global Eulerian graph is a serial computation. In addition, it requires two times of all-to-all communication to accomplish the updating of global connectivity with the present algorithm. Performance deterioration due to frequent communication during the adaptive refinement process is minimized in the current approach. Furthermore, we do not grid up the boundary and interior of a partition separately, but single-time grid generation on each partition is satisfactory.

The entire process of remeshing has two advantages: First, it is a fully decoupled method when generating a new grid on a sub-domain such that communication is not used in the local grid generation. The remeshing of the interior and boundaries of a partition is applied at the same time, which avoids the sequential update of the interior and boundaries of a graph [82, 83]. Second, the updating of connectivity of the global Eulerian graph concentrates on the adaptation-affected regions, which is effective in terms of computation.



(a) (b)

(c) (d)

Figure 3-12. An example of the geometry-based grid refinement using two processors. (a) Eulerian partition Ω_1 and Ω_2 are denoted in red and blue grid with an airfoil-shape immersed boundary. (b) Eulerian sub-domain Ωs_1 and Ωs_2. Shaded cells are overlapping zones of each sub-domain. (c) The recursive refinement is applied to the Eulerian sub-domains. (d) The grid generated in the overlapping zone is discarded. An algorithm corrects and synchronizes the global face indices at partition boundary.

3.5.4 Domain re-decomposition and data migration

We rebalance the Eulerian computation after the grid adaptation by the heuristic partitioning library, ParMETIS. It handles multiple objectives optimization on redistribution cost, load balance, and number of edge-cut together. Because the new global grid needs a set of global ordered index in CSR format as the input of ParMETIS, we use a reorder routine assigning natural index system to active cells in each partition and then combine the assignments from all processors to form a new set of global index system. An "active" cell means a cell staying in a partition domain and being an effective identity. An inactive cell is a discarded index referring to an obsolete cell due to the coarsening operation as shown in Figure 3-13. The deactivation approach is convenient for data structure operation at this stage. A memory adjustment operation will remove inactive index pointers in the latter procedure.

Before discarding the original sub-domain, information such as cells' coordinate, level, cell-centered material, pressure, and temperature are collected in a set of data arrays for redistribution. Data redistribution may be communication-intensive when an original partition barely overlaps with the new partition. If the AMR is performed at an adequate interval, a re-partition operation should return a new partition with little change. Diffusive scheme of re-partition routines ParMETIS_V3_AdaptiveRepart usually leads to moderate data migration cost based on our experience.

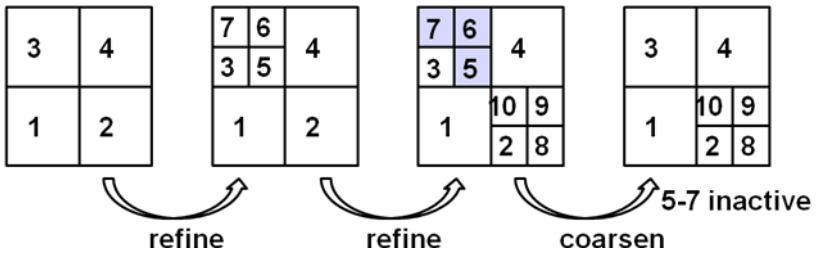


Figure 3-13. Cell order and deactivation of the current cell-based unstructured AMR. Splitting one cell and adding three more indices to three children cells for the refining operation and merging four cells by deactivating three cell indices for the coarsening operation.

We organize cell-centered and face-centered data to arrays of user-defined types. As a result, two messages (one for cell-centered and one for face-centered data) are enough to accomplish data migration between two processors. These data chunks are exchanged by the non-blocking send/receive communication addressed in section 3.4. Once the redistribution of data is completed, we discard the original Eulerian sub-domains, and then construct new Eulerian sub-domains and all variables. The Lagrangian sub-domains are redefined based on the new Eulerian sub-domains.

3.6 Scaling of computation and communication cost

Computation-to-communication ratio primarily determines the parallel performance of procedures. We summarize the computation and communication cost in Table 3-2 to illustrate how procedures scale with problem size. Here, N is the grid size of a problem. There are four categories of the procedure types: Eulerian task, Eulerian-Lagrangian task, Lagrangian task, and AMR. The Eulerian tasks refer to the computation on the Eulerian frame, such as the field equation solver. The leading cost of the Eulerian computation is solving the linear sparse matrix of pressure Poisson equation, which is, at most proportional to $O(N^3)$. Communication required for Eulerian tasks are data at

partition boundaries, which are in the size of "surface" and proportional to $O(N^{2/3})$. For Lagrangian tasks, computation and communication scale with the Lagrangian surface area $O(N^{2/3})$. For AMR using geometry-based refinement, the computation scales with $O(N^{2/3})$, and the data needed to exchange is as large as the volume of an entire domain at the process of constructing global grid connectivity and reordering. As a result, for a fixed-size problem, Eulerian tasks have the highest computation-to-communication ratio and AMR has the least among all. Tasks such as surface tension calculation involve Eulerian-Lagrangian interactions have computation cost between O(N) and $O(N^{2/3})$. While the communication cost of the Eulerian-Lagrangian tasks is proportional to $O(N^{1/3})$ with the current Lagrangian domain decomposition method, it can be more than O(N) by adopting a load-balanced decomposition without considering the data locality of Lagrangian markers.

Table 3-2. Scaling of computation and communication cost of the parallel, adaptive Eulerian-Lagrangian interface tracking method.

	Computation	Communication
Eulerian tasks	$O(N^2 \sim N^3)$	$O(N^{2/3})$
Eulerian-Lagrangian tasks	$O(N \sim N^{2/3})$	$O(N^{1/3})$
Lagrangian tasks	$O(N^{2/3})$	$O(N^{1/3})$
AMR	$O(N^{2/3})$ (geometry-based)	O(N)

3.7 Solving procedure

The procedures of the parallel Eulerian-Lagrangian method are summarized in Figure 3-14. Note that the operations denoted in italics involve both Eulerian and

Lagrangian variables. First, the initialization of simulation creates a Cartesian grid to encompass solid interface and refine it adaptively to designated resolution with the identification of ghost cells. The Eulerian domain is then partitioned by ParMETIS, and the Lagrangian domain is spatially decomposed based on marker's locality. In the time integration loop, a marker is updated to new position by a velocity field interpolated from the Eulerian domain and re-meshing may be applied to interface whenever needed. The updated interface renders a new distribution of fluid properties that inter-processor communications are needed to update fluid density, viscosity, and thermal conductivity at overlapping zones of sub-domains. Source terms surface tension and mass transfer across the fluid interface distribute their impact to cells at a fixed distance. The velocity field is updated with the Runge-Kutta/Crank-Nicolson integration by the projection method. Based on the predicted velocity field $\mathbf{v}^*$ and the prescribed execution period, dynamic AMR invokes subsequent procedures: parallel mesh generation, domain re-decomposition, data migration, and generation of new Eulerian and Lagrangian sub-domains. The detail of parallel adaptive mesh refinement is explained in previous sections. PETSc and hypre are both implemented as the linear solver for the discretized Poisson equation. Once the divergence-free velocity field is obtained on an Eulerian domain, velocity is projected to Lagrangian markers by the discrete Dirac delta function. The topology reconstruction is invoked when required.

1 Initialization

2 Eulerian domain decomposition (ParMETIS)

3 Lagrangian domain decomposition

4 Time integration loop :

5 Marker movement

6 Interface modification: smooth, refine, coarsen

7 *Lagrangian domain decomposition*

8 *Determine cell-centered material and properties*

9 *Source term computation: surface tension and mass transfer/ heat flux*

10 Intermediate velocity $\mathbf{v}^*$ by RK-CN integration

11 IF : Dynamic adaptive mesh refinement

12 Parallel mesh generation

13 Domain re-decomposition(ParMETIS)

14 Data migration

15 Rebuild Eulerian sub-domain and *Lagrangian sub-domain*

16 Pressure Poisson equation hypre/PETSc

17 Corrected velocity $\mathbf{v}^{n+1}$

18 Energy equation by RK-CN integration

19 *Velocity interpolation to Lagrangian markers*

20 Topology change test

21 IF merge or breakup : *Topology reconstruction of the Lagrangian interfaces*

Figure 3-14. The solving procedure of parallel Eulerian-Lagrangian method with cell-based unstructured AMR.

Chapter 4.

Validation and performance

4.1 Validation

The parallel, adaptive Eulerian-Lagrangian solver is validated for the implementation of the ghost cell reconstruction of the sharp interface method and moving boundary tracking with continuous interface method. The computations involving the sharp interface method include internal, isothermal and thermal flows and external flows. Moving boundary computations are validated for single bubble rising and binary droplet collision at low Weber number. Results of the current approach are compared with the solutions from literature. All of the cases shown here are solved with the third-order ENO scheme around the fluid boundaries and the second-order central difference scheme in the rest regions.

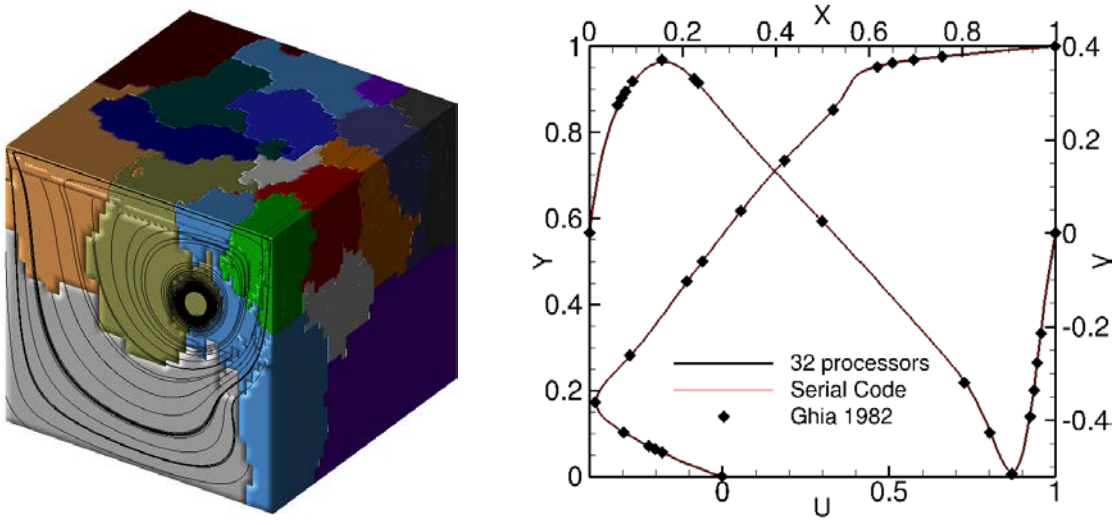
4.1.1 Cavity flow

The first case is a lid-driven cavity flow at Reynolds number 1000 with grid size 80 x80x80. The computational setup is presented in Table 4-1. We use a 3-D domain to simulate the 2-D flow by assigning periodic boundary condition at the front and back of the cube. The solid boundary conditions are enforced by the ghost cell reconstruction of the sharp interface method. Figure 4-1(a) shows partitions and streamline at a certain time step. The steady state solution from a 32-processor computation is given in Figure 4-1(b)

together with a benchmark work of Ghia et al. [84].

Table 4-1. Computational setup of validation on cubic cavity flows

	Lid-driven flow	Natural convection flow
Non-dimensional parameter	$Re = 10^3$	$Ra = 10^5$
Grid	Uniform 80x80x80	Uniform 40x40x40
Solid boundary conditions	Sharp interface method	
# of processors	32	16

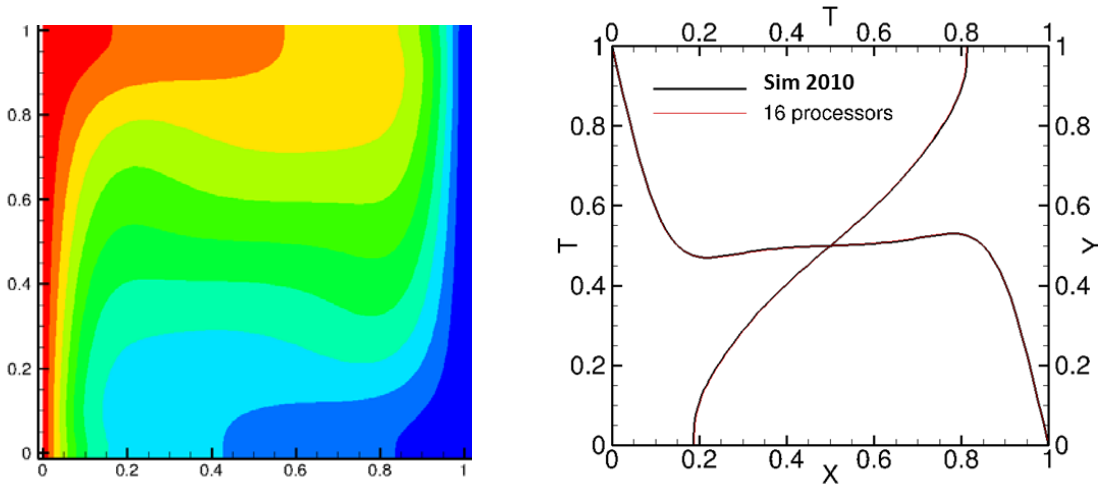


(a) (b)

Figure 4-1. Lid-driven cavity flow at Re = 1000. (a) Stream line of a 16-processor test at the developing stage of recirculation. The colored blocks represent different partitions. (b) Velocity component at x and y coordinates at the central vertical line and central horizontal line of cavity.

A test of natural convection driven by buoyancy force is presented in Figure 4-2 to validate the parallel implementation of the energy equation. The fluid is initially static in the cavity with a high-temperature side wall at left, low-temperature side wall at right, and adiabatic wall at top and bottom. The sharp interface method including the reconstruction of temperature and velocity and pressure at the ghost cells is used as the enforcement of the boundary condition of solid interface. For such a case, the Rayleigh

number is 10^5, and the resulting temperature distribution at a steady state is shown in Figure 4-2(a) for temperature contour and Figure 4-2(b) for temperature profile at the horizontal and vertical center of the cavity. Computational results confirm the solution produced by a parallel code is consistent with the solution from Sim [70] in an error range of 10^{-7}, which is the convergence criterion of the pressure Poisson equation.



(a) (b)

Figure 4-2. Natural convection in square cavity at Rayleigh number 10^5 on a 16-processor computation. (a) Temperature contour of natural convection. (b) Temperature along vertical center and horizontal center of the cavity.

4.1.2 Ghost cell method: Uniform flow past a circular cylinder/sphere

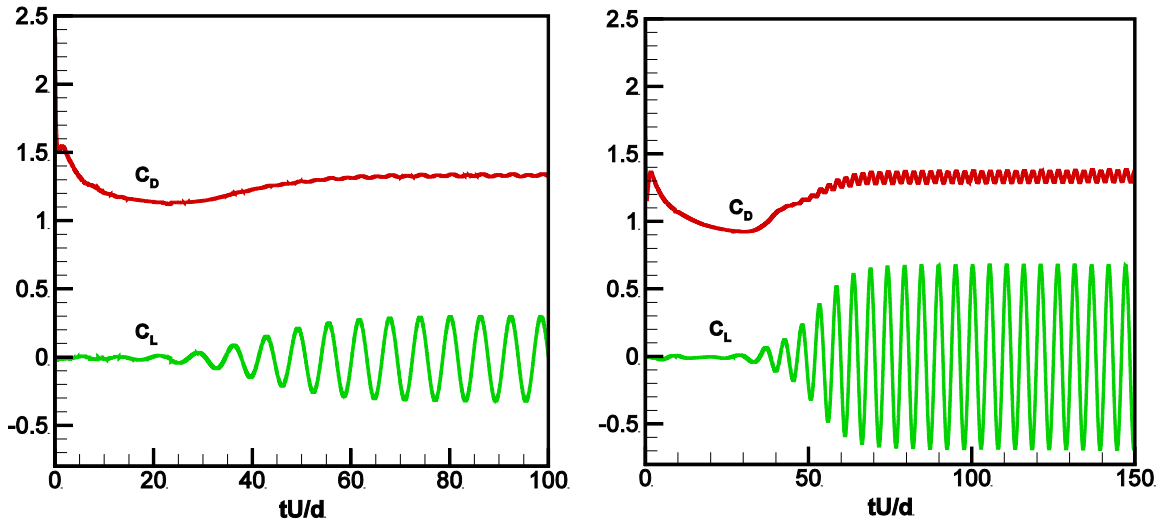
We validate the embedded pressure boundary condition and the ghost cell reconstruction at solid boundaries for problems of an immersed circular cylinder and sphere in the uniform flow. The tests of the uniform flow past a cylinder is at Reynolds number 40, 100, and 200 with static adaptive grid around solid interface on 32 processors. Table 4-2 is the computational setup, where D is the diameter of the circular cylinder or sphere.

Table 4-2. Computational setup of uniform flow past objectives.

	Flow past circular cylinder	Flow past a sphere
Reynolds number	40, 100, 200	200
Grid	D/Δx=32 (5 levels) D/Δx=64 (6 levels)	D/Δx=80 (6 levels)
Domain	30Dx20D	50Dx30Dx30D
Solid boundary conditions	Sharp interface method	
# of processors	32	64

The drag and lift coefficient, separation angle, length of recirculation zones, and Strouhal number are tabulated in Table 4-3. Here, results from references [27, 76, 85-87] are computed by immersed boundary method and reference [88] is from a body-conforming grid. At Reynolds number 40, the computed drag coefficient is 1.52 and the length of circulation zone is 2.23, which agree with the results from literature. At Reynolds number 100 and 200, the vortex shedding behavior is captured and non-dimensional parameters, Strouhal number, and periodic fluctuation of drag and lift coefficient are tracked in Figure 4-3. The computed Strouhal numbers are 0.164 and 0.192 for Reynolds number 100 and 200 with lift coefficient 0.31 and 0.68 respectively. Our results agree well with previous studies [85, 87] using immersed boundary method.

The sequential pictures in Figure 4-4 show histories of evolving vortex in the wakes behind the cylinder and a snapshot of adaptive grid configuration is presented in Figure 4-5. Highest-level grid dynamically follows the high vorticity regions. This simulation was computed on a 350x250 base grid with five levels of refinement that provides maximum spatial resolution 1/160 cylinder-diameter around the immersed solid boundary.



(a) (b)

Figure 4-3. Lift and drag coefficient of uniform flow past a circular cylinder. (a) Re = 100. (b) Re = 200.

Table 4-3. Comparison of drag coefficient, length of the recirculation zone (computed from rear end of cylinder), separation angle, lift coefficient, and Strouhal number.

Re	40			
	C_D	θ	L/D	Grid and resolution Δ =
Fornberg [88]	1.52	55.6	2.24	Body conforming grid
Ye [27] et. al. [27]	1.52	--	2.27	--
Taira and Colonius [85]	1.54	53.5	2.30	1/50D
Kim et. al. [76]	1.51	--	--	1/30D
Colonius and Taira [86]	1.55	--	2.20	1/50D
Linnick and Fasel [87]	1.54	53.6	2.28	1/87D
Current	1.60	51.8	2.06	1/32D
	1.54	53.6	2.23	1/64D

Re	100			200			Resolution Δ =
	C_D	C_L	St	C_D	C_L	St	
Fornberg [88]	1.058	--	--	0.829	--	--	Body conf.
Taira and Colonius [85]	1.34±0.048	±0.068	0.197	1.35±0.048	±0.68	0.196	1/50D
Kim et. al. [76]	1.33	0.32	0.165	--	--	--	1/30D
Colonius and Taira [86]	1.34±0.045	±0.068	0.195	--	--	--	1/50D
Linnick and Fasel [87]	1.34±0.009	±0.333	0.166	1.34±0.044	±0.69	0.197	1/87D
Current	1.33±0.017	±0.116	0.152	1.300±0.028	±0.48	0.176	1/32D
	1.33±0.010	±0.31	0.164	1.326±0.043	±0.68	0.192	1/64D

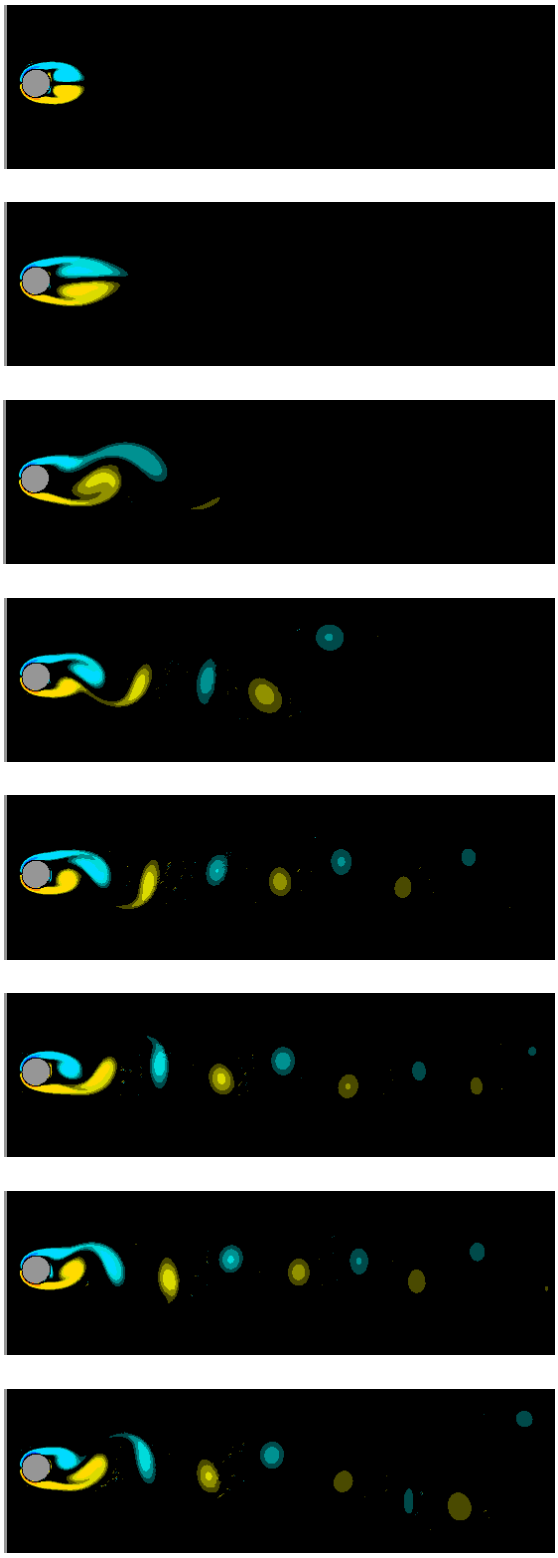


Figure 4-4. Vorticity contour of uniform flow past a circular cylinder at Re = 200.

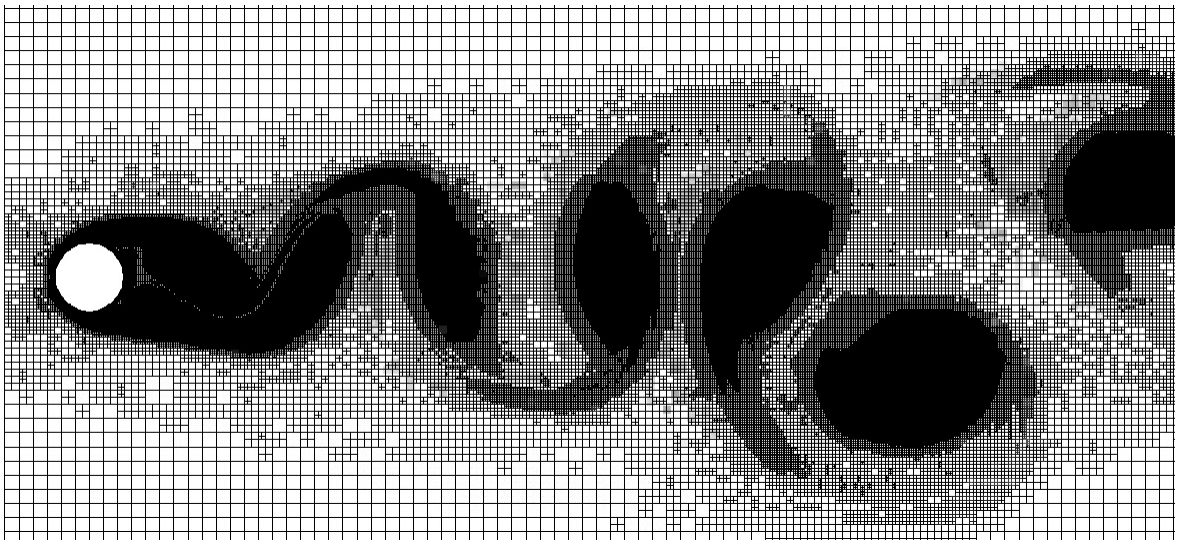


Figure 4-5. Dynamic mesh refinement based on vorticity.

Next validation is uniform flow past a sphere with 3-D AMR computation. At Re = 200, a steady recirculation is observed behind the sphere as shown in Figure 4-6. Features of a flow field at a steady state are compared with the results from body-conforming grid computation of Johnson and Patel (1999) in Table 4-4. We observe a smaller recirculation zone with the separation point slightly far than the results of Johnson and Patel. This difference may come from the ghost-cell reconstruction for enforcing boundary condition of immersed boundaries. This computation has a six-level refinement, and the resolution at the neighborhood of immersed boundaries is 1/80 diameter of the sphere. The problem size grows from 1.4 million to 5.5 million cells on a 64-processor computation. The pressure field is contoured on the sphere surface.

Table 4-4. Drag coefficient C_d, separation angle θ, length of the recirculation zone (X_s), center of recirculation zone (X_c, Y_c) at Re = 200.

	θ	X_s	X_c	Y_c	C_d
Johnson and Patel [89]	115	1.46	0.885	0.358	0.775
Present	118	1.42	0.836	0.345	0.756

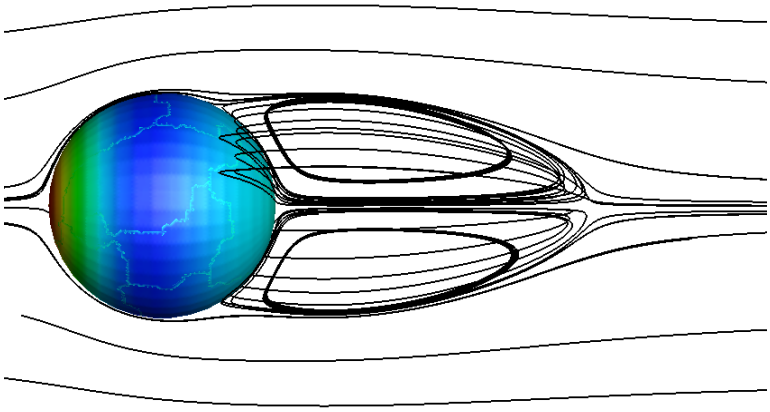


Figure 4-6. Pressure contour on a sphere and streamlines at Re = 200.

4.1.3 Moving boundary 1: Rising bubbles

In this section, several cases are presented to validate the interface-tracking algorithm with a dynamic adaptive refinement. These cases are buoyancy-driven bubbles with three sets of parameter combination between Morton number and Eötvös number. Computational setup is listed in Table 4-5. Here, D_0 is the initial drop diameter.

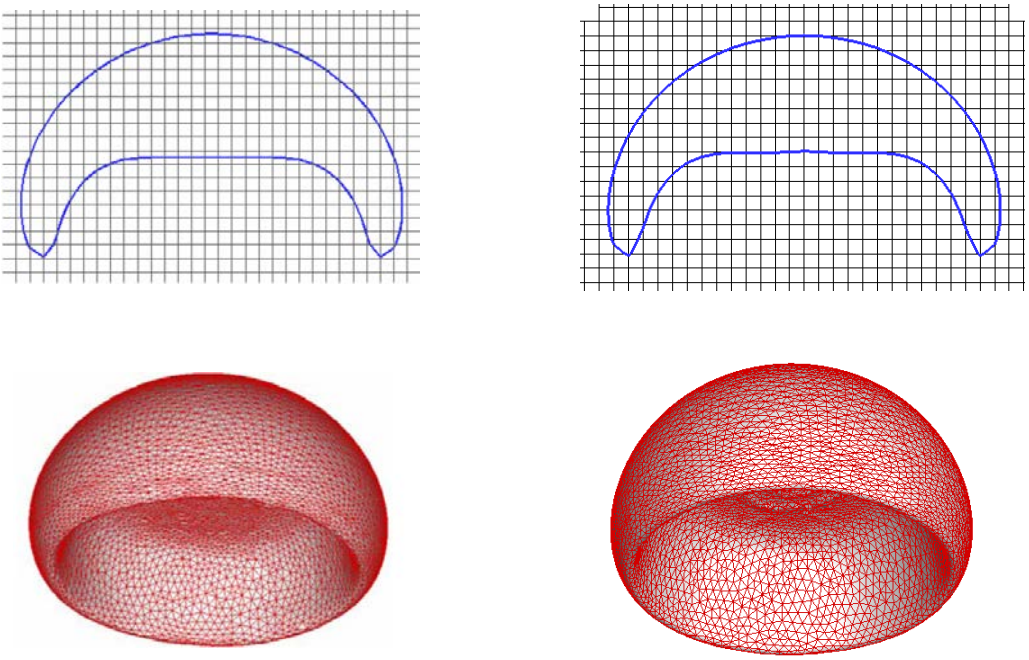
Table 4-5. Computational setup of rising bubble problems.

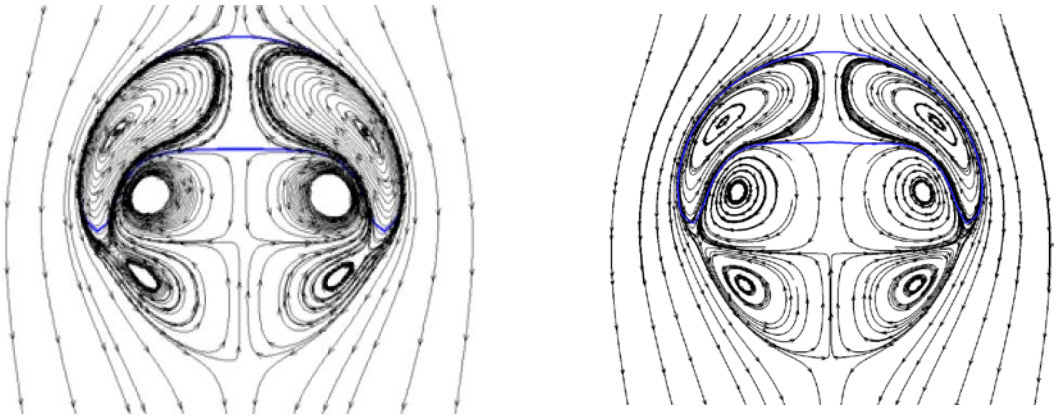
	Rising bubble problem
Morton number	$0.1\text{-}10^3$
Eötvös number	9.71-97.1
Density and viscosity ratio	100
Grid	$D_0/\Delta x = 19.2$ (3 levels)
Domain	$10D_0 \times 5D_0 \times 5D_0$
Boundary conditions	Slip boundary condition at side and bottom walls; Outlet boundary condition for top wall.
Fluid boundary condition	Continuous interface method
Lagrangian mesh modification	Smoothing/refining/coarsening
# of processor	8

We replicate three rising bubble computations in Singh [81] and Annaland et al. [90] to validate our parallel implementations, and compare terminal rising Reynolds with the experimental result of Grace [91]. For these cases, Morton number ranges from 0.1 to 10^3 and Eötvös number from 9.71 to 97.1, which cover three types of morphological deformation: ellipsoidal, dimpled ellipsoidal and skirted.

Table 4-6. Computation of rising bubbles: terminal shape of bubbles and rising Reynolds number.

	Parameter		Terminal shape	Rise Reynolds number			
	M	Eo		Grace [91]	Annaland et al. [90]	Singh [81]	present
(a)	$1.0x10^{-1}$	9.71	Ellipsoidal	4.6	4.3	4.6	4.6
(b)	$1.0x10^{3}$	97.1	Dimpled ellipsoidal	1.7	1.7	1.7	1.7
(c)	$9.71x10^{-1}$	97.1	Skirted	20	18	17.8	17.8





Singh [81] Present result

Figure 4-7. Indicator function contour, mesh and bubble shape, and streamlines of CASE (c).

These cases result in constant velocity, steady movements under the balance of buoyancy force and fluid drag. A rise Reynolds number can be defined based on the rising velocity and fluid properties. In Figure 4-7, indicator function at $I = 0.5$, mesh and streamlines from Singh [81] and current results are compared side by side at a steady movement stage of case (c). In Table 4-6, the rise Reynolds number of cases (a) and (b) is consistent with experimental results. For these three cases, averaged problem size is about 120,000, which is quite a small computation load for eight processors.

4.1.4 Moving boundary 2: Binary droplet collisions

These two demonstrations are cases adopted from Qian and Law [92]. Weber number and Reynolds number are similar between two cases but the histories of morphological change are very different due to impact factor. The impact factor is a ratio of the distance between the center of two droplets along the direction of movement and droplet diameter, which describes how off-axis of a collision is. Table 4-7 shows the detail setup of these validations. Note that four different grid resolutions are used in case

(a) for a grid sensitivity study.

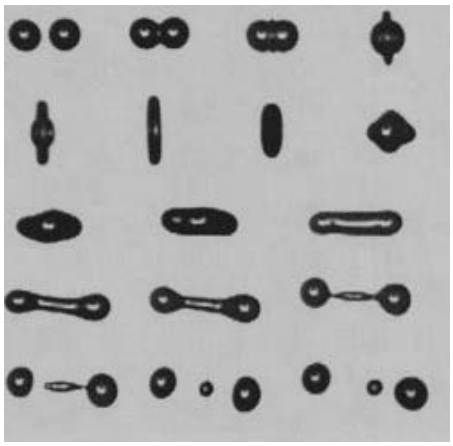
Table 4-7. Computational setup of droplet collision at low Weber number.

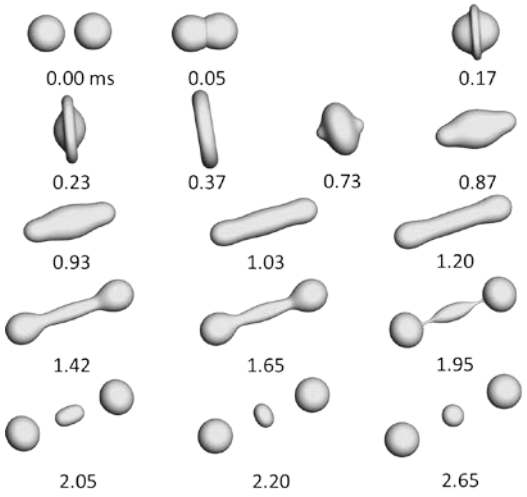
	Case a	Case b
Weber number	61.4	60.1
Reynolds number	296.5	302.8
Impact factor	0.06	0.55
Density and viscosity ratio	666.081 and 179.28	
Grid	$D_0/\Delta x = 16$ (3 levels) $D_0/\Delta x = 32$ (4 levels) $D_0/\Delta x = 64$ (5 levels) $D_0/\Delta x = 128$ (6 levels)	$D_0/\Delta x = 32$ (4 levels)
Domain	$9D_0 \times 4.5D_0 \times 4.5D_0$	
Boundary conditions	Outlet boundary condition for all sides;	
Fluid boundary condition	Continuous interface method	
Lagrangian mesh modification	Smoothing/refining/coarsening and topological reconstruction	
# of processor	32	32

Case (a) is a near head-to-head collision. The droplets merge into a single droplet and then the merged body breaks into three secondary droplets after a cycle of radical-to-axial deformation. Figure 4-8(a) shows the history of interfaces during the collision. The second case is off-axis collision. Two droplets coalesce into a single body. It stretches and twists for a while, then evolves into a two-head body with an elongating ligament connecting the two bulkheads and ends up with seven secondary droplets. Two primary bulk bodies are at the end sides and a centralized body sits in the center. There are two smaller satellites from the ligament in between the center body and the primary bulk bodies. The debris having a characteristic length as the width of a local Eulerian grid may shrink or disappear if reconstruction is applied again. This is an inherent consequence of

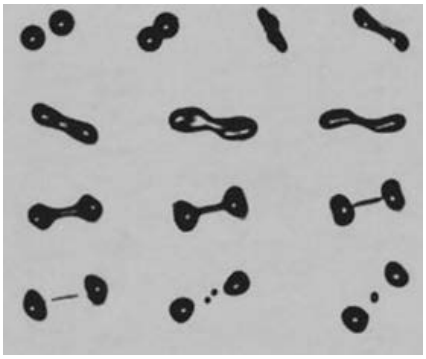
calling the reconstruction algorithm without enough resolution on the Eulerian grid. When the Weber number increases further, the interfaces present much smaller, subtle structures that such resolution cannot resolve interface features properly. This situation will be shown again in chapter 5 in detail. The collision histories of case (a) and (b) from experiments and computations are shown at the left and right columns respectively with real-time notation in Figure 4-8. The difference between experimental and numerical results is most likely because the collision plane is not parallel to camera plane in experimental works such that elongating bodies rotate with respect to the observing plane and other axis as well.

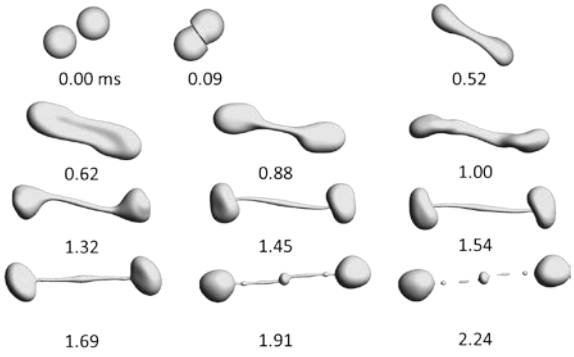
(a)





(b)





Qian and Law [92]

Present results

Figure 4-8. Binary droplet collision history at (a) We = 61.4, Re = 296.5 and impact factor 0.06. (b)We = 60.1, Re = 302.8, and impact factor 0.55.

A plot of Lagrangian interface, velocity vector and adaptive grid on plane y = 0 at non-dimensional time T* = 15.0 is shown in Figure 4-9. The adaptive grid effectively promotes grid around interfaces and high vorticity regions to the highest level in which the resolution Δx is $D_0/64$.

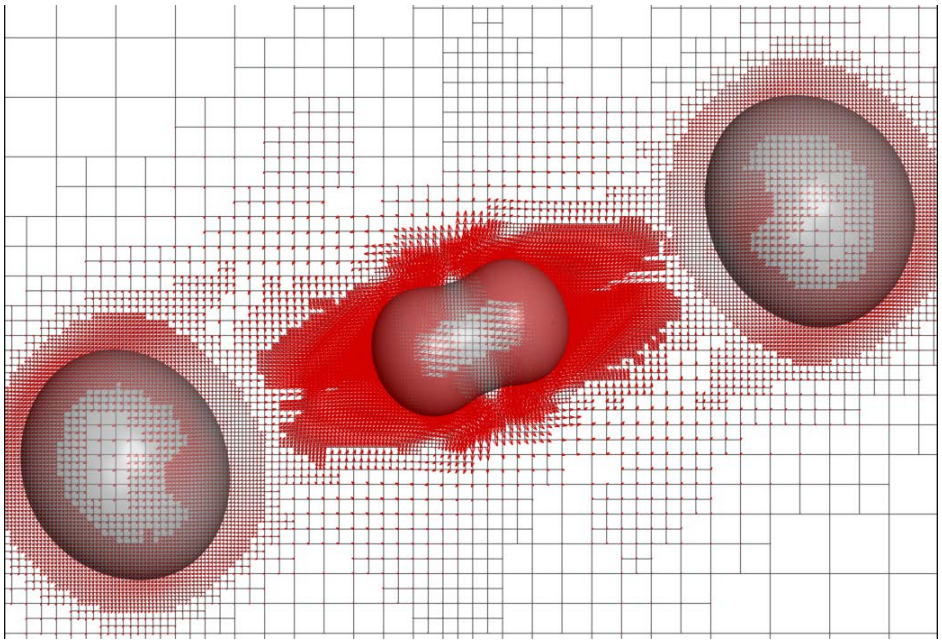


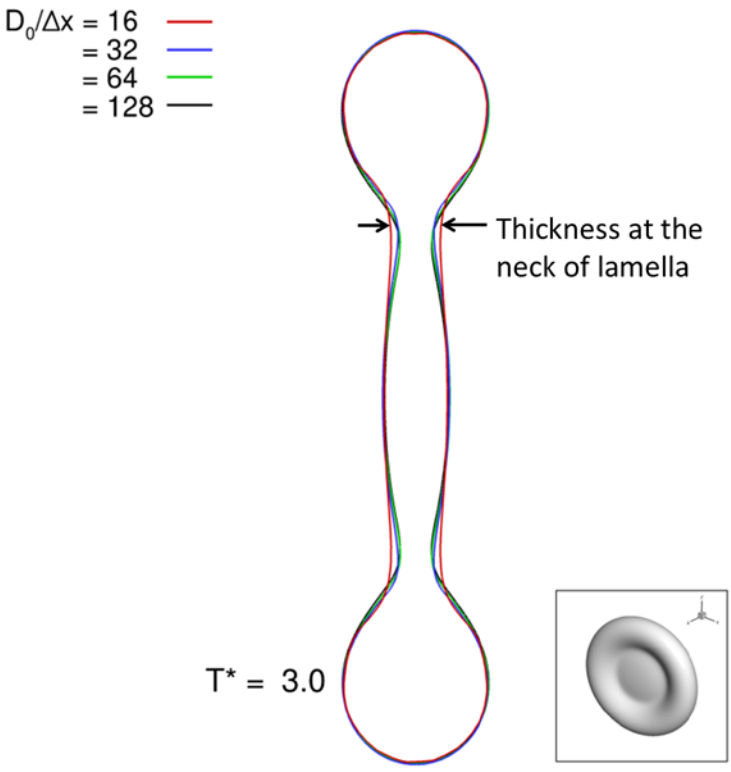
Figure 4-9. Top view of Lagrangian interface (grey), adaptive grid with velocity vector at T* = 15.0.

The grid sensitivity analysis repeats case (a) for four sets of grid resolution $D_0/\Delta x$ = 16, 32, 64, and 128. The interface profiles from resolution level at cut plane y = 0 are shown together to show discrepancies. Numerical error is evaluated by comparing the characteristic thickness, length and width of interfaces from the results of $D_0/\Delta x$ = 16, 32 and 64 with the results of the finest grid $D_0/\Delta x$ = 128 at three time snapshots, T* = 3.0, 5.0, and 7.0. Arrowed markers in Figure 4-10 denote the characteristic size used in the error estimation. For T* = 3.0, discrepancy on the neck thickness of the lamella is

evaluated by

$$\text{Error} = \text{abs}(h - h_{128}) / h_{128} \tag{29}$$

, where h_{128} is the neck thickness of solution with $D_0/\Delta x = 128$. Similar way of error computation is used for thickness at the center of the merged body at T* = 5.0 and length and width of interface profiles at T* = 7.0. The errors from different grid resolutions are compiled in Table 4-8 and Figure 4-11. Based on the table we can find errors of $D_0/\Delta x$ = 64 are mostly under 1% except the center thickness at T* = 5.0. From these three sets of numerical error, it is concluded that the simulation of this off-axis droplet collision at low Weber number generally converges with accuracy between first and second orders.



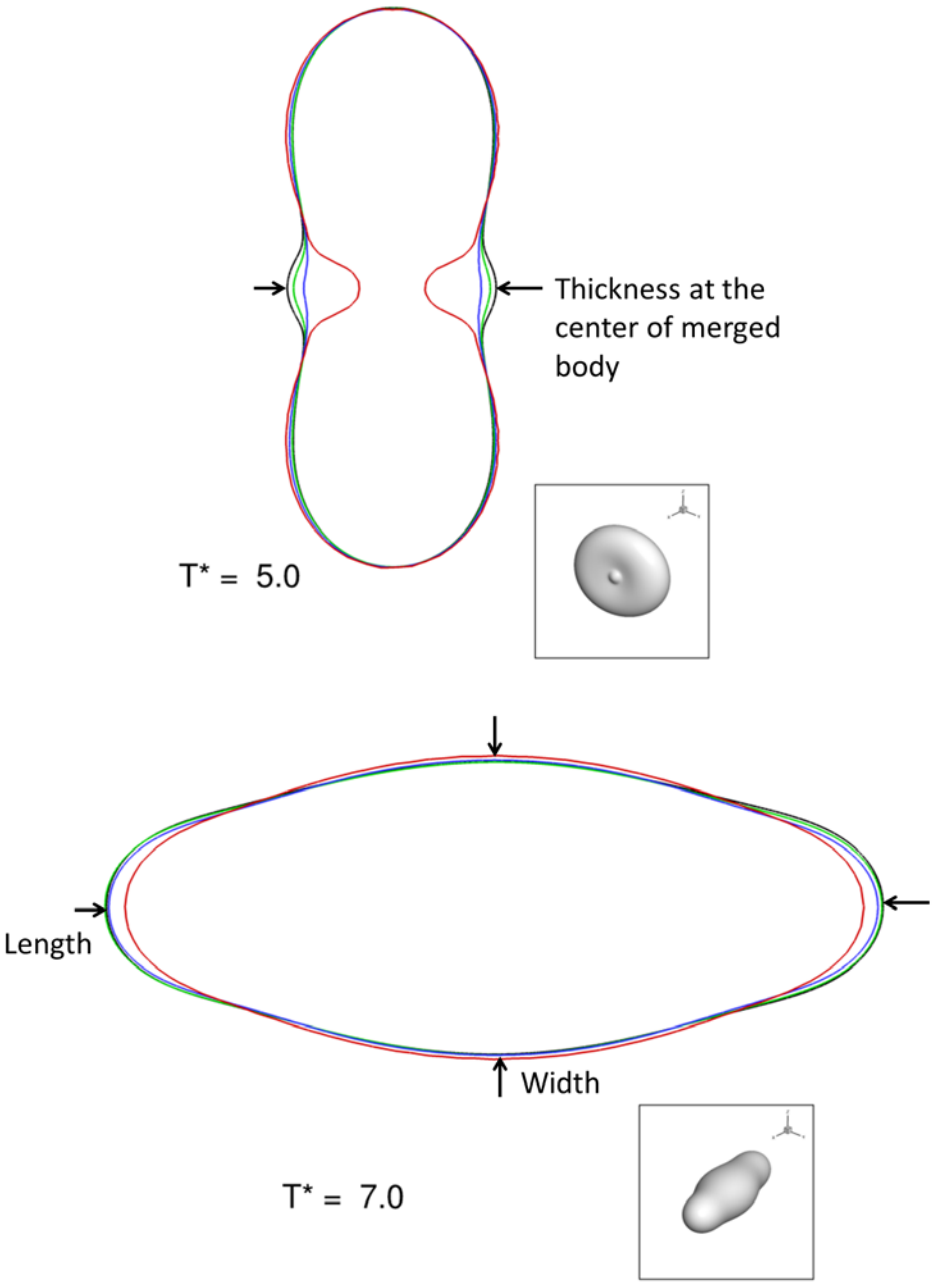


Figure 4-10. Interface profiles of case (a) on y = 0 cut plane at T* = 3.0, 5.0, and 7.0. Results from four resolution setups are represented by red, blue, green, and black lines for the finest grid $D_0/\Delta x$ = 16, 32, 64, 128, respectively.

Table 4-8. Error in the neck of the merged body at T* = 3.0, thickness at the center of lamella at T* = 5.0, and length and width of the merged body at T* = 7.0. Error is defined in Eq. (29).

Error types	$D_0/\Delta x=16$	$D_0/\Delta x=32$	$D_0/\Delta x=64$
Neck thickness at T* = 3.0	0.542	0.121	0.006
Thickness at the center of the merged body at T* = 5.0	0.680	0.150	0.058
Length of the merged body at T* = 7.0	0.052	0.012	0.002
Width of the merged body at T* = 7.0	0.039	0.010	0.004

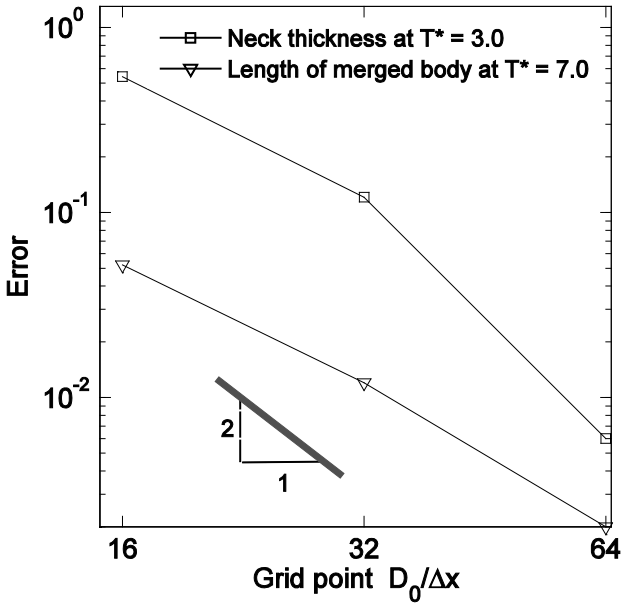


Figure 4-11. Error of the neck thickness and length of the merged body.

4.2 Performance

We first investigate the parallel performance of the field equation solver, the standalone strong and weak scalabilities of the cell-based unstructured AMR method, and then the overall performance of the present approach for a practical problem. These numerical experiments were conducted on the NYX machine with the machine specification as addressed in section 3.4 of communication strategies.

4.2.1 Performance of the field equation solver

The performance of the field equation solver is evaluated by computing the classic lid-driven cavity flow on three grid sizes—1.6×10^5 cells, 1.0×10^6 cells, and 4.0×10^6 cells—without dynamic AMR. The linear solver used here is the PETSc conjugate gradient method with Jacobi preconditioner [77]. Since the graph of a linear system of the discretized Poisson equation changes with the partitions, the iteration number reaching a constant rounding-error varies with the number of processors used. In order to control the workload spent on solving a linear system on a varying number of processors, the execution time of solving the Poisson equation is under the basis of the fixed iteration number of the linear solver. The number of iterations of the Poisson equation solver is fixed at a number such that the normalized residual is ensured to be less than 10^{-6}. The time spent on the solver is evaluated based on a wall-clock time of 10 time-step computation. The speedup is evaluated against a single processor computation for grid size 1.6×10^5 and 1.0×10^6, and an eight-processor computation for grid size 4.0×10^6. A summary of information about the performance test of the field equation solver is shown in Table 4-9.

Table 4-9. Performance test of field equation solver

Problem definition	Lid-driven cavity flow with ghost cell method for solid boundary conditions
Grid size	6×10^5, 1.0×10^6, and 4.0×10^6
Iteration number of Poisson equation solver	200, 700, and 1000 for the three sets of grid, respectively
Time recorded	Wall-clock time of 10 time-step computation

Overall, parallel performance is dominant by the efficiency of linear solver, which is controlled by two factors: the computation-to-communication ratio and shared cache size per CPU. For a fixed-size problem, increasing the number of processors decreases computation-to-communication ratio, and results in the slowdown of speedup. In case of shared cache effect, its impact on the parallel performance is related to the size of the working data in solving a linear system. A typical parallel program assigns a fraction of the entire data to each processor. However, the entire data set does not fit into the cache on a single processor execution but a parallel run executing with the same problem on multiple processors may have working data assigned to each processor that can fit in its local shared cache. In case of a computation having fully cached decomposed working data, a high hit rate and super-linear speedup are achieved.

Figure 4-12 shows the speedup of three cases. Favorable speedup is observed. Due to decreasing computation-to-communication ratio, speedup slows down with the increasing number of processors on 16 processors and 80 processors for grid sizes 1.6×10^5 and 1.0×10^6, respectively. For grid size 4.0×10^6, we see a large processor count outperform a small processor count because of cache effect. For an eight-processor execution, this problem has the working data of linear system solver excessively larger than the 8M-bytes L3 cache and results in a significantly low cache hit rate that

overestimates the speedup of multiple-CPU computation. The deterioration of efficiency arising from the decreasing computation-to-communication ratio is not observed in the scope of 240 processors.

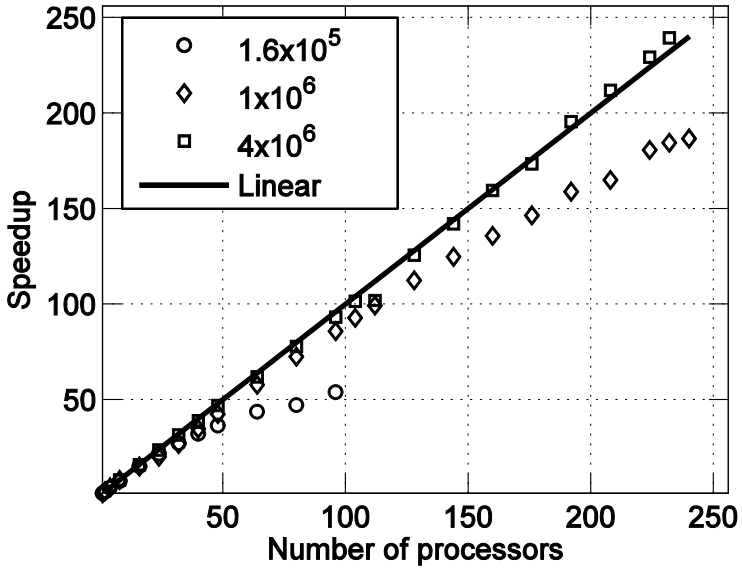


Figure 4-12. Speedup of the field equation solver for grid sizes 1.6×10^5, 2×10^6 ,and 4×10^6.

Figure 4-13 is the parallel efficiency based on the number of cells per processor. The flow solver has the best efficiency of $1.0\text{-}2.0\times10^4$ cells per processor. A degrading efficiency due to a decreasing computation-to-communication ratio is around 1.0×10^4 cells per processor. Using a large number of cells per processor has the efficiency around 1 but has an unfavorable cache hit rate for the procedure of solving the Poisson equation.

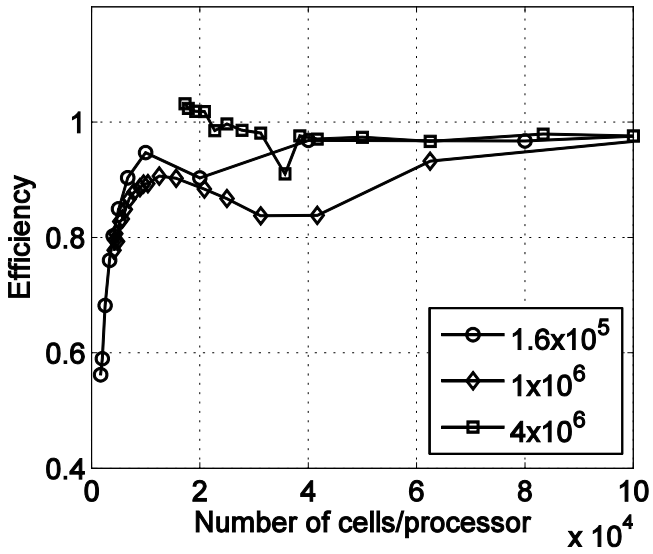


Figure 4-13. Parallel efficiency with respect to the number of cells per processor. The best usage of computational power of field equation solver is at 10,000 to 20,000 cells per processor. This range is determined by two factors: the computation to communication ratio and cache size of a system.

4.2.2 Performance of cell-based unstructured AMR

We study the parallel performance of AMR by conducting both strong and weak scaling studies for the present AMR approach. The refinement (or coarsening) is applied on an initial uniform grid by the random assignment of adaptation flags on every partition. We use a random assignment of the adaptation flag such that run time spent on the repartition is small and consequently data redistribution is not dominant. Each partition obtains 2% of cells having refinement flags that produce 14% of grid size increment, and the spatial distribution of the applied refinement is arbitrary. In the strong scalability test, we start with a grid size 8.19×10^6 and then reach about 9.3×10^6 after AMR. Because the data packing of our approach is unstructured and cell-based, the number of refinement levels in a domain will not affect the data-fetching rate theoretically. Hence, this experiment is considered one level of refinement. The characteristic of performances is

applicable to cases having multiple-levels refinement. Table 4-10 summarizes the setup of AMR performance test.

Table 4-10. The setup of strong and weak scaling tests of AMR

	Strong scaling	Weak scaling
Problem definition	Random-assigned refinement 2% of Eulerian cells are refined.	
Procedures involved	Adaptive flag assignment, remeshing, data redistribution, load balance, reorder cell, sub-domain construction	
Grid size (cells)	$8.19\times10^6 \rightarrow 9.3\times10^6$	6.4×10^4/processor
# of processor used	8-192	8-128

Three primary sub-procedures compose an AMR operation: adaptation flag, remeshing, and construction of a new sub-domain. Remeshing includes the parallel local grid generation and updating of the global Eulerian graph (connectivity). The construction of new sub-domains includes reordering of cells, load balancing by ParMETIS, defining the overlapping zone between partitions, data redistribution, and defining new sub-domain data/variables. We investigate the strong scalability on the last two primary procedures, remeshing and sub-domain construction. Figure 4-14 presents the overall strong scalability of remeshing and sub-domain construction. These two sub-procedures of AMR contribute comparable wall-clock time in all cases. The speedup of the AMR slows down around 64 processors and then levels off at 128 processors. The overall efficiency of AMR is 0.68 at 128 processors. The cause of slow down on remeshing is mainly due to serial operation on updating the global Eulerian graph. Other overheads in AMR come from all-to-all communication. All-to-all communication is used by the reordering cell in the sub-domain construction. When it is used to scatter large data, it causes a significant deterioration of performances, and even worse deadlock.

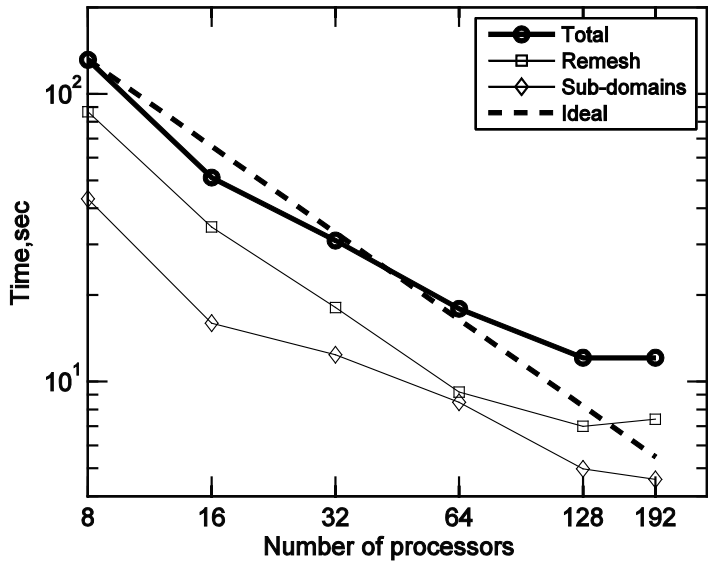


Figure 4-14. Wall-clock time of an AMR operation on strong scaling basis. The original grid size is 8.19×10^6, and 2% of grid points are refined. Remeshing and sub-domain construction scale up to 128 processors with efficiency about 0.68. All-to-all communication is the major cause of overhead in the remeshing and reordering procedures.

We study the weak scalability of the AMR by fixing the grid size per processor to 6.4×10^4. A weak scalability can provide a useful understanding on the feature of the current cell-based unstructured AMR. Especially, a program using intense far-end communication, such as all-to-all communication, should show an unfavorable weak scalability. This type of communication is adopted in updating the global Eulerian graph and reordering the cell of an entire domain. On the other hand, tasks involving near-end communication are generally free from communication overhead, and thus favorable weak scalability is expected.

In this experiment, the number of processors involved spans from 8 to 128, which has a total grid size range from 5.12×10^5 to 8.192×10^6. When doubling the number of processors used, we double the width of the computational domain in one spatial direction. We utilize ParMETIS to initialize grid decomposition, and it results in a nearly

identical grid size per partition. Every processor assigns an adaptation flag randomly on 2% of the cells in its own partition. Each experimental test is repeated five times, and an averaged wall-clock time is recorded. We calculate the time required to complete the following procedures: adaptation flag, local grid generation, update global grid, reorder cell, load balance, data redistribution, and sub-domain construction. The efficiency is calculated based on an eight-processor computation.

Overall, the efficiency of AMR is 0.99, 0.73, 0.48 and 0.26 at 16, 32, 64, and 128 processors (Figure 4-15). However, when looking at each procedure, adaptation flag and data redistribution scale to 128 processors. These two procedures have communication only with its neighbors. The efficiency of local grid generation and sub-domain construction reaches 0.5 at 128 processors. As the number of processors is increased, “update global grid” and “reorder cell” become the most inefficient procedures. Serial computation and communication overhead are primary sources of overhead in these two procedures. We use serial operations on some part of updating the global grid, and all-to-all communication to renew the global indices. Because of using the all-to-all communication, we observed that majority of communication overhead comes from the saturation of system send buffer. Due to the large amount of data and message counts, a “send” may be idle on waiting for an available system buffer. Even with the non-blocking communication routines, messages are delayed due to insufficient system buffer memory. In some cases, we can alternatively circumvent the requirement of all-to-all communication by serial computation on the global grid data in every processor, but this approach incurs serialization such that it hampers the speedup. Using user-provided buffering, such as the MPI_Bsend, may amortize this problem.

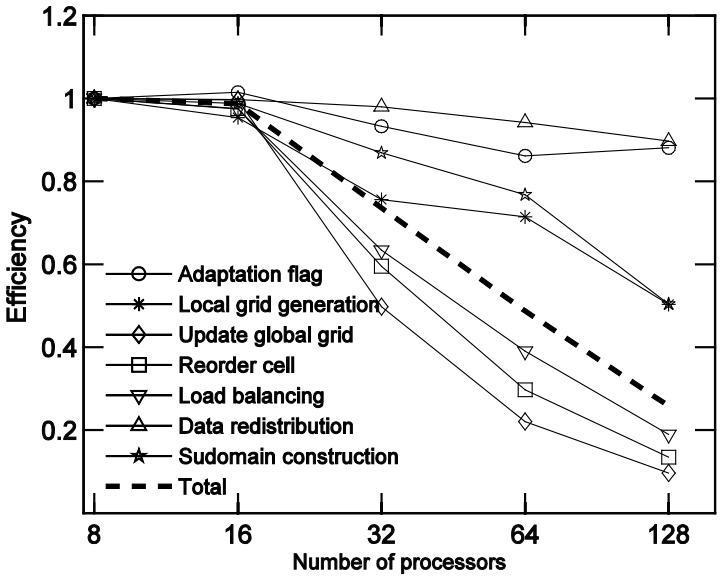


Figure 4-15. The efficiency of an AMR operation on a weak scaling basis. The total wall-clock time of one AMR operation is 2.79, 2.83, 3.85, 5.74, and 10.83 seconds for 8 to 128 processors respectively. The procedures "Adaptation flag, " "Local grid generation, " "Data redistribution, " and "Sub-domain construction" using the near-end communication pattern scale better than those counterpart procedures using all-to-all communication, such as "Updating global grid" and "Reorder cell. " Updating global grid is the major performance hurdle due to serialization and communication overhead.

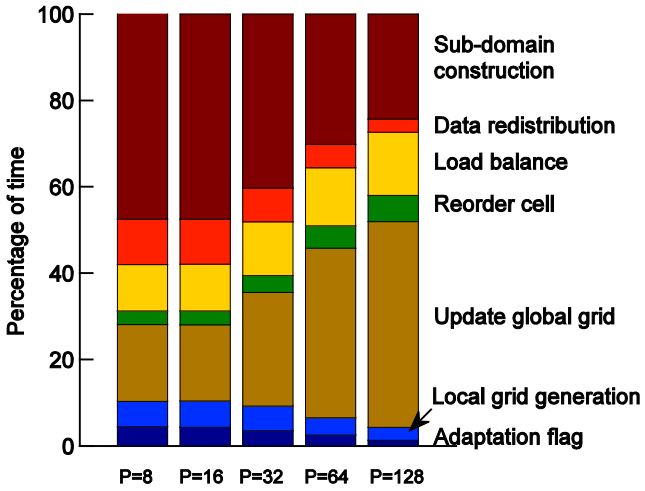


Figure 4-16. Breakdown of an AMR operation under a weak scaling basis.

4.2.3 Strong scaling on a practical problem: An off-center binary droplet collision

We use a practical example, the binary droplet collision to illustrate the overall performance of the parallel adaptive Eulerian-Lagrangian method. This investigation concentrates on the parallel performance of the AMR, interface tracking method and field equation solver. The performance is highly dependable on the effectiveness of the decomposition of the Eulerian and Lagrangian domains and communication. The computational setup is listed in Table 4-11.

Table 4-11. Conditions and parameters of the performance test using adaptive Eulerian-Lagrangian interface tracking method.

Problem definition	Off-axis binary droplet collision; Re = 302.8, We = 60, and impact factor 0.55
Resolution	$D_0/\Delta x = 128$ (six-level refinement)
Grid size	Eulerian cells: 2.2×10^6; Lagrangian marker : 3.9×10^5
# of processors used	8-192
Time recorded	Wall-clock time of 62 time-step computation including 18 times of AMR
Procedures involved	Field equation solver (Eulerian) Interface tracking (Eulerian-Lagrangian) Lagrangian mesh modification (Lagrangian) AMR

Figure 4-17 is a snapshot of interfaces with a sliced view of adaptive Cartesian grid at Weber number 60, Reynolds number 302, and impact factor 0.55. The AMR algorithm effectively refines the grid at the fluid boundaries and coarsen the grid where grid resolution is not demanding.

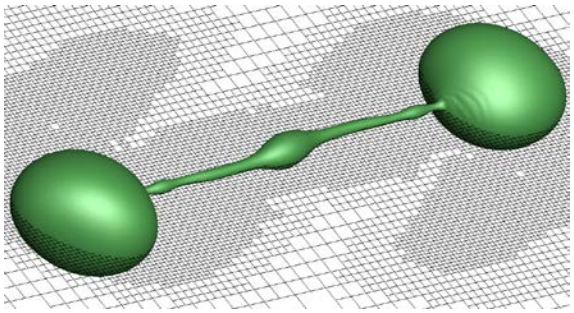


Figure 4-17. Two droplets collide eccentrically to each other at We = 60.1 and Re = 302.8 with six-levels of refinement. Initial grid size is 2.2×10^6 cells. This computation involves moving interface tracking, Lagrangian mesh modification (coarsen, smooth, and refine), interfaces reconstruction algorithm, and AMR techniques. This simulation takes about 4 days to complete a serial computation. With the parallel implementation on 32 processors, it takes about 3.5 hours.

In this experiment, the wall-clock time for the 62 time-step computation is recorded, while the AMR is applied 18 times. The AMR operation invokes geometry-based adaptation around moving Lagrangian interfaces at every 4 time-steps and solution-based adaptation at every 20 time-steps. This setup is for evaluating the performance of the AMR, but usually, we use larger spatial range of refinement to ease the need of the geometry-based adaptation or less AMR check frequency to maintain a favorable overall efficiency. We evaluate the strong scalability for an initial grid of 2.2×10^6 cells with a six-level refinement. The problem is run on 8 to 192 processors. Figure 4-18 and Figure 4-19 describe the execution time of these primary procedures. We categorize these procedures according to their computation frame: interface shape modification (Lagrangian); surface tension computation, marker movement and material determination (Eulerian-Lagrangian); Cartesian grid solver (Eulerian); and AMR. AMR shows the least speedup, which levels off after 32 processors. The reason for the low AMR efficiency is a consequence of load imbalance due to localized adaptation around the interface and the least computation-to-communication ratio. Frequent request of

AMR also increases the ratio of serial computations in the simulation. The computation time of AMR is 4% to 13 % of the total wall-clock time. Even with such frequent calling of the AMR algorithm, the field equation solver still uses the majority of the wall-clock time.

The Lagrangian procedure, which is the interface modification including smoothing, refining, and coarsening of the Lagrangian interface occupies a nearly constant portion of the total wall-clock time among all the test runs, and its speedup levels off at 128 processors. Of all the tests, every processor has a partial of the Lagrangian interface, but the load of the Lagrangian computation is not ideally balanced. This is one of the reasons for the quick slowdown of its speedup.

We observe that surface tension computation costs 20% to 23 % of the wall-clock time, while the Eulerian-Lagrangian computation uses 40% to 43 % of the wall-clock time. Overall, the Eulerian procedures scale well as we observed in the standalone test of the field equation solver, and the Eulerian-Lagrangian procedure shows a lower parallel performance than the Eulerian procedures, which is mainly due to the lower computation-to-communication ratio. The influence of load imbalance on the performance of the Eulerian-Lagrangian tasks is not as much as that of the computation-to-communication ratio since the trend of wall-clock time descends in a similar rate as the Eulerian task. The slowdown is a consequence of frequent data exchange between the Eulerian and the Lagrangian domains. This observation implies that without the decomposition of the Lagrangian interfaces based on the locality of markers, the Eulerian-Lagrangian tasks will suffer with more severe communication overhead, and have worse performances for cases using a large number of processors.

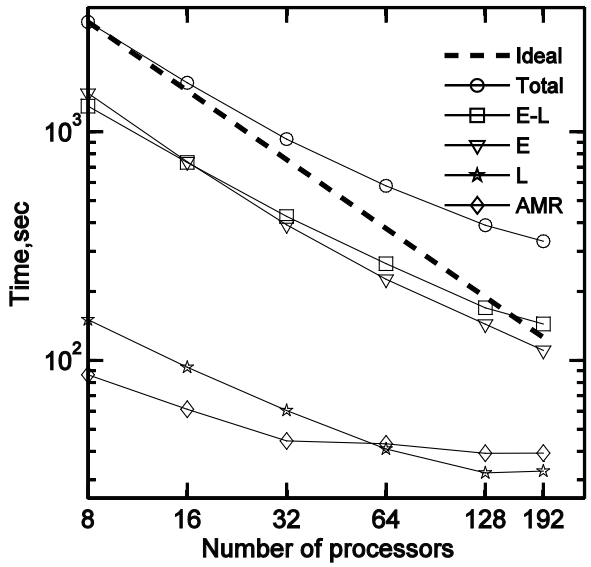


Figure 4-18. Execution time of the Eulerian-Lagrangian method for a binary droplet collision computation. The original grid size is 2.2×10^6. Overall efficiencies are 0.65 and 0.48 at 64 and 128 processors, respectively. Procedures are categorized into four task groups. Eulerian (E): the field equation solver; Eulerian-Lagrangian (E-L): the marker movement, cell material determination, and surface tension computation; Lagrangian (L): the interface shape modification; AMR: the adaptive mesh refinement.

In summary, the parallel performance of each group is proportional to its computation-to-communication ratio, in a descending order as Eulerian, Eulerian-Lagrangian, Lagrangian and AMR. This observation suggests that for a fixed-size problem, the Eulerian-Lagrangian method has a performance upper bound, which is the performance of the Eulerian tasks. Of course, the parallel performance of the Eulerian-Lagrangian method is dependent on the size and distribution of the Lagrangian interfaces in an Eulerian domain, but the overall efficiency will not exceed the efficiency of the field equation solver regardless of the parallelism adopted for the Lagrangian interfaces. An ideal load-balanced algorithm for the Lagrangian interfaces will not benefit the overall performance so much due to its low computation-to-communication ratio. In contrast, the communication due to Eulerian-Lagrangian interactions is the primary

overhead for large problems or processor counts.

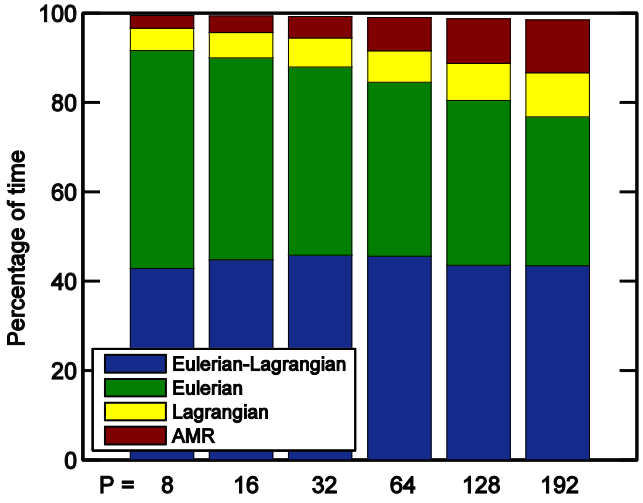


Figure 4-19. Breakdown of execution time of the Eulerian-Lagrangian method for a binary droplet collision computation. Eulerian (E): the field equation solver; Eulerian-Lagrangian (E-L): the marker movement, determination of cell material, and surface tension computation; Lagrangian (L): the interface shape modification; AMR: the adaptive mesh refinement.

Chapter 5.

Droplet collision at high Weber number regime

We are interested in the breakup of droplets and liquid jets at higher Weber number regimes to understand the competing mechanisms behind the morphology and develop more accurate atomization models. Qian and Law [92] were the first to experimentally determine the collision regimes according to the impact factor at low Weber number, as shown in Figure 5-1. The range of Weber number is below 100. Later many numerical studies were conducted to replicate droplet collisions at the same conditions of Qian and Law [8, 61, 93, 94]. Studies using VOF [94], level-set [93] and front-tracking methods [8] can successfully resolve most features of interfaces which have been shown in the experimental works.

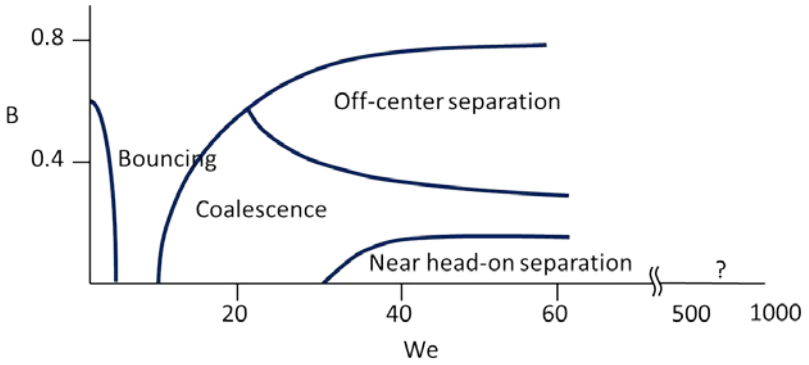


Figure 5-1. Binary droplet collision regimes as function of the impact factor and Weber number Morphologies at high Weber number regime are still unclear. Reproduced from Qian and Law [92].

Pan et al. [62] presented experimental measurements of higher speed, binary

head-on collision at Weber number ranging from 200 to 5000. The experiments show interface expansion, fingering, retreating and prompt splattering as Weber number increased. The collision outcome is similar to the droplet splashing on solid plates without the surface roughness and wetting effect of the solid phases. Due to high speed and small length scale, the experimental images provide little information for the understanding of interface breakup mechanisms. The colliding droplets experience coalescence, violent deformation, and then breakup. The size of shattered satellite droplets is dramatically reduced such that a multi-scale resolution is necessary. Three-dimensional numerical solution for high Weber number collision is also absent in the present time since considerable computational power and modeling accuracy are required. As a result, by using the parallel adaptive Eulerian-Lagrangian method, this study aims to simulate some of droplet collision cases at this Weber number regime, tries to provide insightful detail of interface instabilities, and addresses the competing mechanisms defining the primary breakup.

Table 5-1. Conditions of droplet collision including We, Re, Oh, morphology.

	We	Re	Oh	Morphology				
				Rim on the sheet	Rim detachment	Fingering	Breakup	Aggression
1	210	2982	0.0049	√				√
2	277	4686	0.0036	√		√	√	√
3	442	6207	0.0034	√		√	√	
4	688	6207	0.0042	√		√	√	
5	878	6650	0.0045	√		√	√	
6	1176	7700	0.0045	√	√		√	
7	1520	8750	0.0045	√	√		√	

Table 5-1 tabulates the dimensionless parameter based on the initial droplet size and collision velocity, and collision results. The two phase are air at room temperature

and water with surface tension coefficient 0.065 N/m, density 998 kg/m^3, viscosity 0.01 N/m^2-s. The range of Weber number is from 200 to 1500 and Reynolds number is from $2x10^3$ to $8x10^3$. The Ohnesorge number is in the order of 10^{-3}, which implies the viscous effect is a minor factor of the global collision outcome, and inertia and surface tension forces are dominant. For Oh < 10, a thin liquid sheet always has a rim attached to its edge due to the surface tension forces [95]. Free surface instability is observed on the toroid rim.

Table 5-2. Velocity, physical length, and time scale of the experiments, and numerical resource used in the computation.

We	Collision velocity	Initial diameter D_0	time scale	Averaged secondary droplet size	Baseline grid size	Max total grid size	Resolution, $D_0/\Delta x$
210	3.89 m/s	1.0 mm	0.1-1 ms	N/A	E : $2.4x10^6$ L : $2.9x10^5$	E: $2.2x10^7$ L: $2.5x10^6$	128
277	4.26 m/s	1.1 mm		523 μm			
442	5.13 m/s	1.2 mm		428 μm			
878	9.50 m/s	0.7 mm		108 μm			
1176	11.0 m/s	0.7 mm		28.7 μm	E : $5.5x10^6$ L : $6.7x10^5$		192
1520	12.5 m/s	0.7 mm		16.0 μm			192

Assessing interface evolution under these conditions is extremely challenging because of the time and length scales of collision processes. Table 5-2 shows the physical time and length scales, velocity in the experiments, and numerical resources used in the computation. Droplet diameter in the experiments is about 1 mm while the collision velocity is in the range of 4-12 m/s. The time from droplet coalescence to the primary breakup is less than one millisecond. Both experimental and numerical results show that the secondary droplet size is reduced as Weber number is increased. For Weber number 1520, the averaged secondary droplet diameter is 16 micrometer and the size distribution is 4-25 micrometer, which is close to the finest grid width in this computation (equivalent

to 3.6 μm). The physical time scale is about 0.1 millisecond from coalescence to primary breakup.

The cell-based unstructured AMR effectively manages the computational resources with moving boundaries during the computation. Figure 5-2 shows that the Eulerian grid size dynamically increases from 2.4 million to 12 million for case 3 (We = 442 and Re = 6207) as the circular disk diameter expands. In case of We 1520, the size of the initial Eulerian grid and Lagrangian markers are $5.5x10^6$ and $6.7x10^5$ respectively, and they increase to $2.2x10^7$ and $2.5x10^6$ in the end of simulation. This case uses seven-level refinement on a $20D_0x20D_0x20D_0$ domain, where D_0 is the diameter of a droplet. For equivalent spatial resolution, a simulation using a uniform grid would require approximately 56 billion cells - more than an order of magnitude larger than the largest grid with AMR. This clearly shows the advantage of adaptivity for multi-scale, moving boundary problems.

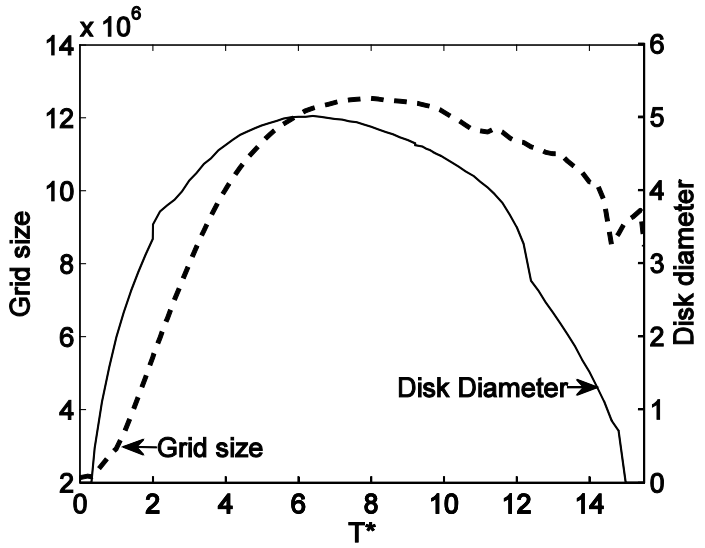


Figure 5-2. Grid size and dimensionless circular disk diameter (normalized by the initial droplet diameter) with respect to the dimensionless time $T^* = tU_0/D_0$. The growth rate of grid size follows the size of the interface.

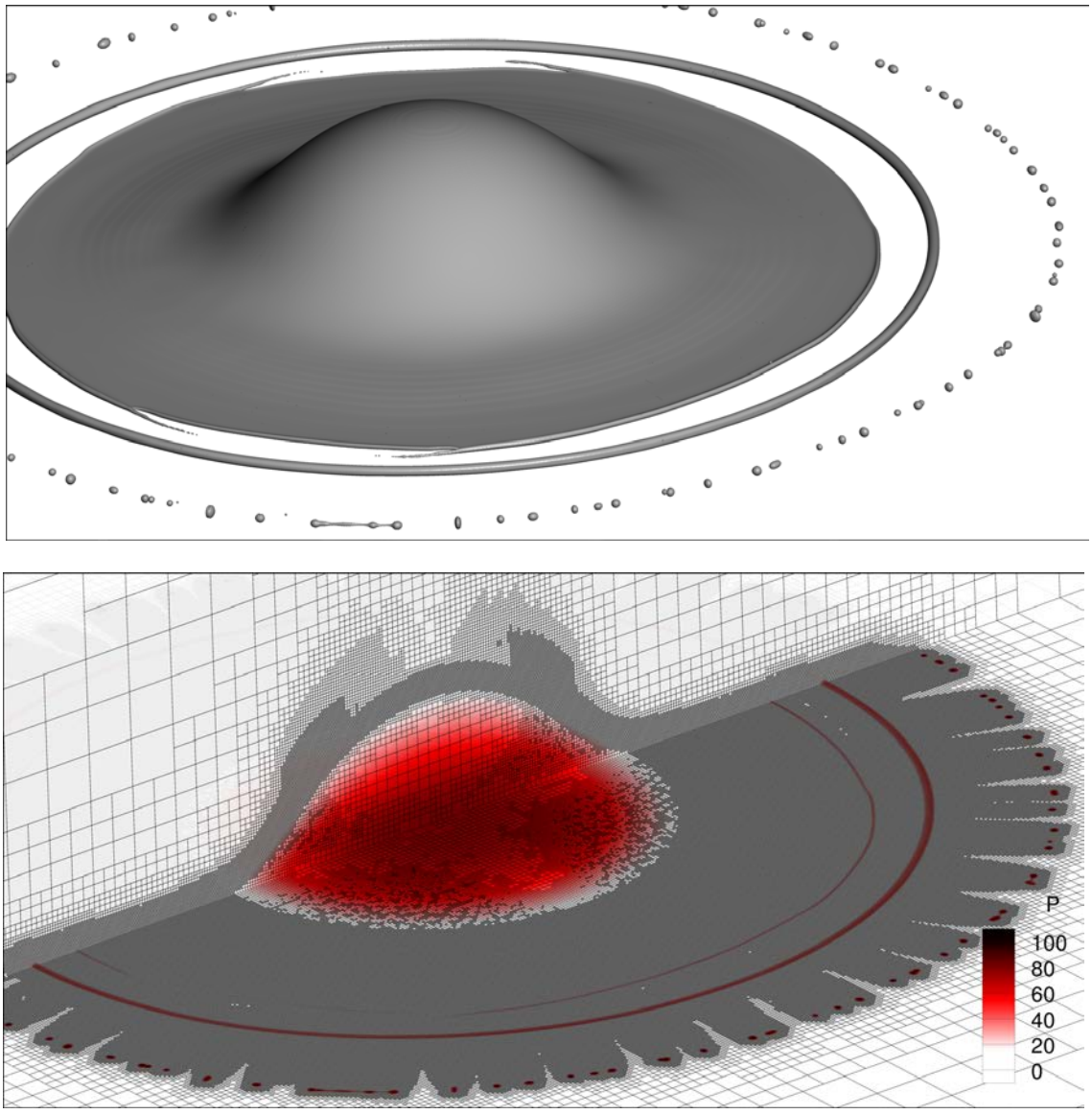


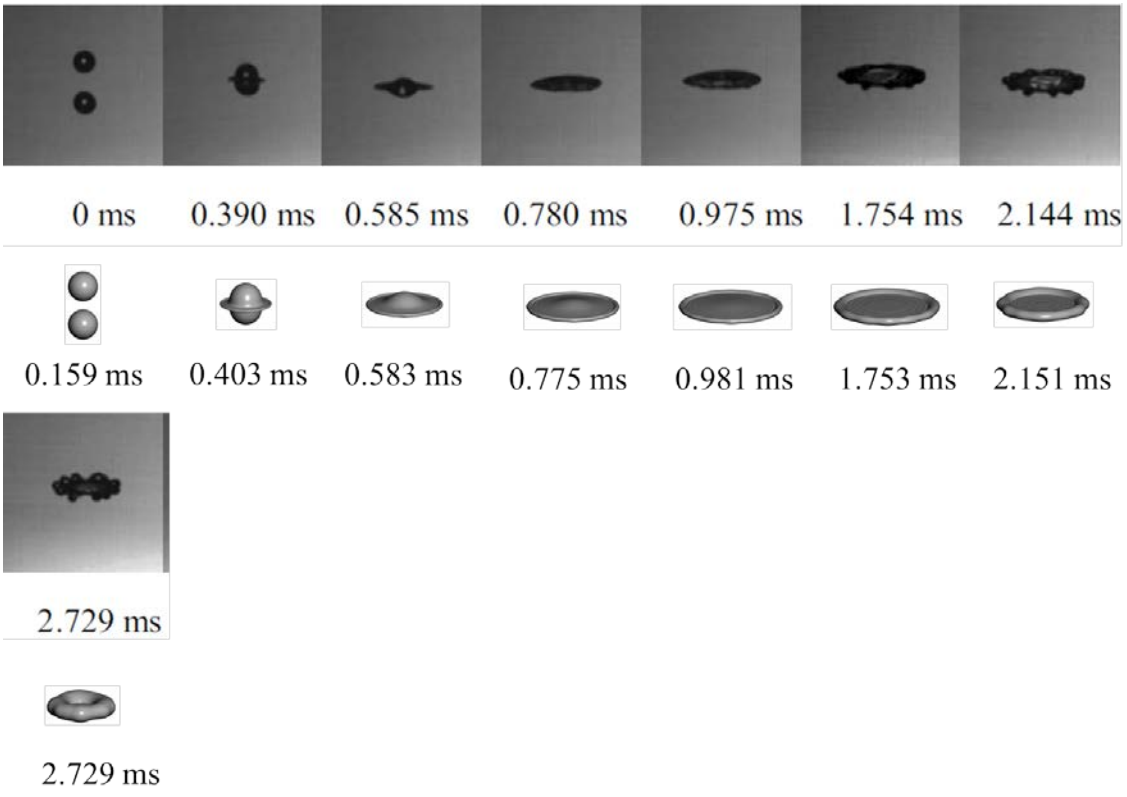
Figure 5-3. Interfaces, adaptive grid with seven-level refinement, and pressure contour on z = 0 and r-z cut-planes of case 7 (We = 1520) at T* = 0.084 ms.

Figure 5-3 (a) and (b) are the interfaces, adaptive Cartesian grid, and pressure contour on z= 0 and r-z cut-planes of the We=1520 case. This snapshot is after the breakup of the detached rim (see Appendix Case 7). The rim separation from the circular sheet and wavy structures are observed along the circumference of the rim. This rim breaks into hundreds of satellite droplets similarly to a single water jet breaking up after

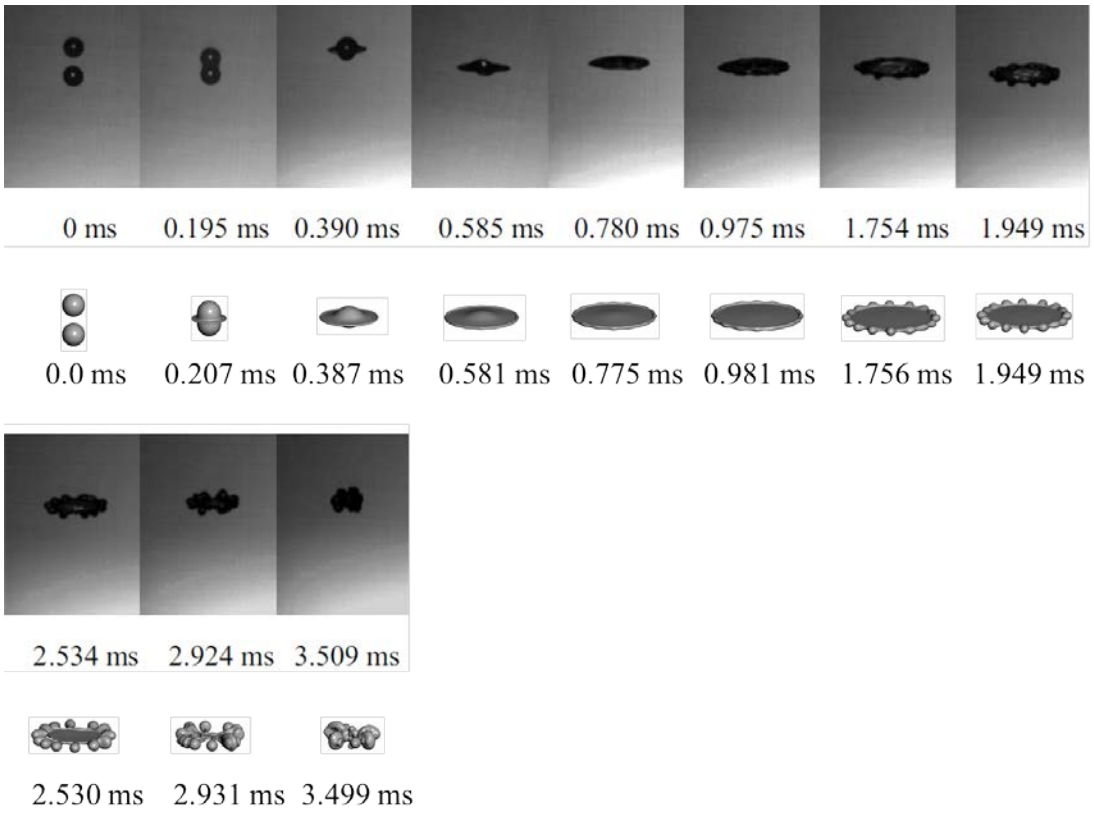
coming out of a faucet. The end-pinching effect generates another rim on the liquid sheet again. Pressure inside liquid droplets is higher than surrounding environment due to the surface tension force.

5.1 Collision history

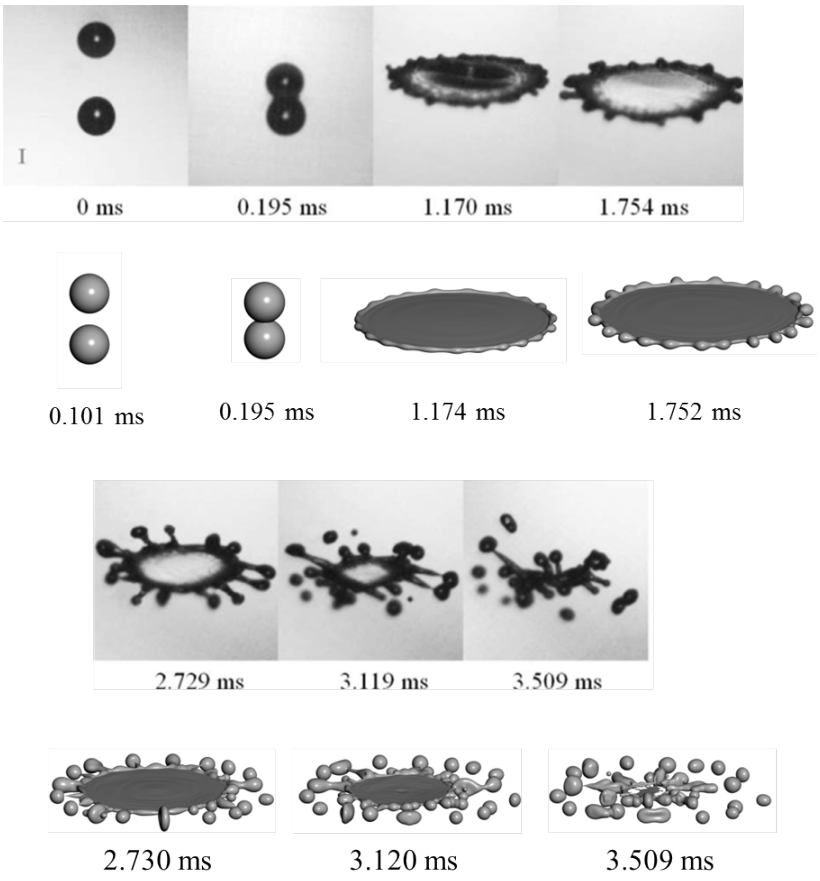
Figure 5-4 shows snapshots of the interfaces after the collision for the simulation along with experimental images. The impinging flow from both sides of two droplets extrudes a circular sheet at the waist of the merged body and expands with a growing rim at the fringe of the sheet. When the circular sheet is expanding, perturbations along the periphery of the rim (longitudinal direction) are observed. Thinning and necking effects may generate finger-like structures and further break the rim into satellite droplets. The diameter of the extruding sheet, instability pattern, size and distribution of satellite droplets are qualitatively matched with experiments. More detailed collision histories for case 1, 2, 3, 5, and 7 are available in Appendix A.



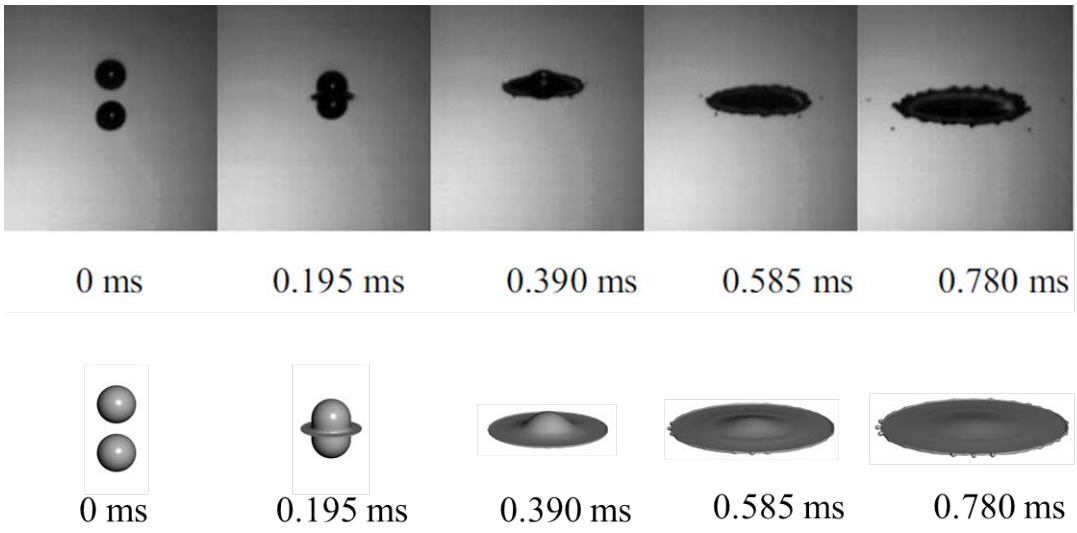
(a) Case 1: We = 210, Re = 2890



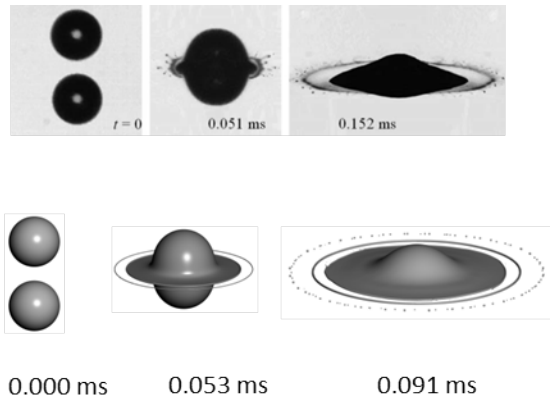
(b) Case 2: We = 277, Re = 4686



(c) Case 3: We = 442, Re = 6207



(d) Case 5: We = 878, Re = 66650



(e) Case 7: We = 1520, Re = 8750

Figure 5-4. Collision history of different conditions. Numerical works are shown underneath experimental pictures at the corresponding physical time. Experiment pictures adopted from Pan et al. [62].

5.1.1 Breakup diameter of circular sheet

Figure 5-5 shows the breakup diameter normalized by initial droplet diameter D_0 versus Weber number. The breakup diameter measures the distance from the circular sheet to the collision center when the primary (first-time) disintegration of the merged droplet happens. The result from case 7 (We = 1520) has larger breakup diameter than the experiment result since in our simulation the prompt splattering is not observed. Apart from this, the breakup diameter is consistent with experimental results. Computation of case 7 uses a grid with resolution $\Delta x = D_0/192$. A finer grid such as $\Delta x = D_0/256$ should be tested to see if the prompt splattering can be captured.

The distribution of breakup diameter of the experimental work fluctuates for Weber number less than 1200. It may be due to the difficulties to control exact head-to-head collision. The droplet diameter in experiments is 0.7mm~1mm and droplets are accelerated by high-speed impinging air with velocity 4-12 m/s. It is challenging to control head-on collision precisely under those conditions. The uncertainty of

experimental values is shown in the figure.

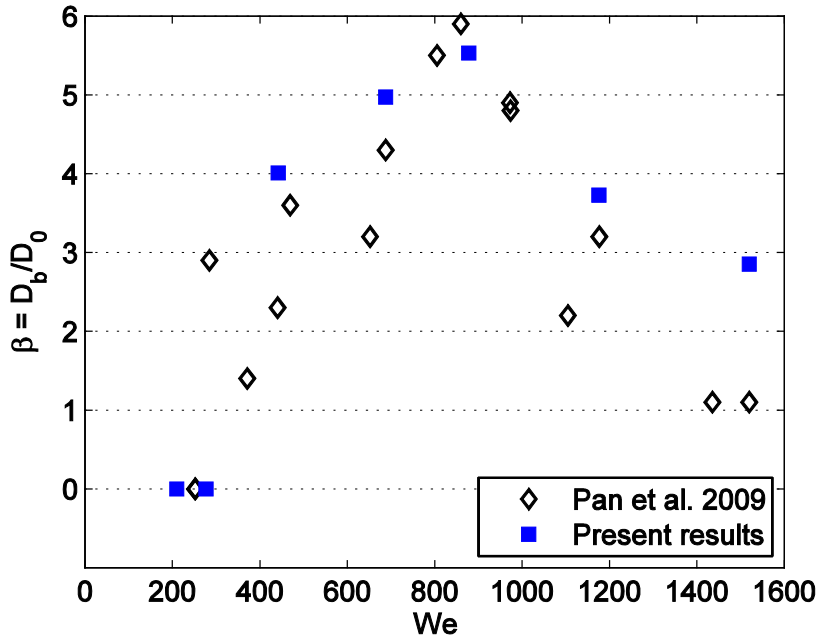


Figure 5-5. The breakup diameter versus Weber number. Experimental data reproduced from Pan et al. [62].

5.2 Observation of interface evolution

The result of the droplet collision is coalescence of the impacting droplets for all conditions at the beginning of the interface interaction. Bouncing back does not happen since surface tension is relatively small at higher Weber number regimes. As the droplets merge and continue to squeeze each other, a circular sheet is extruded like a jet. The circular sheet expands in a nearly axisymmetric manner with small perturbations on its periphery. A circular rim at the fringe of the circular sheet gradually grows due to the end-pinching effect of the surface tension force, which is so called Taylor-Culick rim (TC rim) [64, 65, 96]. Figure 5-10 shows the structure of the TC rim with the circular sheet. Disturbances along the rim are progressively amplified at the same time. For lower inertia conditions, such as We < 1000, the diameter of the rim increases as long as the circular sheet is connected with it, and thinning and necking effects happen due to the surface tension forces along the longitudinal direction shape the Taylor-Culick rim to be a nodule-like structures. This is a typical process of Rayleigh-Plateau instability [63, 97]. Furthermore, the rim and sheet may retreat at later time, and the retraction of the circular sheet together with the necking effect on the rim isolate the nodule-like structures, which is so called “fingering” (Figure 5-4(c) and Appendix Case 3). The result of fingering is that the rim breaks into liquid ligaments or smaller secondary droplets. In case of high-inertia collision, such as case 6 and 7, the rim separates from the circular sheet at very early times since surface energy force cannot hold the inertia of the rim. Retraction and the fingering effect are not observed for these two cases, and Rayleigh-Plateau instability alone determines the evolution of breakup process (Figure 5-4(e) and Appendix Case 7). A summary of the overall interface evolution for Weber number from 200 to 1500 is in Figure 5-6.

Generally, the processes of droplet collision are coalescence, deformation, and breakup. The global outline of the merged droplet linearly evolves with respect to time at the initial coalescence stage, especially before the extrusion of the circular sheet. Roisman et al. concluded that for high-enough Weber and Reynolds numbers, the flow far from the sheet edge generated by droplet collision is universal, almost independent of Weber and Reynolds number, and the characteristics of Taylor-Culick rim and circular sheet are determined by the edge effects such as end-pinching [98].

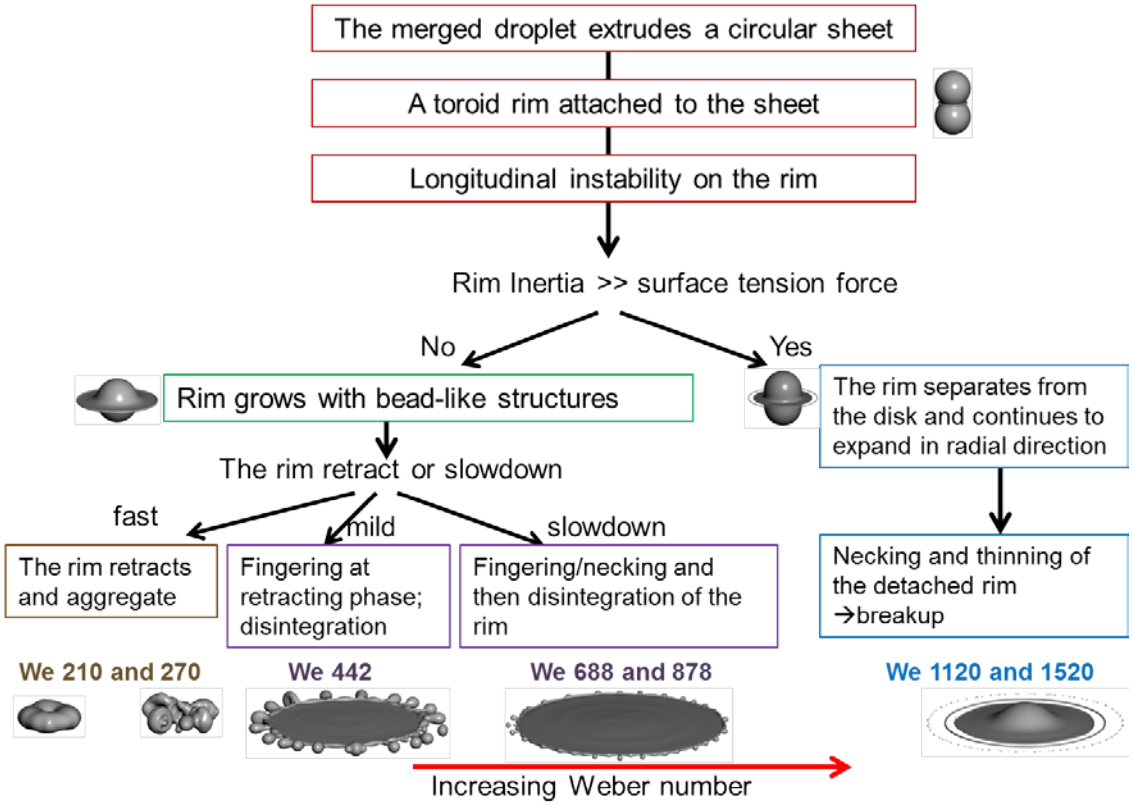


Figure 5-6. Summary of interface evolution.

Our results qualitatively agree with this conclusion. In the past, models simplifying the shape of the merged body as a pancake and using the energy conservation between surface energy and kinetic energy [62, 99] are not suitable approximations for

high Weber number droplet collision. In the high-inertia situation, the liquid sheet is very thin with a bulk rim attached, which invalidates the usage of the pancake model. The disintegration of the merged droplets is our primary interest. We will focus on the dynamics of the rim and instabilities for the determination of the size and distribution of the secondary droplets.

5.3 Nonlinearity of disturbances

The longitudinal instability on the Taylor-Culick rim is nonlinear at high Weber number regimes. There are two numerical factors affecting the instantaneous amplitude of disturbances: grid resolution and convergence error. Grid resolution effects include the discretization error and numerical accuracy of Lagrangian interfaces. The convergence error refers to residual errors of solving pressure Poisson equation, which combines effects of different linear system of equation due to different processor/node combination (the graph of linear system), round-off errors, and the usage of AMR in the computation. In this section, we first look at the effect of grid resolution on the evolution of the circular sheet and the disturbance on Taylor-Culick rim, and then exam the influence of the convergence error to the disturbances.

5.3.1 Grid dependency

The grid resolution effect is investigated by varying the finest-level grid size ($D_0/\Delta x$ = 64, 128, and 192) and fixing the processors used and convergence criterion of the pressure Poisson equation in the computation. The interface profiles of case 1 (We = 210 and Re = 3890) at the r-z cut-plane are provided in Figure 5-7. Profiles from results of different grid resolutions are superimposed on each other for showing grid dependency.

Before T* = 3.0, the interfaces from the finest and median resolution grid are nearly overlapped, but solutions from the coarsest grid show delay of circular sheet expansion. At time T* = 4.0, the amplitude of longitudinal disturbances of the finest grid are stronger than the median and coarsest grid. Note that T* = 4.0 is already at rim retraction stage. More profound delay of interface propagation and diffusion of instabilities are observed for the two coarser grid at T* = 5.0. Since the continuous interface method is used for modeling the discontinuity of phase boundaries, a finer grid gives shaper presentation of surface tension forces such that shaper surface tension force induces higher amplitude of instabilities and faster circular sheet retraction. Numerical viscosity also smears the disturbances on the TC rim. Figure 5-8 is the interface profiles on z = 0 cut-plane at T* = 1.0, 3.0, and 5.0 and wavenumbers per radian on the periphery of the circular sheet at T* = 5.0. At T* = 5.0, the disturbances at the fringe of the circular sheet from the finest-grid solution is smaller than results from the coarser grids. It is due to the fact that the circular sheet already retreats to smaller radius, which may suppress the growth of disturbances on the rim as concluded by Agbaglah et al. saying that thicker rim attenuates the instabilities [100].

The normalized power versus wavenumbers of three resolution level at T*=5.0 shows no trend of convergence, but the range of wavenumbers are consistent. One of the reasons of this inconsistency is the resolution of surface tension force. The amplification of the interface instability is driven by surface tension force, whose sharpness is determined by the grid resolution. As the surface tension force is smeared to 4-cell width across the interfaces due to the continuous interface method, the amplitude of wavenumber may evolve to a different pattern due to surface tension forces varying with

grid size. This speculation can be validated by using the sharp interface method for moving fluid boundaries with increasing resolution level to see whether the wave amplitude converges.

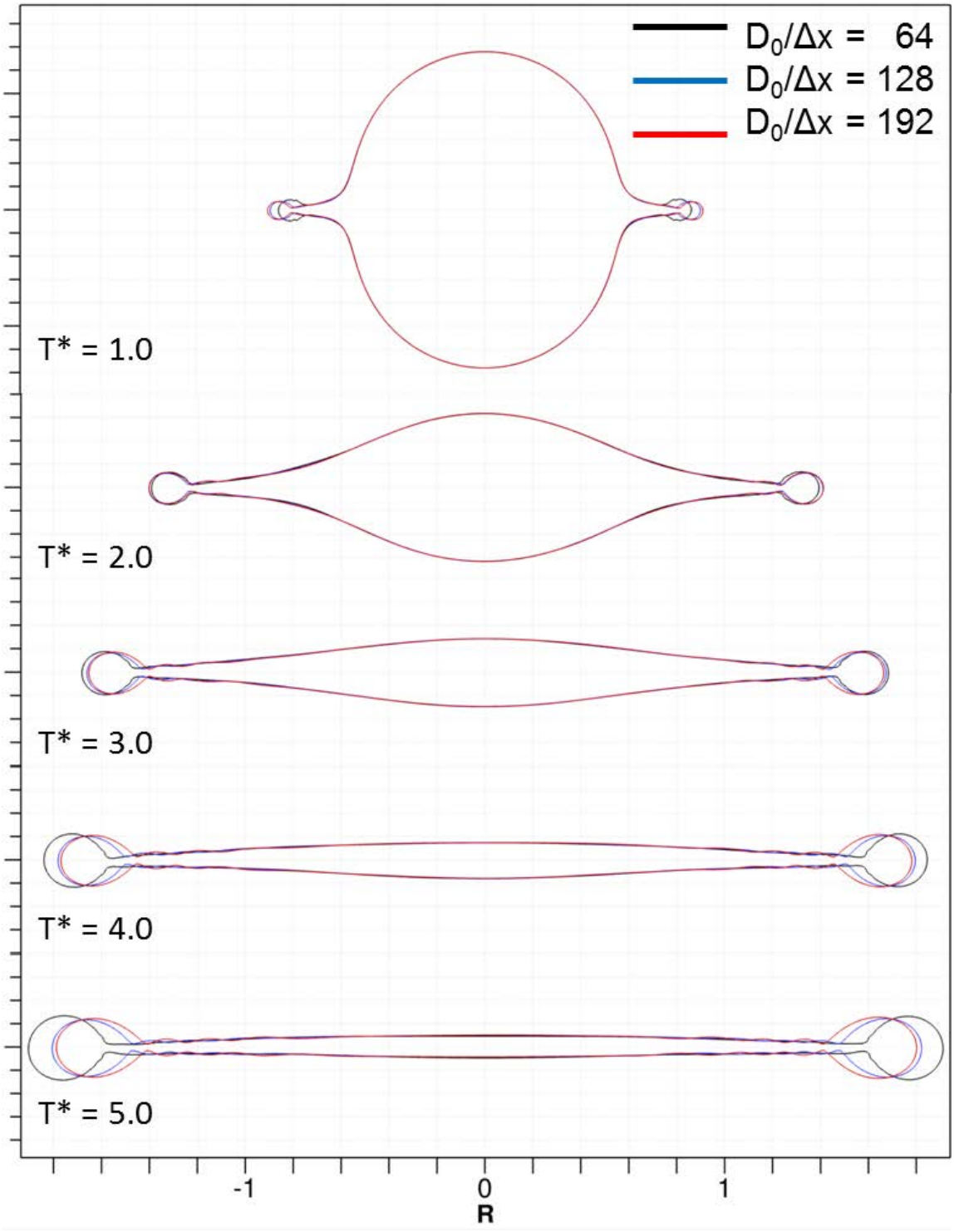


Figure 5-7. Interface profiles on r-z cut plane of case 1: We = 210 and Re = 3890. T* is dimensionless time as $T^* = tU_0/D_0$.

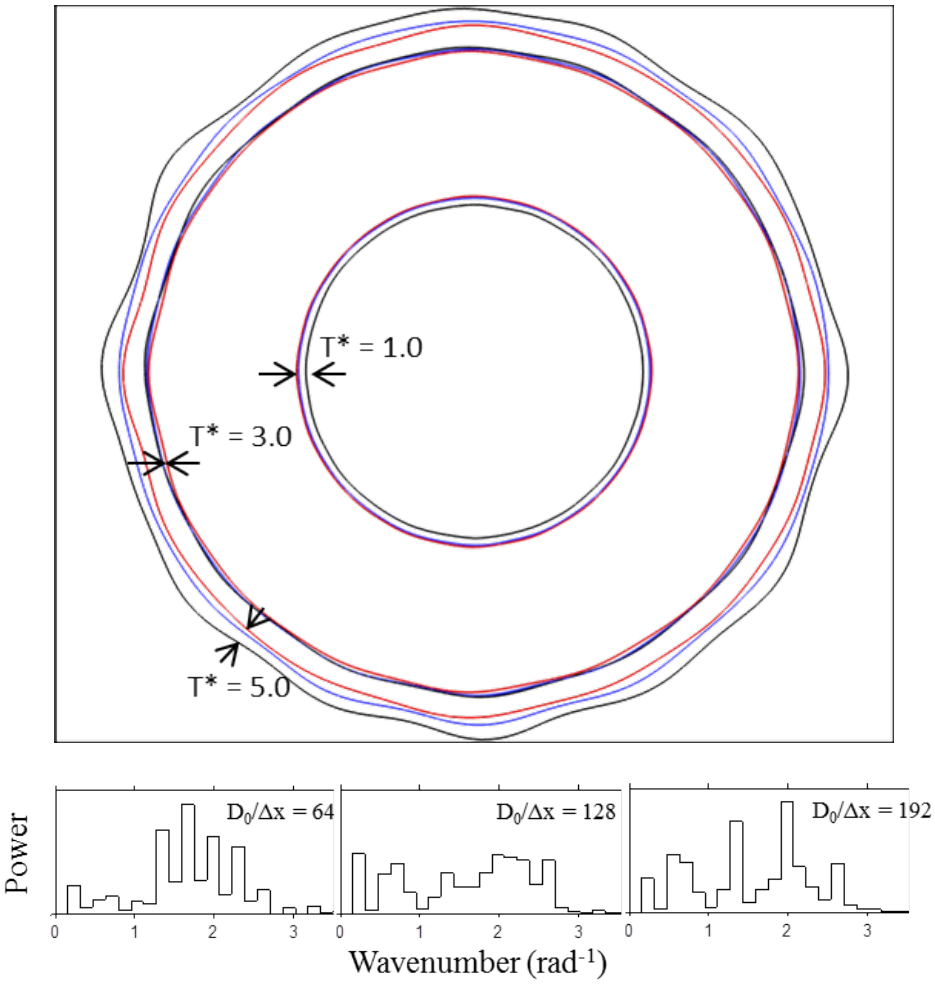
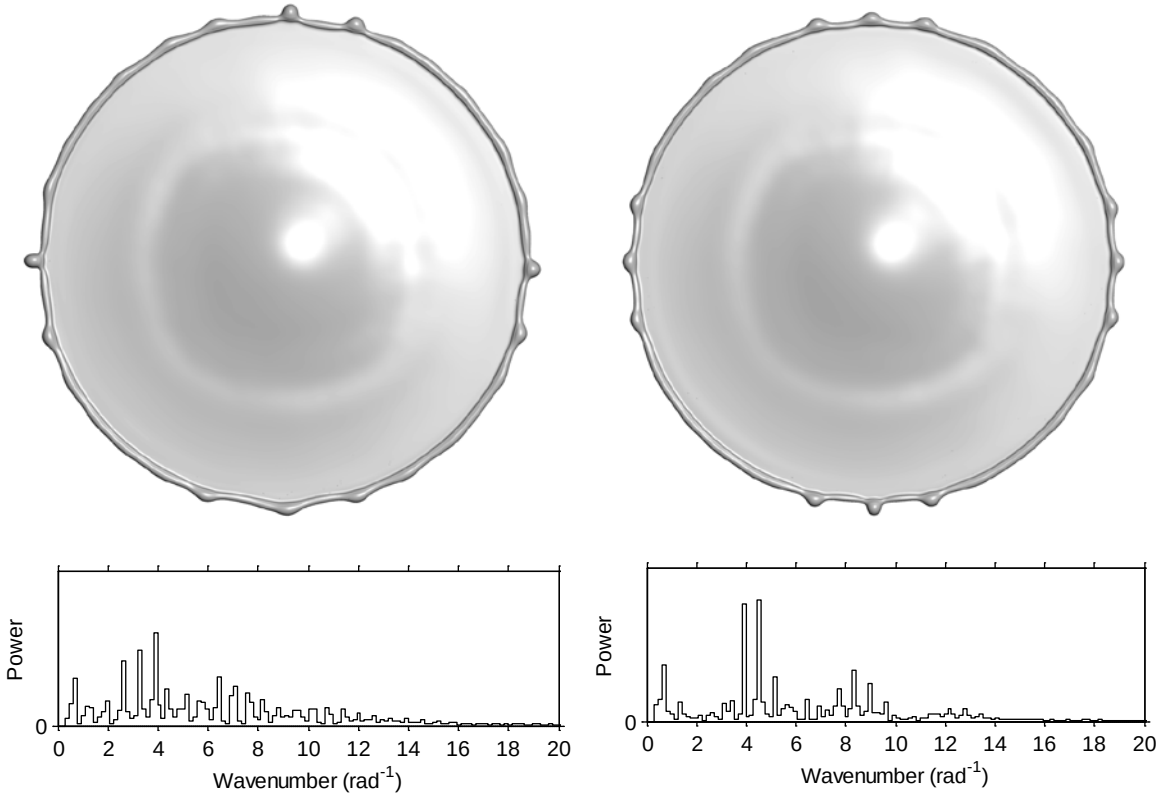


Figure 5-8. Interface profiles on z = 0 cut-plane and normalized power of wavenumbers per radian at T* = 5.0.

5.3.2 The effect of residual error

In addition to the grid resolution effect, the convergence error of the pressure Poisson equation can change from computation to computation. Varying residual of solving different linear system of equation of pressure Poisson equation results in different instantaneous disturbance pattern under the high Weber number condition. It is difficult to verify whether the AMR gird, round-off error from different processors, or different linear system for the Poisson equation is the deterministic factor to the growth

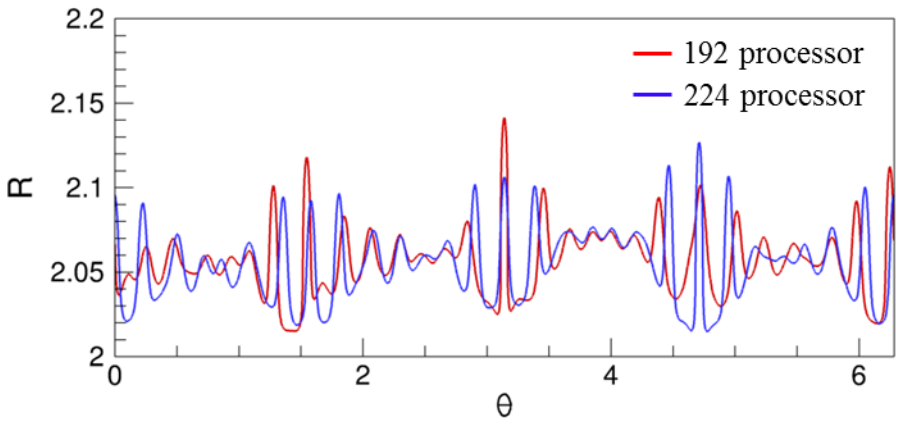
of the disturbances, but all of these factors contribute to numerical error in the iterative solver. Here we use an experiment to show how the residual errors affect the disturbance on the Taylor-Culick rim. Two problems with identical grid configuration and conditions (case 4, We 688) are solved by two computational setups, a 192-processor and a 224-processor run. The convergence criterion of the linear solver is 10^{-6}. Disparities of the disturbance on Taylor-Culick rim are shown in Figure 5-9(a). At T* = 2.4 (the 2063rd time step), the ranges of spectrum are similar, but the power spectrum of 192-processor computation is more even than that of the 224-processor computation. This example shows that the instantaneous distribution of the longitudinal instability is sensitive to the residual error at the evolving phase. In addition, the maximum amplitude of the instabilities from two computational setups has similar magnitude as shown in Figure 5-9(b). This observation suggests that the residual error influences the spatial distribution of the disturbances, but the overall instability growth rate in two computational setups is similar. Therefore, the computational setup is not influential to the timing of the primary breakup (or the breakup diameter of Figure 5-5). In summary, although the computational setup of a simulation affects the spatial distribution of the disturbances, the current numerical framework can reasonably resolve the dynamics of the nonlinear disturbances under the high Weber number conditions.



192 processor

224 processor

(a)



(b)

Figure 5-9. Disturbances on the Taylor-Culick rim for case 4, We 688: (a) interfaces and spectra (b) amplitude of disturbance represented by the radius from the circular sheet fringe to the collision center of 192-processor and 224-processor computation at T* = 2.4.

5.4 Droplet disintegration

5.4.1 Dimensionless analysis

The variables in the following context are normalized by the diameter of the original droplet D_0, collision velocity U_0, and the time-scale of $\tau_0 = D_0 / U_0$. Inertia and surface tension are the main competing forces on the dynamics of the TC rim and the circular sheet. Ohnesorge number of these cases based on global length scale D_0 is about $O(10^{-3})$ (shown in Table 5-1). Viscous resistance is relative insignificant with respect to the surface tension forces globally. Other variables used in the following context are the rim velocity U_{rim} and diameter D_{rim}, sheet thickness h and velocity U_{sheet} at the neck of liquid sheet (indicated in Figure 5-10), the distance of TC rim to the droplet center R_{rim}, and maximum rim distance $R_{\max}$, which are all normalized by D_0 and U_0, as shown in Table 5-3. These dimensionless variables used are illustrated in Figure 5-11 to Figure 5-19.

Table 5-3. Relevant dimensionless parameters used in the analysis of binary droplet collision flows.

parameter	Definition
$D_0 = 1$	Initial droplet diameter
$U_0 = 1$	Initial collision velocity
$\tau_0 = D_0 / U_0$	Time scale of collision
$D^*_{rim} = D_{rim} / D_0$	Diameter of the Taylor-Culick rim
$h^* = h / D_0$	Sheet thickness at D_{rim} away from the inner edge of the rim
$U^*_{sheet} = U_{sheet} / U_0$	Sheet velocity at D_{rim} away from the inner edge of the rim
$U^*_{rim} = U_{rim} / U_0$	Velocity at the center of the Taylor-Culick rim
$U^*_{TC} = U_{TC} / U_0$	Normalized Taylor-Culick velocity on the TC rim
$R^*_{rim} = R_{rim} / D_0$	Rim distance, from the droplet center to the center of the rim

$R^*_{max} = R_{\max} / D_0$	Maximum distance from the collision center to the rim

A characteristic velocity, called Taylor-Culick velocity in Eq. (30) is the asymptotic, retreating velocity of a free liquid edge in the long time limit [65], and independent of viscosity [83].

$$U_{TC} = \sqrt{2\sigma / \rho h} \tag{30}$$

In our cases, the normalized Taylor-Culick velocity, $U^*_{TC} = U_{TC} / U_0$, based the circular sheet thickness at the neck of the rim, is around 0.5 to 0.6. We measure the circular sheet thickness at the neck of the Taylor-Culick rim and found that it is nearly constant at the circular sheet expansion phase.

The other characteristic time scale τ_c is liquid capillary time scale based on the sheet thickness *h* and Taylor-Culick velocity, as shown in Eq. (31).

$$\tau_c = \sqrt{\rho h^3 / \sigma} \tag{31}$$

This capillary time scale ranges from 0.01 to 0.04. It is relatively small compared with τ_0. In the interface profile plot of Figure 5-7 (T*=5.0), we can observed the capillary wave disturbance on the circular sheet. Capillary time scale of the current system implies that the capillary wave propagates from one end to the other end of the interfaces in a very short time (much faster than the collision process).

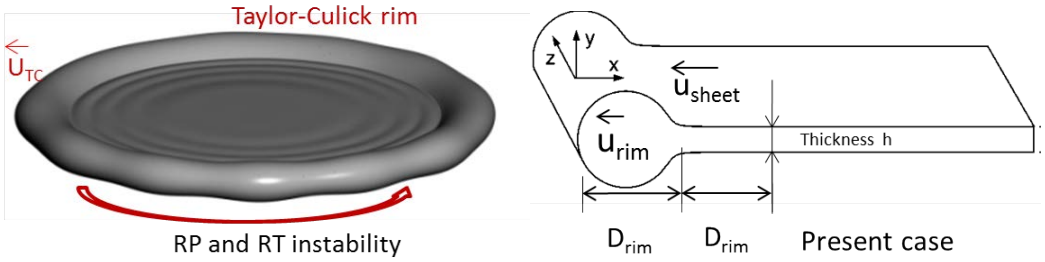


Figure 5-10. Movements and instabilities on the Taylor-Culick rim and the circular sheet.

5.4.2 Dynamics of the Taylor-Culick rim and the circular sheet

The flow field far away from the edge of the extruding sheet is universal, but the rim diameter, distances from the original droplet center, and sheet thickness are dependent on Weber and Reynolds numbers. This observation is consistent with the conclusion by a similarity approach of Roisman et al. [98]. Figure 5-11 to Figure 5-13 show the time history of variables for case 3 (We 442), 5 (We 878), and 6 (We 1176). These cases present typical phenomena including the TC rim expansion and retraction (Figure 5-11 and Figure 5-12) and detachment of the rim from the circular sheet with instabilities resulting in primary breakup (Figure 5-13). In case 3, the TC rim and the circular sheet experience expansion and retraction while the corrugations on the rim are onset. Case 5 shows the amplification of the disturbances on the TC rim and breakup of TC the rim before the retraction phase. Case 6 has the TC rim separated from the circular sheet at $T^* = 0.77$ due to high inertia.

During the collision, the circular sheet is continually thinning, expanding with decreasing velocity. As shown in Figure 5-11 and Figure 5-12, the circular sheet approaches a nearly constant thickness at its neck. This interesting observation is

analogous to a thinning sheet of a droplet colliding upon a solid wall. Egger et al. [101] stated that a droplet impacting on solid surface stops thinning when the free surface meet the boundary layer. The thickness of the thin film is proportional to $\mathrm{Re}^{-2/5}$ as boundary layer thickness.

$$h_f \approx D_0 / \mathrm{Re}^{2/5} \tag{32}$$

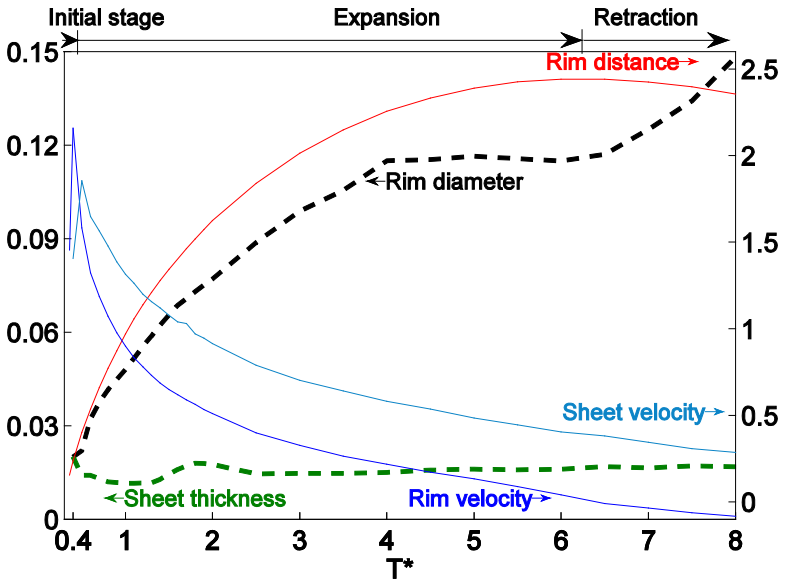
Note that h_f represents the thinnest thickness of an impacting droplet on the solid wall. In case of two droplets impinging each other, there is no boundary layer to be the thickness limit of liquid sheet. In Figure 5-11 and Figure 5-12, the circular sheet reaches a nearly constant thickness at the neck of the rim right after the spreading of the circular sheet. There seems to be an equilibrium status around the connecting point between the liquid sheet and the TC rim. It can be a consequence of force balance between the surface tension force from end pinching effect and momentum transportation via the viscous force. When the sheet thickness *h* at the neck (definition of *h* is illustrated in Figure 5-10) is plotted against We·Re, an inverse linear relationship is shown in Figure 5-14. A linear fit is provided as

$$h \propto (\mathrm{We}\,\mathrm{Re})^{-1.012} \tag{33}.$$

We don't know the exact reasons behind this observation yet. Further investigation is needed to explain this relationship.

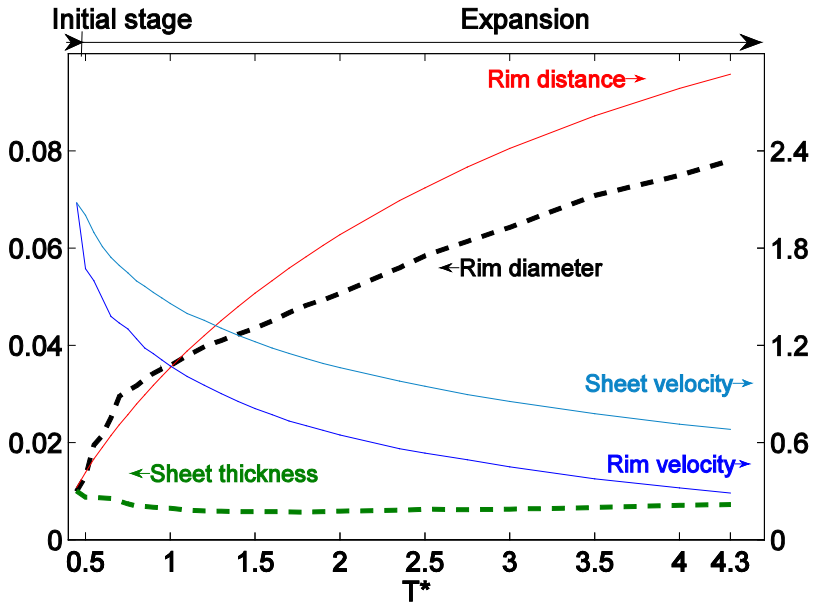
In Figure 5-11 and Figure 5-12, the velocity difference between the TC rim and the circular sheet approaches a constant value. If the velocity of the rim is observed on the frame of the circular sheet, nearly constant speed of the rim is observed. This is similar to the model of a static liquid film retraction from Culick's analysis. We may

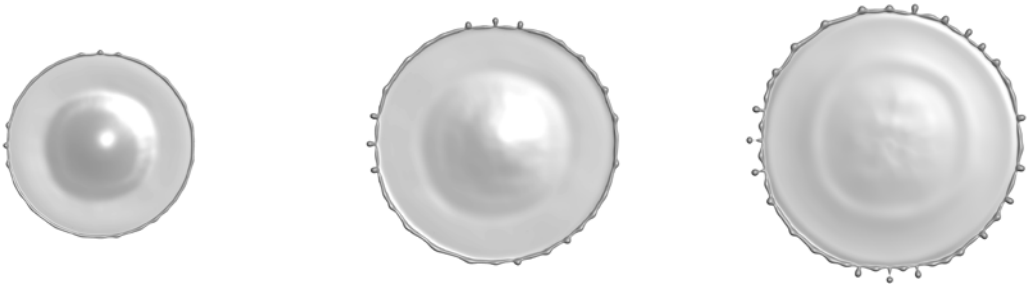
compare the velocity difference with the Taylor-Culick velocity defined in Eq. (30). Figure 5-15 is the relative velocity of the TC rim and the liquid sheet together with Taylor-Culick velocity based on the thickness of the sheet of case 3. The TC velocity are 0.53 based on sheet thickness 0.016 in case 3 (Figure 5-11). The deviation between the rim-to-sheet velocity and the Taylor-Culick velocity is about 0.2. Discrepancy may come from incompatibilities of the assumptions. The sheet thickness is a function of space and time in the computational results and is inconsistent with the assumption of static, constant and infinite flat sheet as in the analysis of Culick [65].



Expansion phase at T* = 4.0 Onset of retraction at T* = 6.0 Retraction phase at T* = 8.0

Figure 5-11. Distance, thickness, and velocity of the rim and the sheet of case 3. Three representative snapshots at T*=4.0, 6.0, and 8.0 show the location of TC rim and longitudinal instabilities on the rim.





Expansion phase at T* = 2.3 Expansion phase at T* = 3.3 Onset of retraction at T* = 4.3

Figure 5-12. Distance, thickness, and velocity of the rim and the sheet of case 5. Snapshots of the expansion phase at T*=2.3, 3.3 and the onset of retraction at T* = 4.3 are shown.

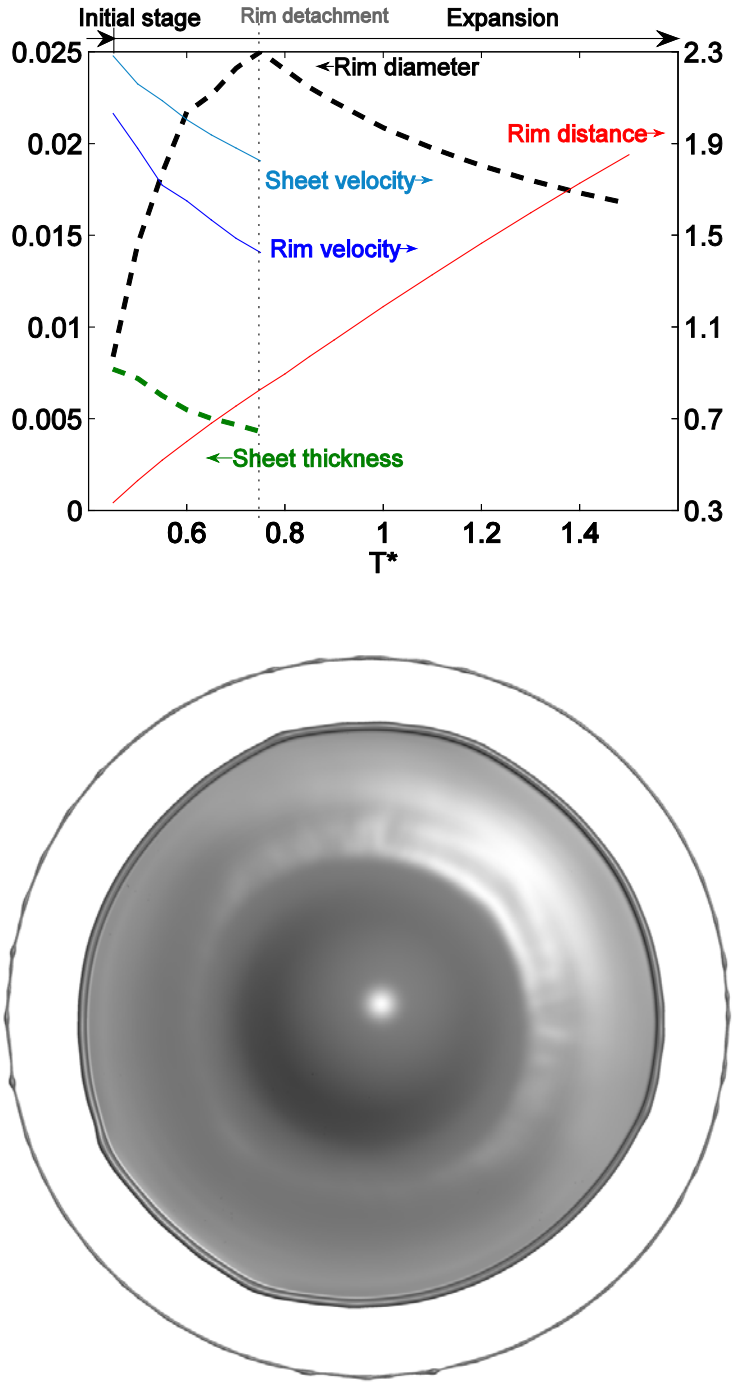


Figure 5-13. Distance, thickness, and velocity of the rim and the sheet of case 6. A top view of the circular sheet, TC rim, and detached rim at T* = 1.5 is shown.

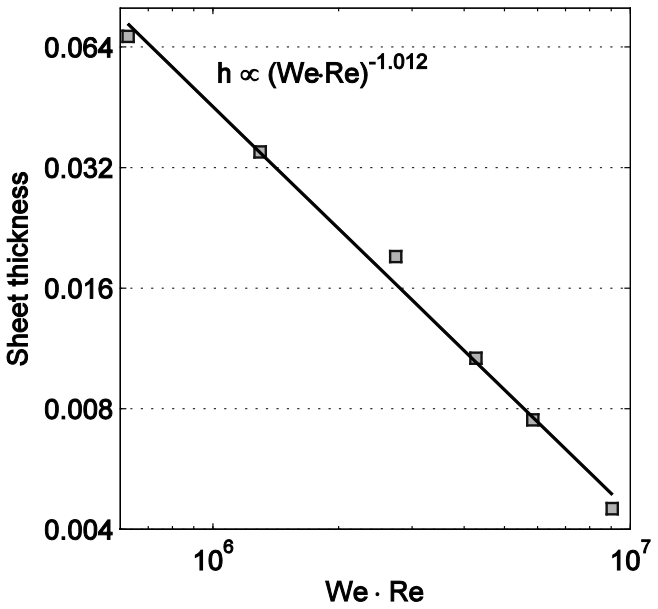


Figure 5-14. Sheet thickness as a function of We*Re.

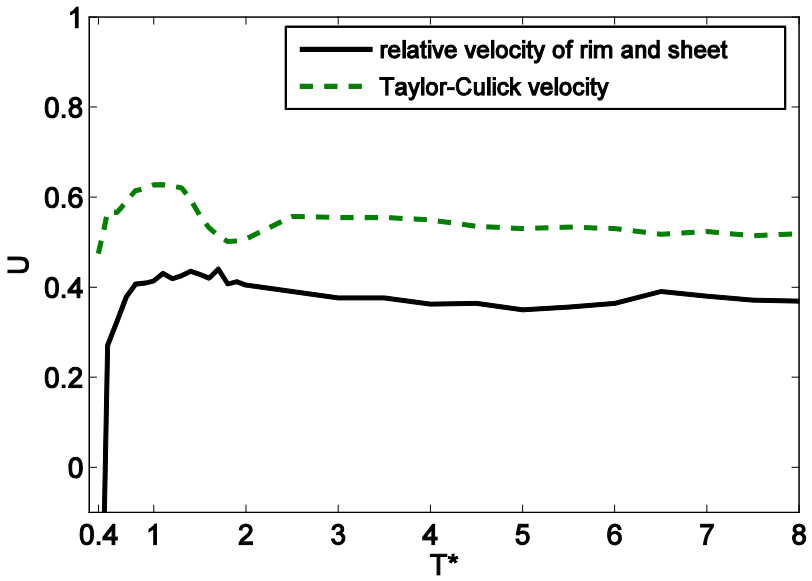


Figure 5-15. Velocity difference between the TC rim and the circular sheet and Taylor-Culick velocity estimated by instantaneous sheet thickness of case 3.

The TC rim is subject of three phenomena. First, the TC rim expands for a while, and then retracts back by surface tension forces if the rim still attached to the circular sheet. Second, the diameter of the TC rim grows as it propagates. The increasing size of the TC rim is related to the velocity difference at the center of the rim and the sheet velocity. The TC rim moves slower than the sheet that results in liquid accumulation in the TC rim. Third, there is longitudinal instability on the TC rim. The evolution of the longitudinal (along-the-edge) disturbances can be anti-symmetric perturbations as Rayleigh-Taylor (RT) instability and symmetric perturbations governed by the Rayleigh-Plateau (RP) instability [63, 100].

Acceleration of the TC rim results in RT instability. As a result, the origin and amplification of RT instability can be qualitatively argued by checking the magnitude of acceleration at the front edge of the circular sheet. A head-on droplet collision generates a radial jet as the redirection of impinging flow at the initial stage. This axi-symmetric jet then rapidly decelerates due to volume expansion and surface tension forces (see velocity profile of liquid sheet for $T^* < 0.5$ in Figure 5-11 to Figure 5-13). In the expanding and retracting phases of the circular sheet, the deceleration is proportional to the surface tension force, which is relatively small compared with liquid inertia under the high Weber number regimes. As a result, the magnitude of acceleration of the rim is strong only in the initial stage of collision, which suggests that RT instability may only have profound impact on the longitudinal disturbances at the early stage of the circular sheet development.

The Rayleigh-Plateau instability describes the instability of free liquid stream, film, or thread. Free surface can wiggle and then defragment as smaller ligaments by

surface tension forces. Rayleigh used a linear stability analysis with periodic, initial perturbation for an infinite liquid thread to examine the most unstable modes of disturbances [63]. Applicability of the linear analysis to describe the instability on TC rim of the present cases is questionable. In Rayleigh's analysis, for a uniform circular liquid thread with diameter d, the possible wavelength that destabilizes the entire system is $\lambda > \pi d$, and the fastest-growing wave length is $\lambda_{max} = 4.51d$.

In literature regarding the instabilities on free surfaces, Krechetnikov provided quantitative information of wavenumber and growth rate on inviscid liquid sheets by theoretical models [102]. He concluded that RT and RP instabilities are interacting components of the instability, and cannot be considered as linear superposition. In addition, the attached circular sheet may affect the amplitude of the perturbation. The morphology on the Taylor-Culick rim has similarities as the instability of crown splash from droplet impingement on a liquid film. Zhang et al. studied the instability of splashing crown generated by falling droplets on liquid films [97]. A conclusion from Zhang et al. is that the peak wavenumber at any given time is determined by not only the instantaneously most rapid growing mode but also by the history of the other modes. They found the amplification of the dominant modes is consistent with the prediction of linear stability (Rayleigh-Plateau), and maximum number of secondary droplets can be determined by the most unstable wavelength. This summary suggests that we may determine the secondary droplet sizes by the diameter of the TC rim and the dominant wave number. More recent work by Agbaglah et al. [100] used a long wavelength approximation model to analyze the transverse instability of a rim on a receding sheet. They concluded RT instability is due to retracting velocity and RP instability takes over

the instability when the rim radius is larger with respect to the sheet thickness. The growth rate of RP instability is $\omega_{RP} \sim k\sqrt{2}$, and RT instability is $\omega_{RT} \sim (-2\dot{v}/\pi)^{1/4}\sqrt{k}$, where k is the wavenumber and $\dot{v}$ is the acceleration. Our cases show the deceleration at the retracing phase of liquid sheet is much smaller than the deceleration at the beginning of the sheet-extruding phase while the periodic wavy structures on the edge is not observed yet. Therefore, we can preliminarily conclude that the RT instability is in charge of small disturbance only at the initial colliding stage, otherwise RP instability decide the wavenumber on the rim at the moment of breakup. Another observation from Agbaglah et al. is that the rim diameter grows almost linearly in time after a transient period, which agrees with the results showing in Figure 5-12. Furthermore, the growth rate of instability on the TC rim increases when the acceleration term is negative. It means fingering effect promotes the disturbance, and instability is attenuated by the forward acceleration. As a result, we observe the disintegration of a rim happens mostly around the onset of rim retraction. The last relevant remark from the work of Agbaglah et al. is that wave with the highest growth rate is not affected by the aspect ratio of the rim diameter and the sheet thickness. In the present numerical results, sheet thickness of case 3 (Figure 5-11) and case 5 (Figure 5-12) are almost constant in time while the diameter the TC rim is growing, and the most profound wavenumbers on the TC rim of these cases are almost fixed as shown in Figure 5-16 and Figure 5-17. This observation is consistent with the conclusion of Agbaglah et al.. The diameter of the Taylor-Culick is less relevant to the wavenumber, but the sheet thickness can be correlated with the final wavenumber on the rim and the size of secondary droplets.

5.4.3 Spectra on the Taylor-Culick rim

The wavenumber on the TC rim of Case 3, 5, and 6 are presented in Figure 5-16, to Figure 5-18 respectively. These results provide an insight of how the wave number of perturbations on the TC rim evolves with the dynamics of the TC rim. The spectra on the edge of the TC rim are calculated by Fourier transform on the distance from the rim edge to the collision center for 360 discretized points. The wavenumber per radian is plotted with normalized power amplitudes at dimensionless time T*. Red text denotes the corresponding wavenumber per radian of the fastest-growing wavelength λ_{max} predicted by the linear stability analysis of Rayleigh and instantaneous averaged rim diameter $\bar{D}_{rim}$.

$$k = R_{rim} / \lambda_{\max} = R_{rim} / 4.51\bar{D}_{rim} \tag{34}$$

Figure 5-16 to Figure 5-18 show discrepancy between the most profound wavenumber at different time instances and fastest-growth wavenumber predicted based on Rayleigh' theoretical analysis. The largest wave does not match with the fastest-growth wave predicted by Rayleigh's analysis since the spectra on the rim are continuous evolution of varying wavenumbers and initial conditions. In addition, theoretical work of Rayleigh considers a uniform perturbed circular thread, which is not the case of realistic problems present here. We have argued that RT instability determines the initial disturbances at the initial stage of the sheet expansion. Furthermore, expansion of the TC rim may affect the growth rate of perturbation. The expanding sheet attached to the TC rim also reduces the amplitude of instability [100]. However, a comparison between current results and the theoretical work of Rayleigh can verify whether the current numerical method resolves the trend of RP instability in a reasonable way.

In case 3, the TC rim is growing at T* = 2.0, and the amplitude of the

disturbances is relatively small compared to the rim distance. Between T* = 4.0 and 6.0, the diameter of the TC rim is nearly constant since it is the onset of retracing. The largest wave is at k = 3.8 Hz/rad, which is smaller than the fastest-growth wave k = 4.3 Hz/rad from theoretical prediction. At T* = 8.0, the rim is receding, and two peak at k = 2.5Hz/rad and 3.8Hz/rad is presented with a wider spectrum.

In case 5, the primary breakup happens before the rim recedes. The profound waves gradually shift from low wavenumber to higher wavenumber. This phenomenon implies wavenumbers around k = 7 to 10 Hz/rad are the fastest-growth group amplified by RP instability. Again, the spreading spectrum is observed as the rim expands. Top views of interfaces show more structures are generated in between old nodule-like structures as the rim and liquid sheet start to recede at T* = 4.3. This observation suggests that the fingering effect enhanced by the receding of rim produces more nonlinearity. These new structures will be the dominant structures as the old nodule-like structures detach from the TC rim.

In case 6, the rim detaches from the sheet at T* = 0.77. After T* = 0.77, the rim is a constant volume toroid. There is no impact from the sheet. We can obtain the most unstable wave k = 8.76 Hz/rad based on the rim diameter at T* = 0.77. A profound and persistent wave at k = 7.0 Hz/rad are observed all the time on the TC rim, which is close to the theoretical, fast-growth wave at k = 8.76 Hz/rad. A group of diminishing waves is at the low wavenumber regime.

The evolution of wavenumbers on the TC rim is summarized in Table 5-4. As the TC rim connects with the circular sheet, the peak wavenumbers slightly shift to higher wavenumbers at the expansion phase, and migrate to smaller wavenumbers at the

retraction phase. The largest amplitude wavenumber on the TC rim is determined at the early stage of the expansion phase, and does not change very much with time. It implies the elongation of the TC rim is not influential on the growth rate of each wavenumber. The rim retraction enhances the aggregation of nearby perturbations such that the peak waves become smaller. In case of a detached rim, the migration to higher peak wavenumber is more obvious than that of an attached rim. We may conclude that the circular sheet attenuates the growth rate of RP instability. For all conditions, the range of spectrum broadens with time. The widening spectra exhibit the fact that longer time allows the system to grow more nonlinearity.

Table 5-4. Summary of the wavenumber evolution on the Taylor-Culick rim.

		Rim growth	Peak wavenumber	Range of spectrum
Attached rim	Expansion phase	Yes	Slightly shift to higher wavenumbers	Widening
	Retraction phase	Yes	Shift to smaller wavenumber	Widening
Detached rim	Expansion phase	No	Shift to higher wavenumbers	Widening

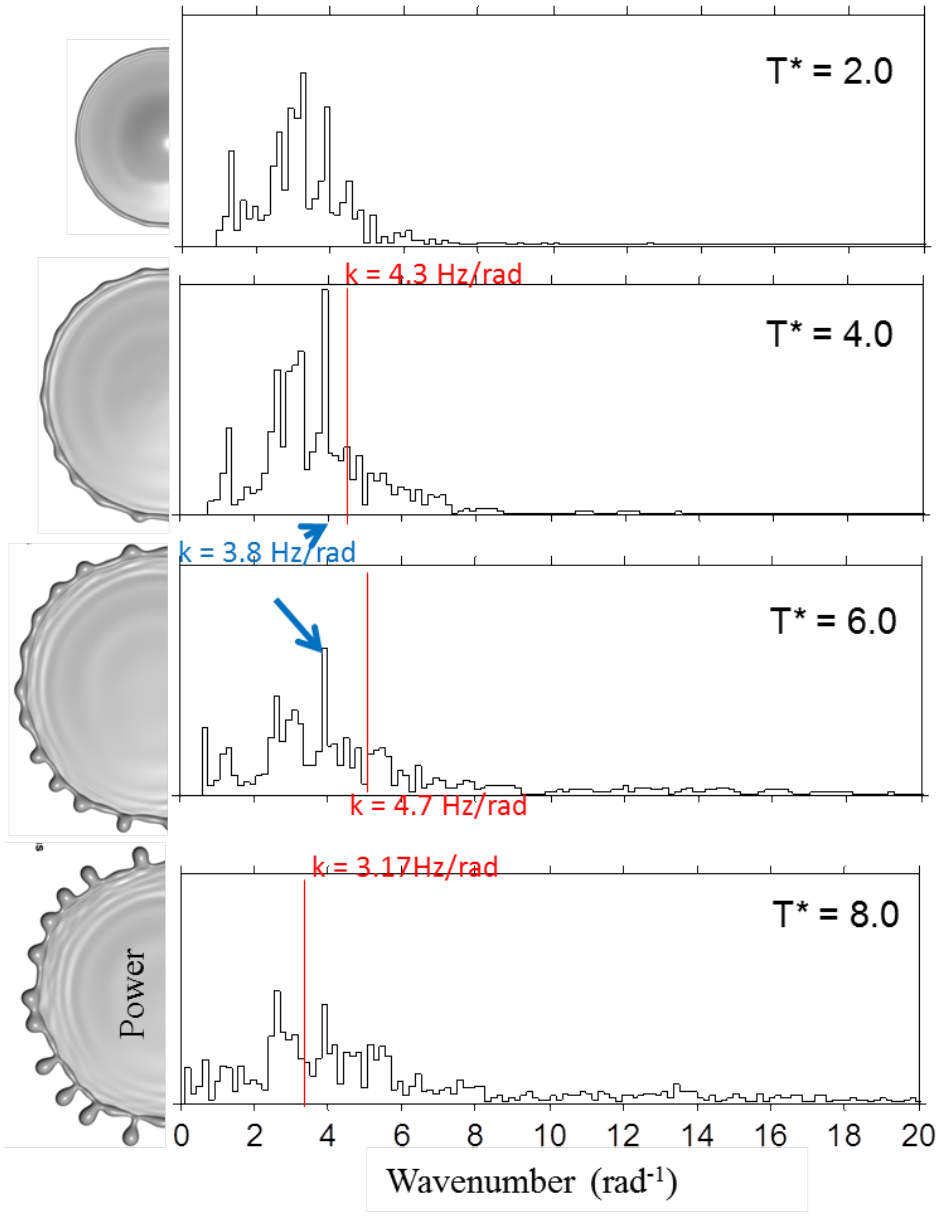


Figure 5-16. Wavenumber on the Taylor-Culick rim: case 3. Red lines are the fastest-growth wavenumber based on Rayleigh's theoretical work and the instantaneous rim diameter.

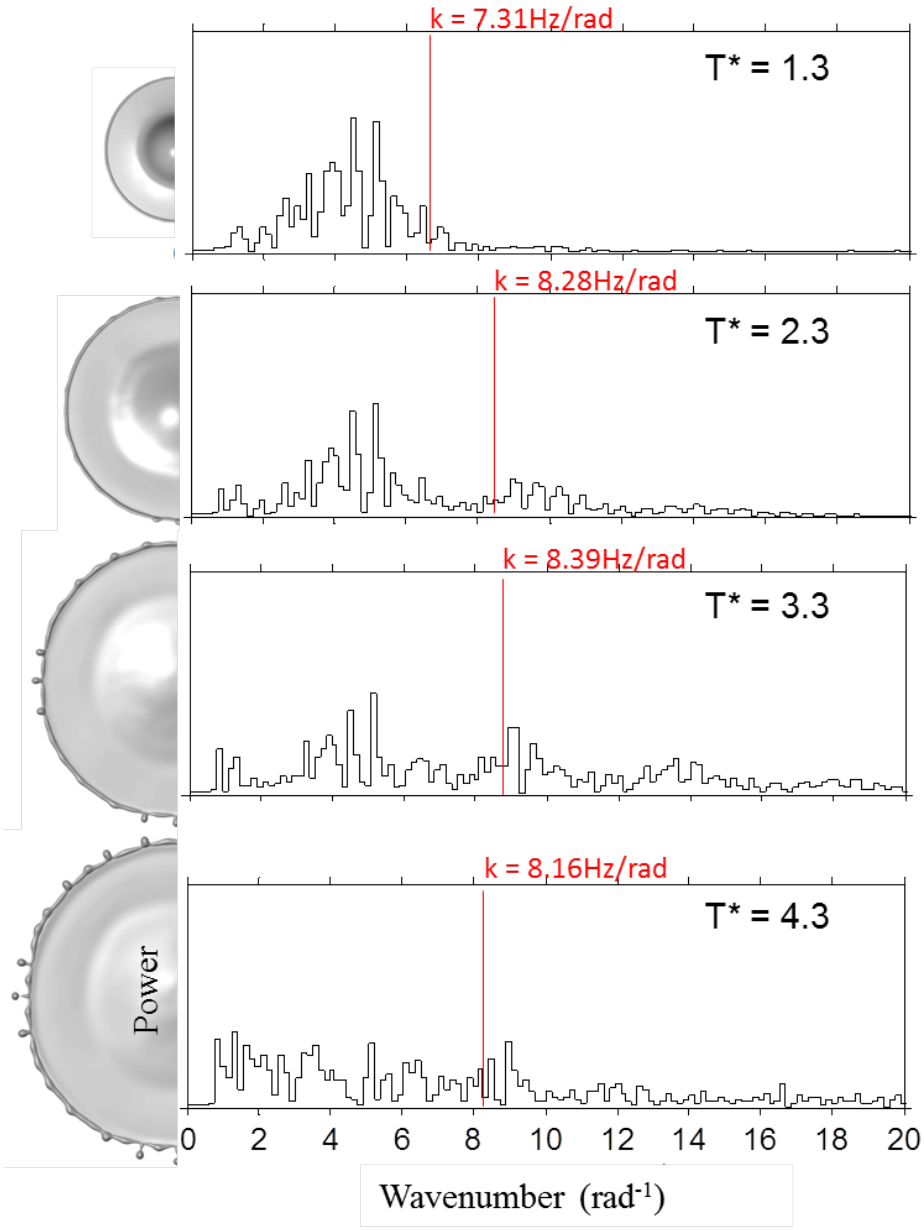


Figure 5-17. Wavenumber on the Taylor-Culick rim: case 5. Red lines are the fastest-growth wavenumber based on Rayleigh's theoretical work and the instantaneous rim diameter.

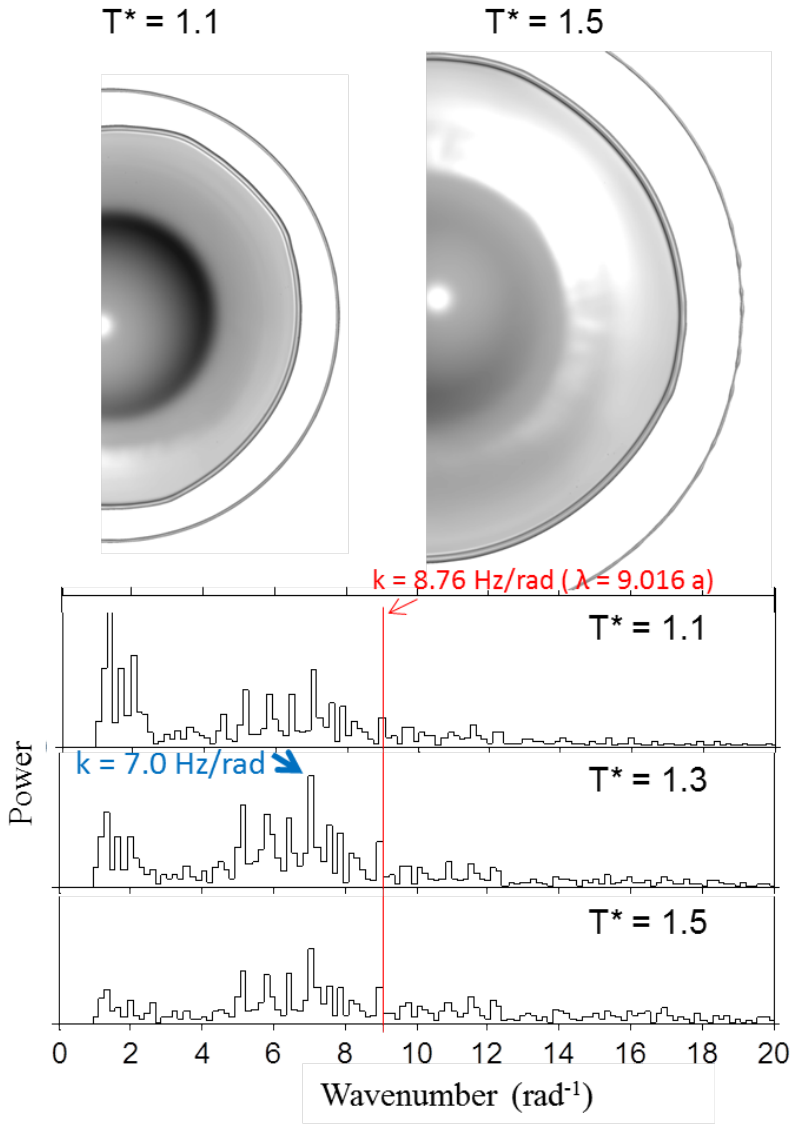


Figure 5-18. Wavenumbers on the Taylor-Culick rim: case 6. The red line is the fastest-growth wavenumber based on Rayleigh's theoretical work and the rim diameter at the moment of separation.

5.5 Secondary droplet size

Figure 5-19 is the size of secondary droplet of the primary breakup. The size of secondary droplet is calculated by measuring droplet diameter directly. When a droplet is an ellipsoid, we calculate an equivalent diameter of droplet by measuring its three semi-principle axes based on volume equivalence. The diameter of secondary droplets and log scale Weber number is nearly in a linear correlation.

In cases of a uniform liquid jet, the surface tension forces break the liquid thread into a series of main secondary droplets and liquid ligaments between the main drops. These ligaments become spherical eventually. One can use the most pronounced wavenumbers on liquid thread and volume conservation to estimate the droplet size distribution [103]. Zhang et al. [97] suggested the secondary droplet size for crown splashing problems can be correlated with the dominant wavenumber on the rim. Ashgriz and Mashayek investigated the viscous effect on the satellite droplet size of liquid jets. They concluded that for Ohnesorge numbers less than 0.1, there is no significant dependence of satellite droplet size with Ohnesorge number [104]. As a result, in present cases, where Ohnesorge number is in the order of 10^{-2}~10^{-3}, the deterministic factor for the size of secondary droplet size is the wavenumber on the TC rim. In some cases, the TC rim may start to breakup while the attached circular sheet is receding. The interaction of the receding sheet and Rayleigh-Plateau instability is the fingering effect. A finger structure may coalesce with its nearby structures, which results in more complex scenarios than a simple breakup based on wavenumbers of the TC rim. This is the primary reason that in case 3 (We 442), which has profound fingering effects, the standard deviation of the size of the secondary droplets is the largest among all cases. Overall, the size of the secondary droplet correlates with the most dominant wavenumber

on the TC rim, however, the factor deciding the most dominant wavenumber on the TC rim is unclear. For a simplified problem as considered by Rayleigh, dominant wavenumbers on a circular thread due to surface tensions forces is a function of the diameter of the rim. In present cases, it is shown that Rayleigh's linear instability model cannot predict the dominant wavenumber accurately. In addition, the diameter of the TC rim is time dependent and the onset timing of Rayleigh-Plateau instability is unclear. As a result, the deterministic factors to the dominant wavenumber of Rayleigh-Plateau instability are unclear. Further investigations are necessary to clarify this issue.

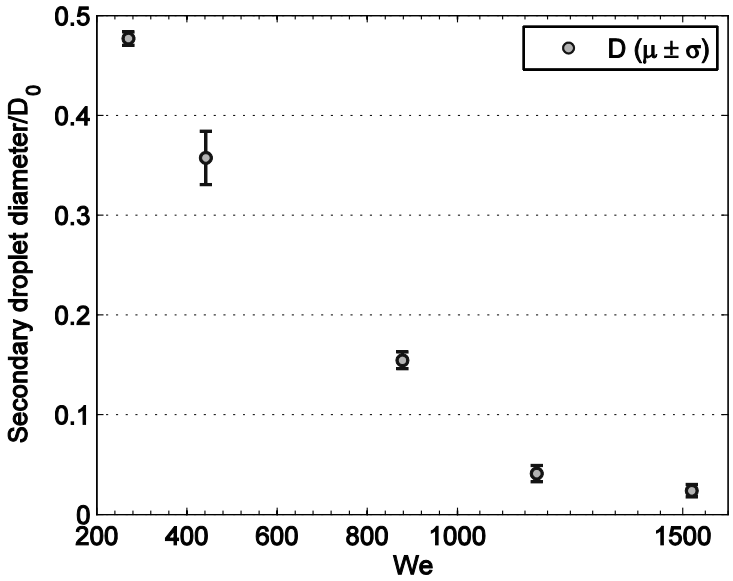


Figure 5-19. Secondary droplet size with respect to Weber number.

Dimensionless parameters based on the averaged length and velocity of the secondary droplet at the moment of the primary breakup may provide insight to breakup phenomena. We_s, Re_s, and Oh_s are listed in Table 5-5. We_s of these cases is from 6.4 to 51.1, and the increment in We_s is primarily due to higher kinetics energy left in the secondary droplets for the higher We cases. Especially for the cases with high, initial

kinetic energy (case 5, 6, and 7), the disintegration happens at the sheet expansion phase. At the expansion phase, the secondary droplets contain more kinetics energy than secondary droplet generated at the retraction phase (case 2 and 3). Oh_s is five thousandth for case 2 and 3, and jump to order of 10^{-2} for higher Weber number cases. It is still in the regime that viscous resistance is insignificant to surface tension force, but viscosity is gradually more influential to the morphology of phase boundaries. We may observe more viscous effect for higher-speed droplet collision. In summary, inertia forces and surface tension forces are still dominant in the small scale, but the nonlinearity of the surface instability results in difficulties for conclusive correlation between the global collision parameter We, Re, and Oh and collision consequence such as secondary droplet diameter.

Table 5-5. Weber, Reynolds, and Ohnesorge number based on the secondary droplet scale at the moment of primary breakup.

	Based on initial droplet size/velocity			Based on secondary droplet size/velocity		
Case	We	Re	Oh	We_s	Re_s	Oh_s
2	277	4686	0.0036	6.4	515	0.005
3	442	6207	0.0034	1.3	223	0.005
5	878	6650	0.0045	20.1	333	0.014
6	1176	7700	0.0045	51.1	272	0.027
7	1520	8750	0.0045	51.1	203	0.036

5.6 Summary

We summarize the observations of the instabilities for the head-on droplet collision under the high Weber number regime. At the initial stage of collision, the merged body ejects a high-speed liquid sheet radially on the impinging plane. This circular jet quickly decelerates as it expands further, and the disturbances on the edge of the sheet are amplified by Rayleigh-Taylor instability. The Taylor-Culick rim grows and propagates at lower speed than the circular sheet. In the long time, Rayleigh-Plateau

instability takes over the growth of the longitudinal disturbances since the growth rate of Rayleigh-Plateau type is much faster than Rayleigh-Taylor type. After the onset of Rayleigh-Plateau instability on the rim, the largest-amplitude wave number will not change a lot, and eventually determine the number of secondary droplets of the primary breakup. An illustration of this process is in Figure 5-20.

Figure 5-20. Interface evolution: instabilities on the edge of the circular sheet.

Chapter 6.

Conclusions

6.1 Summary

The motivation of the current work is to develop a distributed, computational platform for the multi-scale moving boundary problems. There are two primary objects accomplished in this thesis.

(i) Development of a parallel, adaptive Eulerian-Lagrangian method for moving boundary computations

(ii) Simulation of multi-scale droplet collision at a high Weber number regime for in-depth understanding of the interface instability and breakup mechanisms

The development of the parallel, adaptive Eulerian-Lagrangian interface tracking method is extremely challenging for a scalable implementation on distributed-memory systems. The interactions between the Eulerian and Lagrangian domains bring difficulties on load balance and communication complexity. The communication overhead is more severe with increasing grid size and number of processing units. Our concern is the data locality for the Eulerian-Lagrangian interpolation. Without it, a computation shows no scalability. As a result, we decompose Lagrangian domain according to the marker vicinities with respect to an Eulerian partition. Moreover, we introduce a parallel, cell-based unstructured mesh refinement method to bring substantial computational power to handle the multi-scale feature of the fluid boundaries. This numerical framework is an

integration of multiple algorithms including distributed computation of the front-tracking method, the Lagrangian mesh modification and reconstruction algorithms, and the parallel cell-based unstructured AMR. We highlight the performance study and applications as below.

(iii) The current field equation solver achieves the most efficient computation in the range of 10,000 to 20,000 cells per processor. This range is determined by the computation-to-communication ratio and the shared cache size per CPU.

(iv) The cell-based unstructured AMR shows standalone efficiency of 0.66 at 128 processors for a grid size of 8.19×10^6 cells. For a moving boundary simulation on a grid size 2.2 million cells, the overall efficiency of this parallel framework is 0.48 by using 128 processors. Based on a fixed-size problem, the performance each task group shows a descending trend as Eulerian, Eulerian-Lagrangian, Lagrangian, and AMR. This trend is consistent with decreasing value of the computation-to-communication ratio of each task group.

(v) The computation-to-communication ratio is the dominant factor on the efficiency of the parallel Eulerian-Lagrangian method. The load balance of the Lagrangian markers is not so influential on the overall performance due to the relative low computation load on the Lagrangian domain. Moreover, the decomposition approach for the Lagrangian interfaces used in the current work requires the least number of messages and the smallest size of send/receive data for Eulerian-Lagrangian interactions, which ensures scalability in a practical way.

For the computation of droplet collision at the high Weber number regime, here is a summary.

(i) The simulation of binary droplet collision carried on high-density ratio ($O(10^3)$), high Weber number, and high Laplace number $O(10^{5\text{-}6})$ demonstrates the capability of the present framework on multi-scale moving boundary

computations. The grid size of Eulerian domain is up to 22 million and the corresponding size of the triangular, Lagrangian mesh is 2.5 million.

(ii) The merge, expansion of the circular sheet, Taylor-Culick rim generated by the end pinching effect, longitudinal instabilities, fingering effect incurring by the sheet retraction, and breakup of droplets are successfully resolved for Weber number 210 to 1520. The histories of the interface evolution and breakup diameter are in good agreements with experimental results qualitatively and quantitatively.

(iii) Results shows details evolution of interfacial structures, which can provide rich information to elucidate previous arguments based on theoretical and experimental works. The conclusions regarding interface evolution are as follows.

a. For such high inertia flows, the gas-liquid boundary of the merged body far from the edge of the circular sheet evolves linearly with time and is independent of Weber number and Reynolds number. This is consistent with theoretical work using similarity analysis. In contrast, the developments of the rim and circular sheet are dependent on inertia, surface tension forces, and viscosity. The circular sheet has nearly constant thickness at the connecting point with the Taylor-Culick rim as it expands.

b. Rayleigh-Taylor and Rayleigh-Plateau instabilities coexist at the edge of the circular sheet. Rayleigh-Taylor instability is significant at the initial deceleration phase of the circular sheet right after droplet coalescence. Later Rayleigh-plateau instability takes over the growth of disturbances on the Taylor-Culick rim.

c. Rayleigh’s linear stability analysis cannot accurately approximate the dominant wavenumber on the Taylor-Culick rim. The profound wavenumber on the Taylor-Culick rim is relevant to the rim diameter and possibly dependent on the sheet thickness. Propagation of the circular

sheet also affects the spectra. The retraction of the circular sheet enhances disturbances on the Taylor-Culick rim as the fingering effect, and broadens the spectrum of the disturbances. The size of secondary droplets is dependent on the dominant wavenumber and rim diameter at the moment of primary breakup.

6.2 Future Work

This work presents the distributed computation of an adaptive three-dimensional Eulerian-Lagrangian interface tracking method. The results shown have established the effectiveness of the method for multi-scale moving boundary problems in a parallel processing framework.

So far, the largest problem we have solved has 23-million grid points. Beyond that, memory requirement exceeds the physical capacity of the computational nodes (3G bytes/processor). A primary issue is the storage of the Eulerian graph. To simplify the procedure of the load balancing and defining Eulerian and Lagrangian partitions, the unstructured graph of the entire Eulerian domain is retained in all the processors. It is possible to decompose global Eulerian graph to several distributed pieces, and collect each single piece of information from other processors when needed. However, a sophisticated data structure and a complex algorithm are necessary to do so. One of the approaches is using a “localized Eulerian graph.” This approach designs an abstract map of the Eulerian partitions. The partitions are vertices and spatial boundaries between partitions are edges connecting vertices. We decompose this abstract map into a number of groups and build a localized Eulerian graphs based on the collection of partitions in each group. As a result, a processor possesses a local Eulerian graph (connectivity) for cells in a collection, but not cells out of this collection. Restrictions are required for

remeshing and load balancing. Remeshing is applied on each group individually and the load-balancing algorithm can only relocate a cell to its group. This approach can free memory bottleneck and greatly reduces the far-end communications required in the current design.

The other issue is the robustness of interface reconstruction for the topology change. The current framework can handle the topology changes of multiple interfacial bodies at the same time. However, some of the breakup or merge scenarios may generate mesh debris in the level-contour reconstruction. Further improvements on this algorithm are need for the robustness and accuracy. The locally grid-based method for the surface mesh reconstruction developed by Bo et al. [33] is an appropriate direction for future improvement of the present reconstruction algorithm.

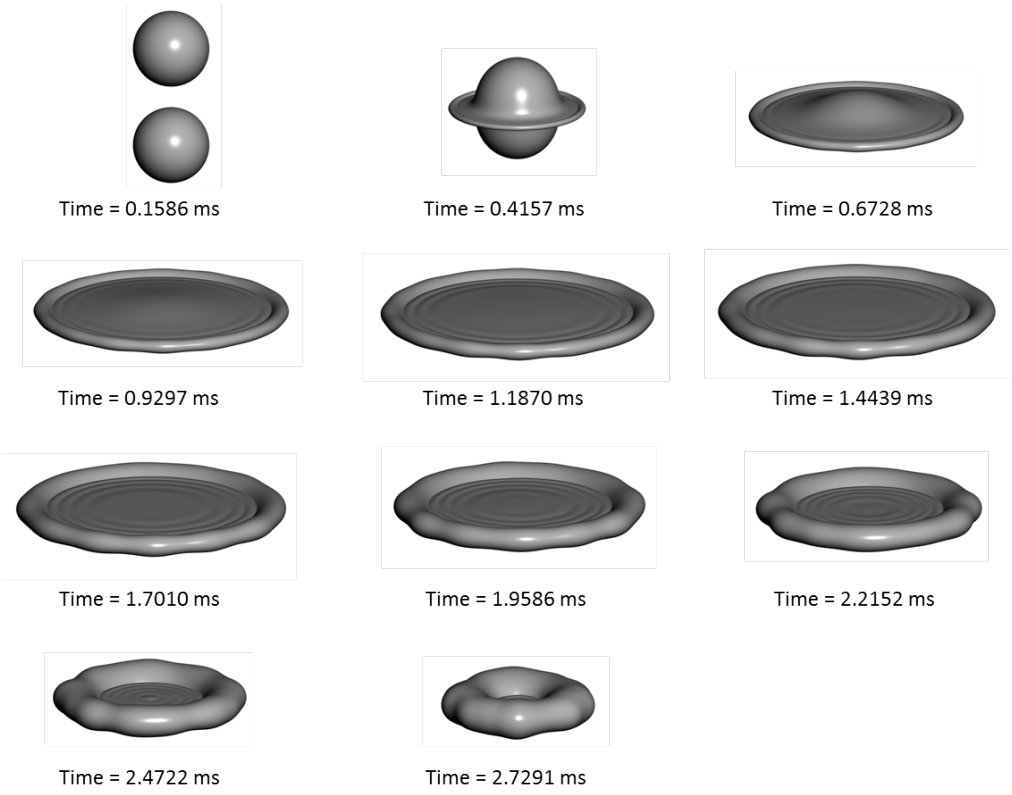
No turbulence modeling is included in this work yet. It should be implemented in the future. Voluminous studies of turbulence flows with the immersed boundary methods have been reported (Iaccarino and Verzicco [74], Yang and Balaras [105], and Gilmanov and Sotiropoulos [106]). The wall boundary treatment should be considered along with the ghost cell reconstruction of the sharp interface method as used in [72].

More case studies of binary droplet collision of high Weber and Reynolds numbers are desirable. These computations can clarify the mechanisms initializing interface stabilities, the effect of rim retraction to the longitudinal disturbance, or the defining factors for the size of secondary droplets of the primary breakup. Computations with gird resolutions higher than $D_0/\Delta x = 192$ are necessary to capture the secondary droplet breakup at Weber number more than 1500. Significant efforts will be required to realize the entire process of prompt splashing.

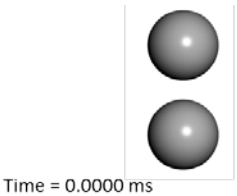
Appendix A.

Droplet collision history

A.1 Case 1



A.2 Case 2

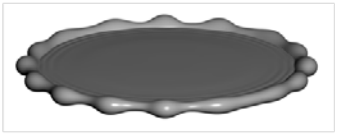

Time = 0.0000 ms

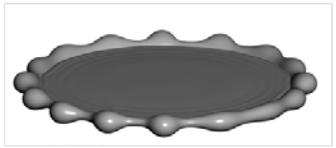

Time = 0.2585 ms

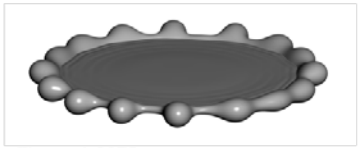

Time = 0.5681 ms

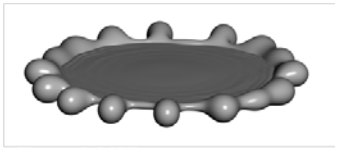

Time = 0.7747 ms

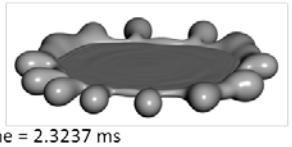

Time = 1.0197 ms

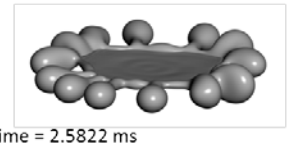

Time = 1.3042 ms

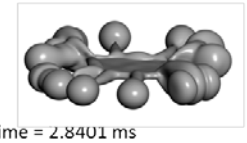

Time = 1.5493 ms

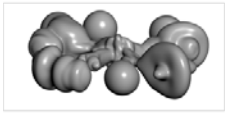

Time = 1.8075 ms


Time = 2.0657 ms

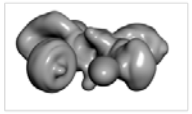

Time = 2.3237 ms


Time = 2.5822 ms

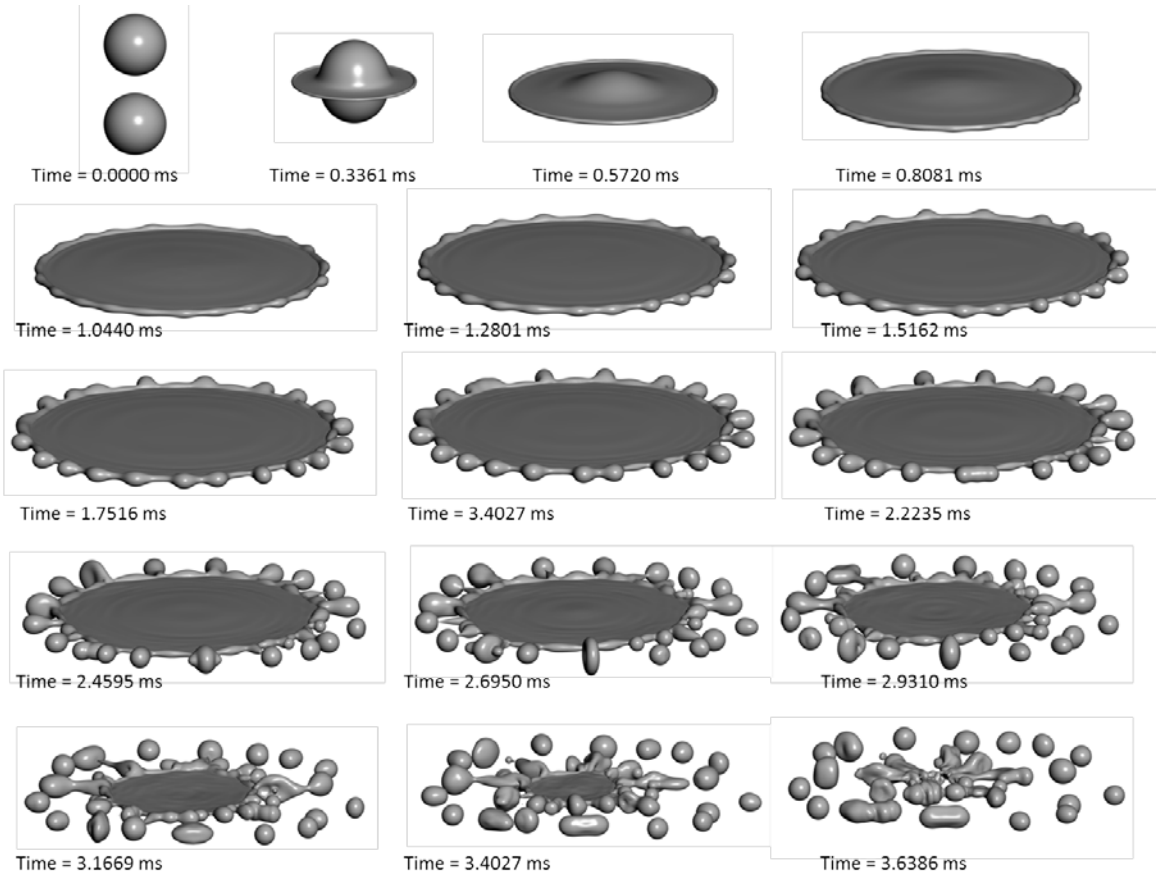

Time = 2.8401 ms


Time = 3.0984 ms

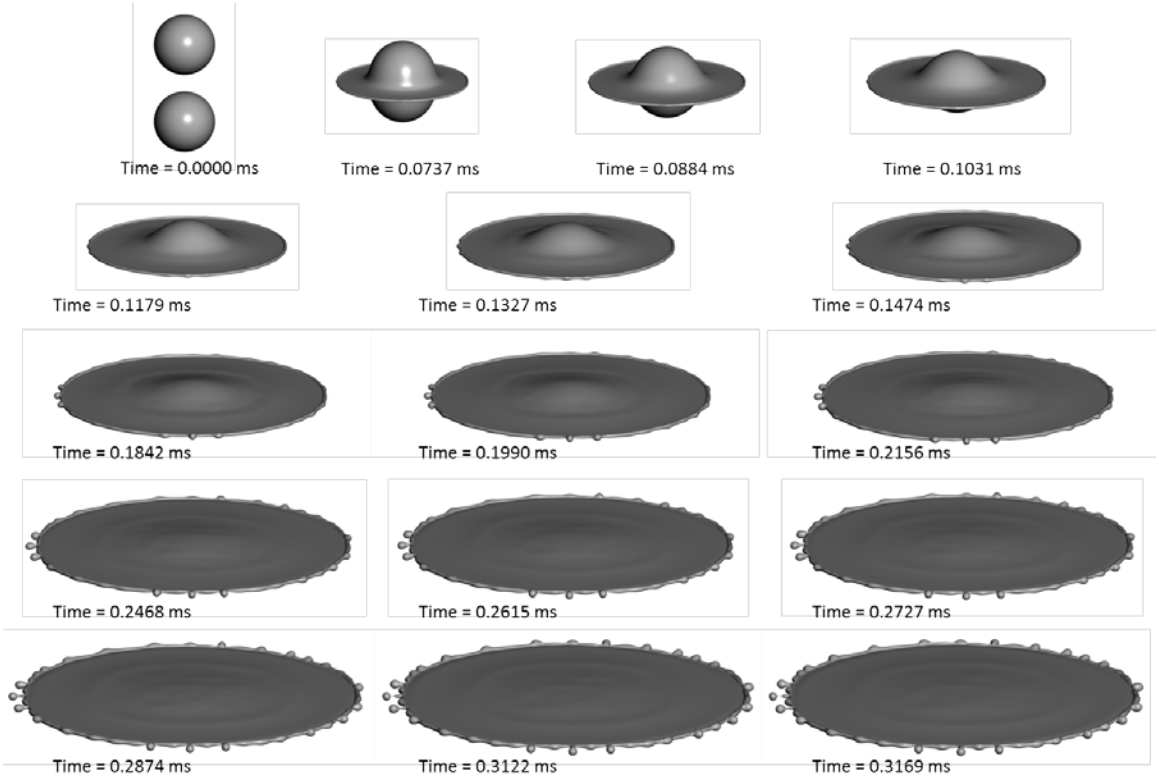

Time = 3.3562 ms


Time = 3.6148 ms

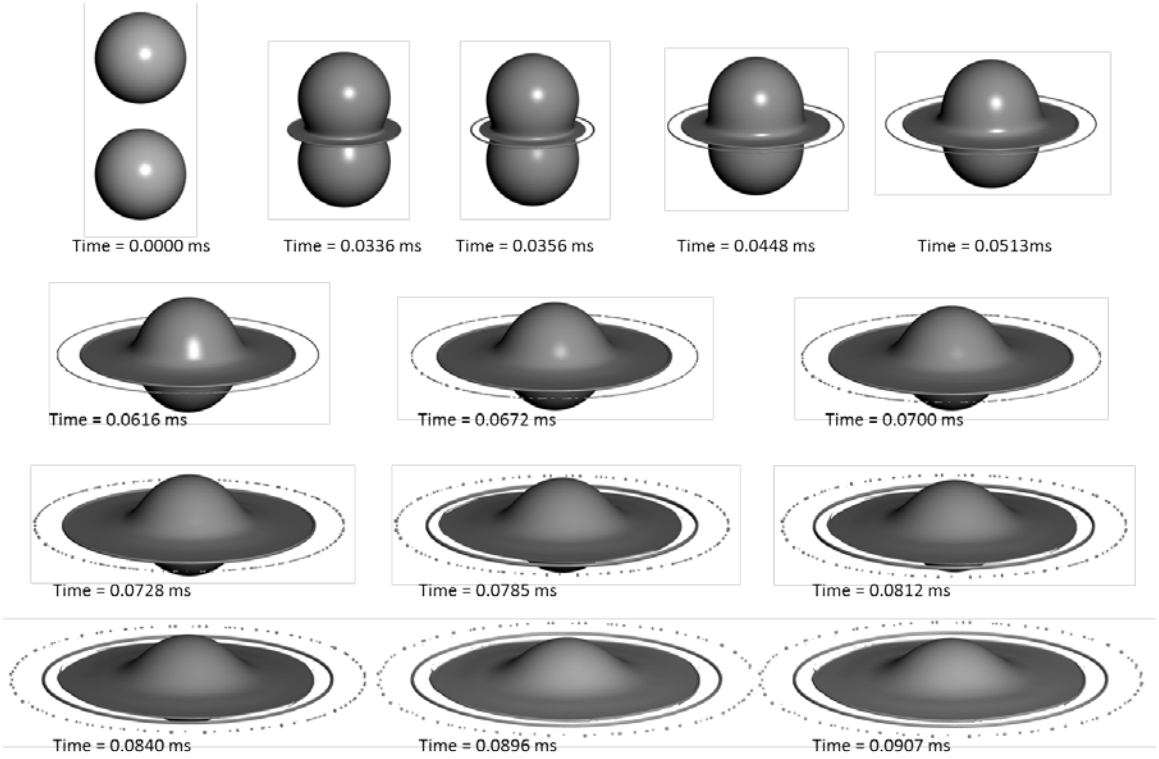
A.3 Case 3


Time = 0.0000 ms
Time = 0.3361 ms
Time = 0.5720 ms
Time = 0.8081 ms
Time = 1.0440 ms
Time = 1.2801 ms
Time = 1.5162 ms
Time = 1.7516 ms
Time = 3.4027 ms
Time = 2.2235 ms
Time = 2.4595 ms
Time = 2.6950 ms
Time = 2.9310 ms
Time = 3.1669 ms
Time = 3.4027 ms
Time = 3.6386 ms

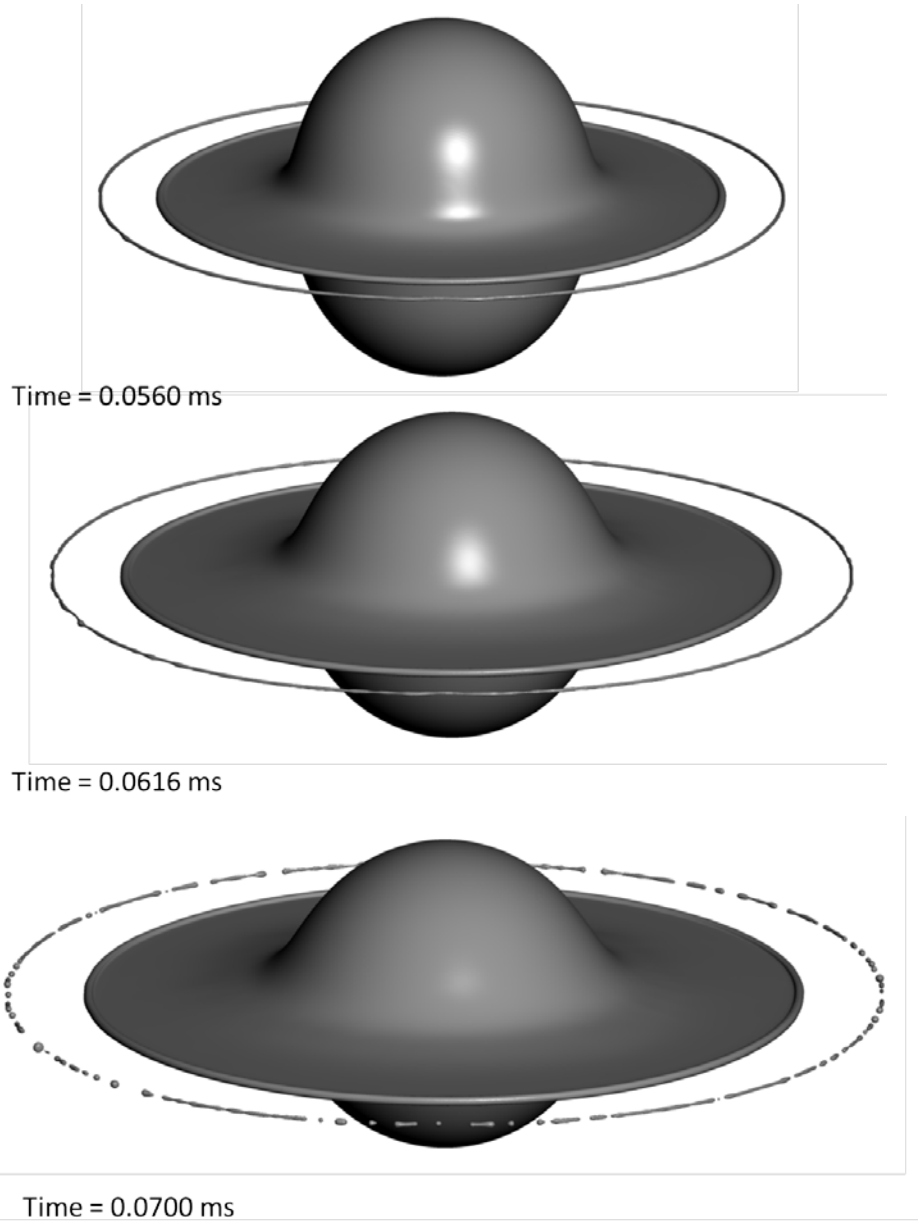
A.4 Case 5


Time = 0.0000 ms
Time = 0.0737 ms
Time = 0.0884 ms
Time = 0.1031 ms
Time = 0.1179 ms
Time = 0.1327 ms
Time = 0.1474 ms
Time = 0.1842 ms
Time = 0.1990 ms
Time = 0.2156 ms
Time = 0.2468 ms
Time = 0.2615 ms
Time = 0.2727 ms
Time = 0.2874 ms
Time = 0.3122 ms
Time = 0.3169 ms

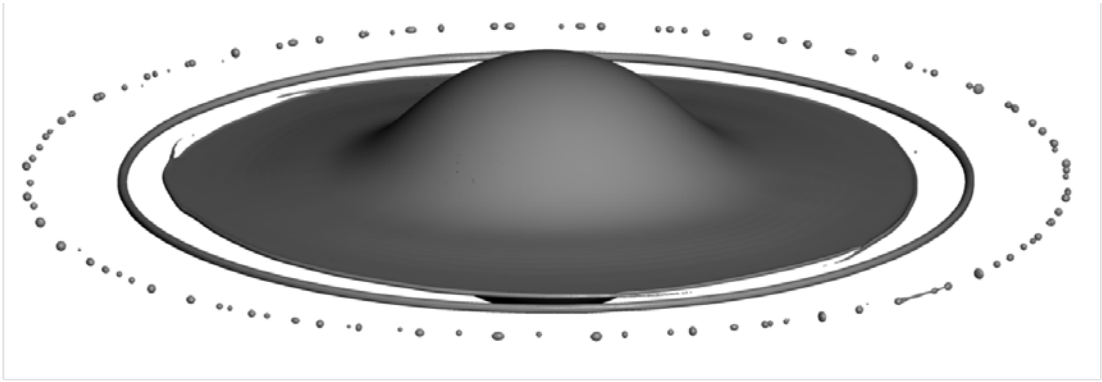
A.5 Case 7


Time = 0.0000 ms
Time = 0.0336 ms
Time = 0.0356 ms
Time = 0.0448 ms
Time = 0.0513ms
Time = 0.0616 ms
Time = 0.0672 ms
Time = 0.0700 ms
Time = 0.0728 ms
Time = 0.0785 ms
Time = 0.0812 ms
Time = 0.0840 ms
Time = 0.0896 ms
Time = 0.0907 ms

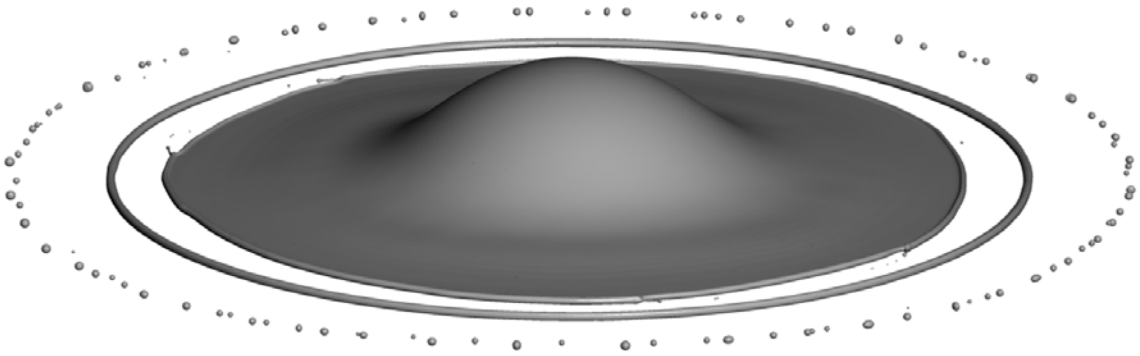
A.6 Case 7: Breakup of the detached TC rim under Rayleigh-Plateau instability



Time = 0.0785 ms



Time = 0.0840 ms



Time = 0.0907 ms

Bibliography

[1] J. Sim, W. Shyy, Interfacial flow computations using adaptive Eulerian-Lagrangian method for spacecraft applications, International Journal for Numerical Methods in Fluids, 68 (2012) 1438-1456.

[2] J. Sim, C.-K. Kuan, W. Shyy, Simulation of Spacecraft Fuel Tank Self-Pressurization Using Eulerian-Lagrangian Method, in: 49th AIAA Aerospace Science Meeting Including the New Horizons Forum and Aerospace Exposition, AIAA, Orlando, FL, 2011.

[3] C.-C. Tseng, Y. Wei, G. Wang, W. Shyy, Modeling of turbulent, isothermal and cryogenic cavitation under attached conditions, Acta Mechanica Sinica, 26 (2010) 325-353.

[4] W.E. Anderson, H.M. Ryan, R.J. Santoro, Impinging jet injector Atomization, in: V. Yang, W.E. Anderson (Eds.) Liquid rocket engine combustion instability, AIAA, 1995.

[5] N. Dombrowski, W.R. Johns, The aerodynamic instability and disintegration of viscous liquid sheets, Chemical Engineering Science, 18 (1963) 203-214.

[6] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas–Liquid Multiphase Flows, Cambridge University Press, 2011.

[7] K. Yamamura, J.-i. Fukuzaki, Y.H. Mori, Clathrate hydrate formation using liquid jets impinging on each other: An observational study using paired water jets or water and methylcyclohexane jets, Chemical Engineering Science, 66 (2011) 1844-1858.

[8] R. Singh, W. Shyy, Three-dimensional adaptive Cartesian grid method with conservative interface restructuring and reconstruction, Journal of Computational Physics, 224 (2007) 150-167.

[9] E. Uzgoren, R. Singh, J. Sim, W. Shyy, Computational modeling for multiphase flows with spacecraft application, Progress in Aerospace Sciences, 43 (2007) 138-192.

[10] C.W. Hirt, B. D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics, 39 (1981) 201-225.

[11] M. Rudman, Volume-tracking methods for interfacial flow calculations, International Journal for Numerical Methods in Fluids, 24 (1997) 671-691.

[12] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, S. Zaleski, Volume-of-Fluid Interface Tracking with Smoothed Surface Stress Methods for Three-Dimensional Flows, Journal of Computational Physics, 152 (1999) 423-456.

[13] D.L. Youngs, Time-dependent multi-material flow with large fluid distortion, in: K.W. Morton, M.J. Bianes (Eds.) Numerical methods for fluid dynamics, Academic Press,

New York, 1982, pp. 273-285.

[14] J.E. Pilliod, E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, Journal of Computational Physics, 199 (2004) 465-502.

[15] J. Lópeza, J. Hernándezb, P. Gómezb, F. Faura, A volume of fluid method based on multidimensional advection and spline interface reconstruction, Journal of Computational Physics, 195 (2004) 718-742.

[16] D. Gerlach, G. Tomar, G. Biswas, F. Durst, Comparison of volume-of-fluid methods for surface tension-dominant two-phase flows, International Journal of Heat and Mass Transfer, 49 (2006) 740-754.

[17] W.F. Noh, P. Woodward, SLIC (Simple Line Interface Calculation), in: A.I. van de Vooren, P.J. Zandbergen (Eds.) Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede, Springer Berlin Heidelberg, 1976, pp. 330-340.

[18] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics, 79 (1988) 12-49.

[19] M. Sussman, P. Smereka, S. Osher, A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow, Journal of Computational Physics, 114 (1994) 146-159.

[20] M. Sussman, E.G. Puckett, A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows, Journal of Computational Physics, 162 (2000) 301-337.

[21] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, New York, 2002.

[22] K. Yokoi, A practical numerical framework for free surface flows based on CLSVOF method, multi-moment methods and density-scaled CSF model: Numerical simulations of droplet splashing, Journal of Computational Physics, 232 (2013) 252-271.

[23] W. Shyy, H.S. Udaykumar, M.M. Rao, R.W. Smith, Computational Fluid Dynamics with Moving Boundaries, in, Taylor & Francis, Philadelphia, 1996, pp. 1-19.

[24] C.S. Peskin, D.M. McQueen, A three-dimensional computational method for blood flow in the heart I. Immersed elastic fibers in a viscous incompressible fluid, Journal of Computational Physics, 81 (1989) 372-405.

[25] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, Journal of Computational Physics, 100 (1992) 25-37.

[26] H.S. Udaykumar, H.-C. Kan, W. Shyy, R. Tran-Son-Tay, Multiphase Dynamics in Arbitrary Geometries on Fixed Cartesian Grids, Journal of Computational Physics, 137 (1997) 366-405.

[27] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries, Journal of Computational Physics, 156 (1999) 209-240.

[28] J. Glimm, J.W. Grove, X.L. Li, D.C. Tan, Robust computational algorithms for dynamic interface tracking in three dimensions, SIAM J. SCI. COMPUT, 21 (2000) 2240-2256.

[29] C.S. Peskin, The immersed boundary method, Acta Numerica, 11 (2003) 479-517.

[30] B.E. Griffith, R.D. Hornung, D.M. McQueen, C.S. Peskin, Parallel and Adaptive Simulation of Cardiac Fluid Dynamics, in: M. Parashar, X. Li (Eds.) Advanced Computational Infrastructures for Parallel and Distributed Adaptive Applications, John Wiley and Sons, 2009, pp. 105-130.

[31] S.V. Marella, A Parallelized sharp-interface fixed grid method for moving boundary problems, in: Mechanical Engineering, University of Iowa, Iowa City, IA, 2006.

[32] G. Tryggvason, B. Bunner, A. Esmaeeli, N. Al-Rawahi, W. Tauber, J. Han, Y.J. Jan, D. Juric, S. Nas, A front-tracking method for the computations of multiphase flow, Journal of Computational Physics, 169 (2001) 708-759.

[33] W. Bo, X. Liu, J. Glimm, X. Li, A Robust Front Tracking Method: Verification and Application to Simulation of the Primary Breakup of a Liquid Jet, SIAM Journal on Scientific Computing, 33 (2011) 1505-1524.

[34] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, Journal of Computational Physics, 228 (2009) 5838-5866.

[35] G. Agbaglah, S. Delaux, D. Fuster, J. Hoepffner, C. Josserand, S. Popinet, P. Ray, R. Scardovelli, S. Zaleski, Parallel simulation of multiphase flows using octree adaptivity and the volume-of-fluid method, Comptes Rendus Mecanique, 339 (2011) 194-207.

[36] Y. Wang, S. Simakhina, M. Sussman, A hybrid level set-volume constraint method for incompressible two-phase flow, Journal of Computational Physics, 231 (2012) 6438-6471.

[37] T. Espostiongaro, C. Cavazzoni, G. Erbacci, A. Neri, M. Salvetti, A parallel multiphase flow code for the 3D simulation of explosive volcanic eruptions, Parallel Computing, 33 (2007) 541-560.

[38] M. Sussman, A parallelized, adaptive algorithm for multiphase flows in general geometries, Computers & Structures, 83 (2005) 435-444.

[39] J.M. Rodriguez, O. Sahni, R.T. Lahey, K.E. Jansen, A parallel adaptive mesh method for the numerical simulation of multiphase flows, Computers & Fluids, (2013).

[40] C.-K. Kuan, J. Sim, E. Hassan, W. Shyy, Parallel Eulerian-Lagrangian Method with Adaptive Mesh Refinement for Moving Boundary Computation, in: 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, AIAA, 2013, pp. AIAA 2013-0370.

[41] C.-K. Kuan, J. Sim, W. Shyy, Adaptive thermo-fluid moving boundary computations for interfacial dynamics, Acta Mechanica Sinica, 28 (2012) 999-1021.

[42] E. Uzgoren, J. Sim, W. Shyy, Marker-based, 3-D adaptive Cartesian grid method for multiphase flow around irregular geometries, Communications in Computational Physics, 5 (2009) 1-41.

[43] J. Capecelatro, O. Desjardins, An Euler–Lagrange strategy for simulating particle-laden flows, Journal of Computational Physics, 238 (2013) 1-31.

[44] D. Darmana, N.G. Deen, J.A.M. Kuipers, Parallelization of an Euler-Lagrange model using mixed domain decomposition and a mirror domain technique: Application to dispersed gas-liquid two-phase flow, Journal of Computational Physics, 220 (2006) 216-248.

[45] M. Herrmann, A parallel Eulerian interface tracking/Lagrangian point particle multi-scale coupling procedure, Journal of Computational Physics, 229 (2010) 745-759.

[46] B. Nkonga, P. Charrier, Generalized parcel method for dispersed spray and message passing strategy on unstructured meshes, Parallel Computing, 28 (2002) 369-398.

[47] B.T.N. Gunney, A.M. Wissink, D.A. Hysom, Parallel clustering algorithms for structured AMR, Journal of Parallel and Distributed Computing, 66 (2006) 1419-1430.

[48] P. MacNeice, K.M. Olson, C. Mobarry, R. de Fainchtein, C. Packer, PARAMESH: A parallel adaptive mesh refinement community toolkit, Computer Physics Communications, 126 (2000) 330-354.

[49] R. Deiterding, A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains, Computers Structures, 87 (2009) 769-783.

[50] M. Berger, I. Rigoutsos, An algorithm for point clustering and grid generation, IEEE Transactions on Systems, Man and Cybernetics, 21 (1991) 1278-1286.

[51] D. Zuzio, J.L. Estivalezes, An efficient block parallel AMR method for two phase interfacial flow simulations, Computers & Fluids, 44 (2011) 339-357.

[52] C. Jablonowski, M. Herzog, J.E. Penner, R.C. Oehmke, Q.F. Stout, B.v. Leer, K.G. Powell, Block-Structured Adaptive Grids on the Sphere: Advection Experiments, Monthly Weather Review, 134 (2006) 3691–3713.

[53] K.G. Powell, D.L.D. Zeeuw, I.V. Sokolov, I. Tamas, Q. Stout, Parallel, AMR MHD for Global Space Weather Simulations, in: T. Plewa, T. Linde, V.G. Weirs (Eds.) Adaptive Mesh Refinement - Theory and Applications, Springer, Berlin Heidelberg, 2005, pp. 473-490.

[54] C. Burstedde, L.C. Wilcox, O. Ghattas, Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing, 33 (2011) 1103-1133.

[55] H.S. Udaykumar, S. Krishnan, S.V. Marella, Adaptively refined, parallelised sharp interface Cartesian grid method for three-dimensional moving boundary problems, International Journal of Computational Fluid Dynamics, 23 (2009) 1-24.

[56] G. Agbaglah, S.e.b. Delaux, D. Fuster, J.e.r. Hoepffner, m. o, C. Josserand, S.e.p. Popinet, P. Ray, R. Scardovelli, S.e.p. Zaleski, Parallel simulation of multiphase flows using octree adaptivity and the volume-of-fluid method, Comptes Rendus Mecanique, 339 194-207.

[57] D. Fuster, A. Baguéa, T. Boeck, L.L. Moyne, A. Leboissetier, S. Popinet, P. Ray, R.

Scardovelli, S. Zaleski, Simulation of primary atomization with an octree adaptive mesh refinement and VOF method, International Journal of Multiphase Flow, 35 (2009) 550-565.

[58] O.S. Lawlor, S. Chakravorty, T.L. Wilmarth, N. Choudhury, I. Dooley, G. Zheng, L.V. Kale, ParFUM: a parallel framework for unstructured meshes for scalable dynamic physics applications, Engineering with Computers, 22 (2006) 215-235.

[59] B.S. Kirk, J.W. Peterson, R.H. Stogner, G.F. Carey, libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations, Engineering with Computers, 22 (2006) 237-254.

[60] M.R.H. Nobari, G. Tryggvason, Numerical simulations of three-dimensional drop collisions, AIAA Journal, 34 (1996) 750-755.

[61] N. Nikolopoulosa, K.-S. Nikasa, G. Bergeles, A numerical investigation of central binary collision of droplets, Computers & Fluids, 38 (2009) 1191-1202.

[62] K.-L. Pan, P.-C. Chou, Y.-J. Tseng, Binary droplet collision at high Weber number, Physical Review E, 80 (2009).

[63] L. Rayleigh, On the Instability of Jets, Proceeding of London Mathmatical Society, 10 (1878) 4-13.

[64] G.I. Taylor, The Instability of Liquid Surfaces when Accelerated in a Direction Perpendicular to their Planes. I. Waves on fluid sheets, Proc. R. Soc. Lond. A, 201 (1950) 192-196.

[65] F.E.C. Culick, Comments on a Ruptured Soap Film, Journal of Applied Physics, 31 (1960) 1128.

[66] C.T. Crowe, Multiphase flow handbook, in, Taylor & Francis, 2005, pp. 1156.

[67] J.O. Hinze, Fundamentals of the hydrodynamic mechanism of splitting in dispersion processes, AIChE Journal, 1 (1955) 289-295.

[68] L.-P. Hsiang, G.M. Faeth, Near-limit drop deformation and secondary breakup, International Journal of Multiphase Flow, 18 (1995) 635-652.

[69] W.L. Haberman, R.L. Morton, An experimental investigation of the drag and shape of air bubbles rising in various liquids, in: D. W. Taylor Model Basin Report, Departman of Navy, Washington, DC, 1953.

[70] J. Sim, 3-D adaptive Eulerian-Lagrangian method for multiphase flows with spacecraft applications, in: Aerospace Engineering, University of Michigan, Ann Arbor, 2010.

[71] C.-K. Kuan, J. Sim, W. Shyy, Parallel, Adaptive Grid Computing of Multiphase Flows in Spacecraft Fuel Tanks, in: 50th AIAA Aerospace Science Meeting Including the New Horizons Forum and Aerospace Exposition, AIAA, Nashville, Tennessee, 2012.

[72] Y.-H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, Journal of Computational Physics, 192 (2003) 593-623.

[73] T. Gao, Y.-H. Tseng, X.-Y. Lu, An improved hybrid Cartesian/immersed boundary method for fluid–solid flows, International Journal for Numerical Methods in Fluids, 55

(2007) 1189-1211.

[74] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, Applied Mechanics Reviews, 56 (2003) 331-347.

[75] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, Journal of Computational Physics, 227 (2008) 4825-4852.

[76] J. Kim, D. Kim, H. Choi, An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries, Journal of Computational Physics, 171 (2001) 132-150.

[77] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc Users Manual, in, Argonne National Laboratory, 2012.

[78] R.D. Falgout, U.M. Yang, hypre: a Library of High Performance Preconditioners, in: P.M.A. Sloot, C.J.K. Tan, J.J. Dongarra, A.G. Hoekstra (Eds.) Computational Science - ICCS 2002, Springer Berlin, Heidelberg, 2002, pp. 632-641.

[79] G. Karypis, V. Kumar, MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0, in, \url{http://www.cs.umn.edu/~metis}.

[80] S. Schamberger, J.-M. Wierum, Graph Partitioning in Scientific Simulations: Multilevel Schemes versus Space-Filling Curves, in: V. Malyshkin (Ed.) Parallel Computing Technologies, Springer Berlin, Heidelberg, 2003, pp. 165-179.

[81] R. Singh, Three-dimensional marker-based multiphase flow computation using adaptive Cartesian grid techniques, in: Department of Mechanical and Aerospace Engineering, University of Florida, 2006.

[82] R. Lohner, Parallel unstructured grid generation, Computer Methods in Applied Mechanics and Engineering, 95 (1992) 343-357.

[83] N. Chrisochoides, Parallel Mesh Generation, in: A.M. Bruaset, A. Tveito (Eds.) Numerical Solution of Partial Differential Equations on Parallel Computers, Springer-Verlag, New York, 2006, pp. 237-259.

[84] U. Ghia, K. Ghia, C. Shin, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, Journal of Computational Physics, 48 (1982) 387-411.

[85] K. Taira, T. Colonius, The immersed boundary method: A projection approach, Journal of Computational Physics, 225 (2007) 2118-2137.

[86] T. Colonius, K. Taira, A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary condition, Computer Methods in Applied Mechanics and Engineering, 197 (2008) 2131-2146.

[87] M.N. Linnick, H.F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, Journal of Computational Physics, 204 (2005) 157-192.

[88] B. Fornberg, A numerical study of steady viscous flow past a circular cylinder,

Journal of Fluid Mechanics, 98 (1980) 819-855.

[89] T.A. Johnson, V.C. Patel, Flow past a sphere up to a Reynolds number of 300 Journal of Fluid Mechanics, 378 (1999) 19-70.

[90] M.v.S. Annaland, N.G. Deen, J.A.M. Kuipers, Numerical simulation of gas bubbles behaviour using a three-dimensional volume of fluid method, Chemical Engineering Science, 60 (2005) 2999-3011.

[91] J.R. Grace, Shapes and velocities of bubbles rising in infinite liquids, Chemical Engineering Research and Design, 51a (1973) 116-120.

[92] J. Qian, C.K. Law, Regimes of coalescence and separation in droplet collision, Journal of Fluid Mechanics, 331 (1997) 59-80.

[93] Y. Pan, K. Suga, Numerical simulation of binary liquid droplet collision, Physics of Fluids, 17 (2005) 082105.

[94] C. Gotaas, P. Havelka, H.A. Jakobsen, H.F. Svendsen, M. Hase, Effect of viscosity on droplet-droplet collision outcome: Experimental study and numerical comparison, Physics of Fluids, 19 (2007) 102106.

[95] N. Savva, J.W.M. Bush, Viscous sheet retraction, Journal of Fluid Mechanics, 626 (2009) 221-240.

[96] G.I. Taylor, The dynamics of thin sheets of fluid. III Disintegration of fluid sheets, Proc. R. Soc. Lond. A, 253 (1959) 313-321.

[97] L.V. Zhang, P. Brunet, J. Eggers, R.D. Deegan, Wavelength selection in the crown splash, Physics of Fluids, 22 (2010) 122105.

[98] I.V. Roisman, E. Berberović, C. Tropea, Inertia dominated drop collisions. I. On the universal flow in the lamella, Physics of Fluids, 21 (2009) 052103.

[99] C. Clanet, C. Beguin, D. Richard, D. Quere, Maximal deformation of an impacting drop, Journal of Fluid Mechanics, 517 (2004) 199-208.

[100] G. Agbaglah, C. Josserand, S. Zaleski, Longitudinal instability of a liquid rim, Physics of Fluids, 25 (2013) 022103.

[101] J. Eggers, M.A. Fontelos, C. Josserand, S. Zaleski, Drop dynamics after impact on a solid wall: Theory and simulations, Physics of Fluids, 22 (2010) 062101.

[102] R. Krechetnikov, Stability of liquid sheet edges, Physics of Fluids, 22 (2010) 092101.

[103] N. Ashgriz, A.L. Yarin, Capillary instability of Free liquid Jets, in: N. Ashgriz (Ed.) Handbook of Atomization and Sprays : Theory and Applications, Springer, 2011.

[104] N. Ashgriz, F. Mashayek, Temporal analysis of capillary jet breakup, Journal of Fluid Mechanics, 291 (1995) 163-190.

[105] J. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, Journal of Computational Physics, 215 (2006) 12-40.

[106] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for

simulating flows with 3D, geometrically complex, moving bodies, Journal of Computational Physics, 207 (2005) 457-492.