論文 / 著書情報
Article / Book Information

題目(和文)	運動量保存型弱圧縮性二相流の大規模 GPU 計算ソルバー
Title(English)	A Momentum-conserving Weakly Compressible Navier-Stokes Solver for Large-scale Two-phase Flow Simulation on GPUs
著者(和文)	YangKai
Author(English)	Kai Yang
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第12521号, 授与年月日:2023年9月22日, 学位の種別:課程博士, 審査員:青木 尊之,末包 哲也,奥野 喜裕,大西 領,門永 雅史
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第12521号, Conferred date:2023/9/22, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

A Momentum-conserving Weakly Compressible Navier-Stokes Solver for Large-scale Two-phase Flow Simulation on GPUs

by

Kai Yang

A thesis submitted in partial satisfaction of the

requirements for the degree of

Doctor of Engineering

in the

Department of Mechanical Engineering

of the

School of Engineering

of the

Tokyo Institute of Technology

Supervisor:

Professor Takayuki Aoki

Spring 2023

Acknowledgments

First and foremost, I would like to express my sincere thanks to Prof. Takayuki Aoki for giving me the opportunity to pursue PhD program in this laboratory, for his guidance and constant encouragement throughout my doctoral research, and for motivating me to engage in academic conferences and publish journal papers.

I am grateful to all members of our laboratory for accompanying me through the special and memorable three years. Because of the outbreak of coronavirus disease, we had to work remotely and didn't get to meet face-to-face for a long time. But we still actively communicate and help each other via video meetings and emails. I deeply appreciate Dr. Sitompul Yos Panagaman and Dr. Shintaro Matsushita for their discussions and insightful suggestions on my work. Thank you Ritsuya Aoki, Yuma Tamaoki, Kento Miyoshi and Hiroatsu Kitagawa for studying on the related topics together and extending my solver to many interesting areas. I want to say thanks to Xuyang Chen, Tongda Lian, Dawei Shen, Haocong Wang, Yuwei Yin and Tinghui Cao who are also lab members from China and other friends I met in Japan, thank you for being involved in my life and spending a portion of these years together. I would also like to thank the secretary Ayako Keneko and secretary Satomi Sugimoto of Aoki lab for helping me with all kinds of business procedures and reminding me of schedule.

I am so thankful for to Prof. Christophe Josserand, Dr. Kuan-Ling Huang, Dr. Mathilde Tavares, Dr. Jiang Nan and Dr. Chuhan Wang of Ecole Polytechnique and Dr. Stéphane Popinet of Sorbonne University for welcoming and hosting me in France during my visiting research. I appreciate Liang Huang and Shaojie Wang for guiding me during my internship at AMD. I also appreciate Dr. Xinxin Zhang, Dr. Xinlei Wang and Shuliang Lu of ZenusTech for leading me into the fascinating field of computer graphics.

I want to give my special thanks to my girlfriend Wenyi Liu for coming to Japan to accompany the study and life with each other. Thank you for giving me courage and emotional support when I am depressed. Last but not least, I want to express my profound gratitude to my parents. Thank you so much for raising me up and supporting me spiritually in my life. During the years of studying abroad, thank you for bringing me strength and for the financial support.

This research was partly supported by a Grant-in-Aid for Scientific Research (S) 19H05613, from the Japan Society for the Promotion of Science (JSPS), and Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (JHPCN), jh200018 and jh210013, and High Performance Computing Infrastructure (HPCI) hp210129 projects, and JST SPRING, grant number JPMJSP2106. I thank the Global Scientific Information and Computing Center, Tokyo Institute of Technology for use of the computing resources of the TSUBAME 3.0 supercomputer and the Information Technology Center of Nagoya University for use of the computing resources of the Flow Type II supercomputer.

Abstract

In order to make the best use of the increasing computing performance of modern processors, new demand on scalability is put forward for the numerical methods. In the simulation of incompressible flow, the implicit Poisson solver usually accounts for a large proportion of execution time, this fact motivates researchers to turn attention to explicit methods for nearly incompressible flow at low Mach number. This thesis is devoted to studying weakly compressible flow simulation for violent two-phase flows with high density ratio.

At first, an evolving pressure projection method for numerical computation of the weakly compressible Navier-Stokes equations is proposed. Fully explicit time integration is achieved using an independent pressure evolution equation. To damp the acoustic wave in a weakly compressible fluid flow, the pressure evolution equation is iteratively computed coupled with a projection step. By introducing the phase field model for interface capturing, this solver can be directly applied to two-phase flow simulations using a one-fluid model. Exact mass conservation is ensured by a finite-volume formulation of the conservative phase field equation. The accuracy of present method is validated by various two-phase flow benchmarks.

Then, a consistent and conservative formulation for mass and momentum transport is proposed in the context of simulating incompressible two-phase flows by using weakly compressible method. Coupled with the volume of fluid method for capturing interfaces, the mass and momentum fluxes are evaluated in a consistent manner using the finite volume method. In addition, a special implementation of the pressure projection is devised to avoid velocity-pressure decoupling on a collocated grid. The momentum conserving property of the solver is demonstrated by solving realistic two-phase flow problems, including the dam break and the liquid jet simulations.

Finally, to address the demand for high-performance large-scale simulation of two-phase flows, the scalability and performance of two conservative weakly compressible solvers on the multi-GPU cluster are evaluated. High-fidelity numerical results of milk crown, dam break and jet atomization are produced. Spatially sparse data structures including adaptive mesh refinement and spatial hash based sparse grid are implemented on GPU to minimize the need for computing resources.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Derived from the continuity and conservation of mass, momentum and energy, the Navier-Stokes equations are one of the predominant theories to describe the motion of fluids. When a constant density is introduced through the incompressibility assumption, the continuity equation of fluid becomes that the divergence of the flow velocity is zero. The pressure gradient term in the momentum equation is merely a multiplier to enforce the divergence-free constraint, based on which a pressure Poisson equation is usually solved. Because of the elliptic-parabolic characteristics of the incompressible Navier-Stokes equations, numerical computations that use fractional-step semi-implicit methods have considerable complexity [1]. Moreover the solution algorithm has limited scalability and poses a great challenge for distributed parallel computing [2–4]. On the other hand, if slight compressibility, i.e., a weakly compressible fluid, is acceptable, the Poisson equation can be replaced by an independent time evolution equation for pressure. The pioneering work of Chorin on the artificial compressibility method (ACM) [5] has enabled a pressure evolution equation to be derived from compressible Navier-Stokes equations under low Mach number and isothermal conditions [6, 7]. This hyperbolic equation for pressure can be explicitly advanced in time by using a local spatial stencil. It significantly benefits scalability of computation in spite of the speed-of-sound restriction on the time step. A comparison of various pressure evolution equations has been conducted to verify their accuracy [8].

Traditionally, the artificial compressibility method is designed for incompressible Navier-Stokes equations, since the term involving the time derivative of pressure vanishes in the steady state, and this results in a divergence-free velocity field. For the unsteady-flow problem, a dual-time stepping technique is used to update the pressure and velocity in another pseudo time until converge to a quasi-steady state [9]. Although this treatment

can completely remove the acoustic effect, it requires a large number of iterations, as in the case of a Poisson solver. The GPU performance comparison of a weakly compressible Navier-Stokes solver with LBM and dual-time stepping ACM has been reported [10]. Weakly compressible fluid simulations have already been successfully applied to flow around complex geometry [11], nested Cartesian grid [12], turbulent flow [13] and two-phase flow [14].

Compared with the case of single phase flow, two-phase flow simulations should consider the large density and viscosity difference across the interface. Moreover, the interface undergoes severe topological deformations, including separation and merging. Various methodologies exist in the past research to distinguish two immiscible fluids and represent the interface, such as the level set method [15], volume of fluid method [16], diffusive interface method [17] and hybrid method [18]. A discussion and comparison of these interface capturing methods can be found in a recent review [19]. Although the computation of Navier-Stokes equations is almost unchanged in the framework of one-fluid model through a coupling with an interface capturing equation [20], it is still challenging for numerical methods in terms of stability and accuracy. The slow convergence of the implicit Poisson solver accounts for a large proportion of execution time, especially when the density gradient of the two-phase flow leads to a non-constant coefficient in the Poisson equation [21]. This fact has motivated researchers to examine explicit weakly compressible Navier-Stokes solvers for two-phase flow simulations.

As a representative weakly compressible fluid model, the lattice Boltzmann method (LBM), which originated from the mesoscale kinetic theory of gases, has been the subject of a growing number of studies on two-phase flows [22–26]. While great progress has been made for large density and viscosity ratios, the stability of LBM for two-phase flows with large Reynolds number is still under investigation. A latest study introduces a filter to the velocity field at the sacrifice of locality of LBM, and it has been shown to work well for violent two-phase flows [26]. However, oscillations in pressure [23] and velocity [25, 26] are observed during the simulations. A comparative study of LBM and ACM has examined these oscillations in terms of acoustic waves [27]. Recently, a weakly compressible, mass and momentum conservative model based on the entropically damped artificial compressibility (EDAC) model is proposed for two-phase flow simulations [28]. It reports that a pressure diffusion term in the pressure evolution equation can reduce the noise in the velocity divergence field. However, this model has to carefully switch off the artificial pressure diffusion in the interfacial region of two-phase flows, but the overshoot of pressure still exists in the vicinity of the interface.

Nevertheless, great difficulties arise when dealing with high density ratio between two phases where the order of 10^3 is an upper limit for many of the existing solvers, typically those that use the Navier-Stokes equations in non-conservative form. The situation be-

comes worse if the motion of the interface is violent. The steep changes of in density and momentum of the fluids across the interface also results in a discontinuity of the pressure gradient, which can be problematic for an unspecialized numerical discretization.

To achieve a robust and accurate simulation of violent two-phase flows with a high density ratio, a consistent formulation of mass and momentum transport is essential [29]. While a staggered grid system is commonly used for the spatial discretization of governing equations, maintaining consistency becomes particularly complex due to the different locations of mass and momentum control volumes. A classical solution is to compute the volume fraction of fluids on a mesh twice as fine as the velocity-pressure staggered mesh. This makes the mass flux through a small control volume directly available for calculating the momentum flux through a large control volume, even if they do not fully coincide. To reduce the computational overhead of solving an advection equation for the volume fraction on sub-cells, some approaches manage to construct auxiliary volume fractions in respective face-centered control volumes of the momentum, either by summing the PLIC-reconstructed values in the sub-cells [30] or by shifting the original volume fractions of two adjacent cells sharing the face by using Weymouth and Yue's advection scheme [31]. While momentum transport can be made consistent with the auxiliary mass, it is not conservative because of the discrepancy in the reconstructed interface. Instead of using the volume fraction equation, another method synchronizes the face-centered density with the level-set function after linearly interpolating the level-set function to the face [32]. Recently, a phase-field model based transport scheme is developed by algebraically modifying the momentum equation taking into consideration the artificial flux term in the phase-field equation [33]. A different realization of consistent transport is presented in [34] where a hybrid staggered/non-staggered grid is employed, and the control volume of the momentum actually occupies two cells associated with the mass. And an identical discrete operator is required for convection of both mass and momentum.

In contrast, a collocated grid is inherently suitable for evaluating convective fluxes of momentum consistently with the mass, because both the density and velocities are defined at the cell center. Since the pioneering work of Bussmann et al. [35], new numerical methods for two-phase flow simulations on a collocated grid, where the consistent and conservative momentum transport can be simply implemented, have been continuously emerging [36–39]. Unlike in the case of a staggered grid, a crucial consideration for ensuring a stable computation on a collocated grid is how to avoid velocity-pressure decoupling. Various techniques, such as the projection filter [40] and double projection [41], have been proposed to enhance the coupling and eliminate numerical modes in the solution. A comparative study has also been reported on the stability and accuracy of pressure projection methods on the collocated grid [42].

1.2 Research Purpose

The two-phase flow solvers discussed above rely primarily on the incompressible Navier-Stokes equations, which require an implicit solution to a pressure Poisson equation. However, iterative methods such as Jacobi, Gauss-Seidel, and SOR are known for their slow convergence rates when applied to two-phase flows with a high density contrast. To address this challenge, Krylov methods and multigrid preconditioners are used to efficiently solve the Poisson equation with variable coefficients [32, 43]. Despite their effectiveness, the complexity of these algorithms presents significant implementation challenges, particularly on large-scale multi-node clusters. Alternatively, the hyperbolic-parabolic system of weakly compressible Navier-Stokes equations offers excellent performance and scalability for large-scale computing on multi-node supercomputers [44]. The simple numerical scheme used for calculating the pressure evolution equation is flexible and can be readily applied to both unstructured and adaptive Cartesian grids, as demonstrated in [45, 46]. This stands in contrast to multigrid-based Poisson solvers, which may be less versatile in their implementation.

Although the weakly compressible Navier-Stokes equations have been successfully applied to two-phase flow simulations [14], several major problems remain. One is the oscillation of the velocity and pressure induced by the acoustic wave in weakly compressible flows. Nevertheless, there has been little research aimed at achieving momentum conservation and consistent transport in weakly compressible method for simulating incompressible two-phase flows. The study based on the EDAC model solves the conservative-form momentum equation in combination with an artificial mass-conservation equation derived from the phase-field model [28]. However, the two-phase flow simulations in that study are limited to moderate density ratios and low Reynolds numbers.

The first purpose of this research is to develop an accurate and robust weakly compressible Navier-Stokes solver for two-phase flow simulations. The divergence-free constraint is not imposed. An evolving pressure projection method is proposed to damp the acoustic wave. Because the overall algorithms have an explicit time integration, excellent computational performance and scalability can be expected on multi-GPU clusters [47, 48].

The second purpose is to develop a weakly compressible Navier-Stokes solver with a consistent and conservative formulation for momentum transport, which is lacking in the research field of weakly compressible method for simulating incompressible two-phase flows to the best of my knowledge. Meanwhile, a novel implementation of the evolving pressure projection method is proposed to damp the acoustic wave, while enforcing the coupling between the pressure and velocities on the collocated grid. The proposed solver aims at providing a robust and accurate computation for violent two-phase flows with

high density ratios. It also takes advantage of the hyperbolic-parabolic characteristics of weakly compressible Navier-Stokes equations for a fully explicit time integration.

The third purpose is to evaluate the performance and scalability of the proposed conservative two-phase flow solvers on the multi-GPU cluster for large-scale parallel computing. The domain partition and communication overlapping techniques are implemented to reduce the overhead and increase the throughput. The accuracy and stability of different numerical schemes are validated by using high-resolution two-phase flow simulations. To further minimize the need for computing resources, the sparse data structures including adaptive mesh refinement (AMR) based on octrees and sparse grid based on spatial hash are implemented with GPU parallel computing.

Chapter 2

Weakly Compressible Navier-Stokes Solver

This chapter first introduces the mathematical model for weakly compressible Navier-Stoke equations, the proposed evolving pressure projection method and two-phase flows. Then the numerical discretization by finite difference method and finite volume method is explained in details. Various single-phase and two-phase flow benchmarks are simulated to validate the present solver.

2.1 Mathematical Model

2.1.1 Fluid Dynamic Equations

The Navier-Stokes equations for fluid flows including the continuity equation and the momentum equation are

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{2.1}$$

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{F}, \tag{2.2}$$

where $\boldsymbol{u} = (u, v, w)$ is the velocity vector, ρ the density, p the pressure and $\boldsymbol{F}$ the external body force. The viscous stress tensor $\boldsymbol{\tau}$ is given by

$$\boldsymbol{\tau} = -\frac{2}{3}\mu \nabla \cdot \boldsymbol{u} \boldsymbol{I} + \mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right), \tag{2.3}$$

where μ is the dynamic viscous coefficient.

Under the assumption of incompressibility, the material derivative of density is constant to 0, i.e.,

$$\frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho = 0. \tag{2.4}$$

The continuity equation is then reduced to the divergence-free condition of the velocity:

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2.5}$$

This condition leads to a Poisson equation that has to be satisfied by the pressure and it be discussed in the next subsection.

Through imposing the isothermal equation of state

$$\frac{dp}{d\rho} = c_s^2 \tag{2.6}$$

and introducing the assumption of small density fluctuation, i.e., $\delta\rho/\rho \ll 1$, which implies the low Mach number constraint [49]:

$$Ma = |\boldsymbol{u}|_{\max} / c_s \ll 1, \tag{2.7}$$

where the c_s is an artificial speed of sound. By expanding Eq. (2.1) with $\delta\rho/\rho$ and substituting $d\rho$ with the Eq. 2.6, it gives

$$\frac{\partial p}{\partial t} + \boldsymbol{u} \cdot \nabla p + \rho c_s^2 \nabla \cdot \boldsymbol{u} = 0. \tag{2.8}$$

Because the fluid velocity is much smaller than the sound speed, the material derivative of pressure can be approximated as

$$\frac{\partial p}{\partial t} + \boldsymbol{u} \cdot \nabla p \approx \frac{\partial p}{\partial t}. \tag{2.9}$$

Finally Eq. (2.8) comes to the original form of artificial compressibility method [50]:

$$\frac{\partial p}{\partial t} + \rho c_s^2 \nabla \cdot \boldsymbol{u} = 0. \tag{2.10}$$

It is worth noting that the entropically damped form of artificial compressibility (EDAC) model [6] and general pressure equation (GPE) [7] contain a pressure diffusion term $\nu_p \nabla^2 p$ in Eq. (2.10) and ν_p is a constant coefficient, which can damp the acoustic wave. In this work, the evolving pressure projection method is proposed to deal with the acoustic wave, see Section 2.1.2. The damping effect is discussed in Section 2.3.1. Since the pressure advection term $\boldsymbol{u} \cdot \nabla p$ in EDAC is negligible, the pressure evolution equation used by us is exactly the same as the original form of the artificial compressibility method.

2.1.2 Evolving Pressure Projection Method

Before explaining the evolving pressure projection method, the original projection method proposed by Chorin [5] to simulate incompressible fluid is briefly recalled. First,

an intermediate velocity $\boldsymbol{u}^*$ is updated from $\boldsymbol{u}^n$ at time step n by solving the momentum equation, Eq. (2.2), without the pressure gradient term,

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\boldsymbol{u}^n \cdot \nabla \boldsymbol{u}^n + \frac{1}{\rho}\nabla \cdot \boldsymbol{\tau} + \frac{1}{\rho}\boldsymbol{F}^n, \tag{2.11}$$

where Δt is a discretized time step. The intermediate velocity needs to be corrected by a pressure projection step, which gives the solution at the next time step $\boldsymbol{u}^{n+1}$:

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1}. \tag{2.12}$$

After taking the divergence of both sides of Eq. (2.12) and enforcing the incompressible condition $\nabla \cdot \boldsymbol{u}^{n+1} = 0$, it becomes a Poisson equation for pressure:

$$\nabla \cdot \left(\frac{1}{\rho}\nabla p^{n+1}\right) = \frac{\nabla \cdot \boldsymbol{u}^*}{\Delta t}. \tag{2.13}$$

In the case of a weakly compressible fluid, the governing equations are obtained from the compressible Navier-Stokes equations, and they are valid when a low Mach number is imposed. Since the divergence-free condition for the velocity field becomes unnecessary, replacing the Poisson equation, Eq. (2.13), with the pressure evolution equation, Eq. (2.10), still has a solid grounding in physics. By taking the divergence of both sides of Eq. (2.12) in derivative form, we have

$$\frac{\partial\left(\nabla \cdot \boldsymbol{u}\right)}{\partial t} = -\nabla \cdot \left(\frac{1}{\rho}\nabla p\right). \tag{2.14}$$

Then, by substituting the velocity divergence term with Eq. (2.10), we arrive at the second-order linear wave equation for pressure:

$$\frac{\partial}{\partial t}\left(\frac{1}{\rho c_s^2}\frac{\partial p}{\partial t}\right) = \nabla \cdot \left(\frac{1}{\rho}\nabla p\right), \tag{2.15}$$

which can be further simplified in the case of a constant density in time and space:

$$\frac{\partial^2 p}{\partial t^2} = c_s^2 \nabla^2 p. \tag{2.16}$$

Because of its hyperbolic characteristics, the wave equation is much easier to treat numerically than the elliptic-type Poisson equation. The evolving pressure projection method is proposed to advance the numerical solutions of velocity and pressure to the next time step. First, the pressure can be easily updated from p^n:

$$\frac{p^{*,0} - p^n}{\Delta t} = -\rho c_s^2 \nabla \cdot \boldsymbol{u}^*. \tag{2.17}$$

By substituting p^{n+1} in Eq. (2.12) with $p^{*,0}$, the intermediate velocity $\boldsymbol{u}^*$ is corrected to $\boldsymbol{u}^{**,0}$. Next, the following iteration process is conducted for N steps:

(i) compute the increment of pressure δp and update the temporary pressure $p^{*,i}$ to $p^{*,i+1}$,

$$\delta p = -\rho c_s^2 \nabla \cdot \boldsymbol{u}^{**,i} \Delta t, \tag{2.18}$$

$$p^{*,i+1} = p^{*,i} + \delta p. \tag{2.19}$$

(ii) update the velocity field $\boldsymbol{u}^{**,i}$ to $\boldsymbol{u}^{**,i+1}$ by projecting the increment of pressure,

$$\frac{u^{**,i+1} - u^{**,i}}{\Delta t} = -\frac{1}{\rho} \nabla \delta p. \tag{2.20}$$

where the superscript i indicates the current step of the iteration and starts from 0. The final solutions of velocity and pressure at step $n+1$ are given by

$$p^{n+1} = p^{*,N}, \tag{2.21}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{**,N}. \tag{2.22}$$

The above method with two iterative steps can be simplified to a new pressure evolution equation as follows:

$$\frac{\partial p}{\partial t} + \rho c_s^2 \left[\nabla \cdot \boldsymbol{u}^* - \nabla \cdot \left(\frac{1}{\rho} \nabla p \right) \Delta t \right] = 0. \tag{2.23}$$

This hyperbolic equation for pressure can be explicitly advanced in time by using a local spatial stencil. It significantly benefits scalability of computation in spite of the speed-of-sound restriction on the time step. It is worth noting that the Eq. (2.23) reduces to the Poisson equation when the speed of sound is infinite or when the evolution of pressure converges with time. This means that in the limit case, the proposed method will converge to the incompressible fluid flow. The assumption of an incompressible fluid actually means an infinite speed of sound, thus, a pressure change anywhere will be transmitted throughout the fluid immediately. As for the weakly compressible fluid under consideration, the above evolving pressure projection method has a physical meaning wherein the perturbation propagates at a finite sound speed for multiple times. This model is expected to alleviate the undesirable effect of the acoustic wave and consequently diminish the fluctuations in the velocity and pressure fields.

2.1.3 Conservative Phase Field Model

Modeling the moving interface accurately is crucial in simulation of two-phase flow. Various numerical methods have been proposed, such as interface tracking by marking

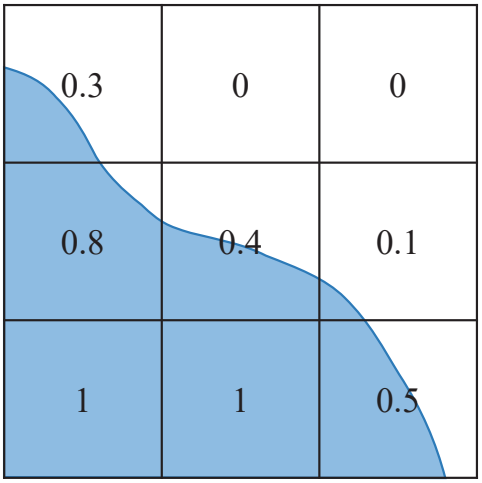


Figure 2.1: Volume fraction of a fluid component in cells.

points [51] and interface capturing by an indicator function of time and space [19]. The interface capturing method has a strong capability to represent large topological changes including separation and merging of the interface, it is more suitable for ubiquitous two-phase flow phenomena in nature and industry.

A widely used one of the interface capturing methods is volume of fluid (VOF) [16], which employs the volume fraction of one component f in a grid cell as the phase indicator function, as illustrated in Fig. 2.1. However, it is numerically more convenient to deal with a diffusive interface than the VOF quantity which has a discontinuity. Derived from the free-energy theory of thin interface, the interface evolution is driven by imbalance of chemical potential to minimize the free-energy function of the system in phase field model. Compared with VOF, the phase field varies smoothly across interface.

A conservative phase field model [52] derived from the Allen-Cahn equation is used in this thesis because it can exactly conserve the mass for incompressible two-phase flow and avoid the computation of forth-order derivative in Cahn-Hilliard equation [53]. The conservative Allen-Cahn equation governing the phase field function ϕ is

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \boldsymbol{u}) = \nabla \cdot \left[M \left(\nabla \phi - \frac{4\phi(1-\phi)}{W} \boldsymbol{n} \right) \right], \tag{2.24}$$

where M is the mobility, W is the width of interface and it is chosen to be $3\Delta x$ in this thesis, $\boldsymbol{n}$ is the unit out-normal vector of interface. The typical values of phase field function are defined as:

$$\phi(\boldsymbol{x}, t) = \begin{cases} 0, & \text{light fluid} \\ 1, & \text{heavy fluid} \\ 0 \sim 1, & \text{interface.} \end{cases} \tag{2.25}$$

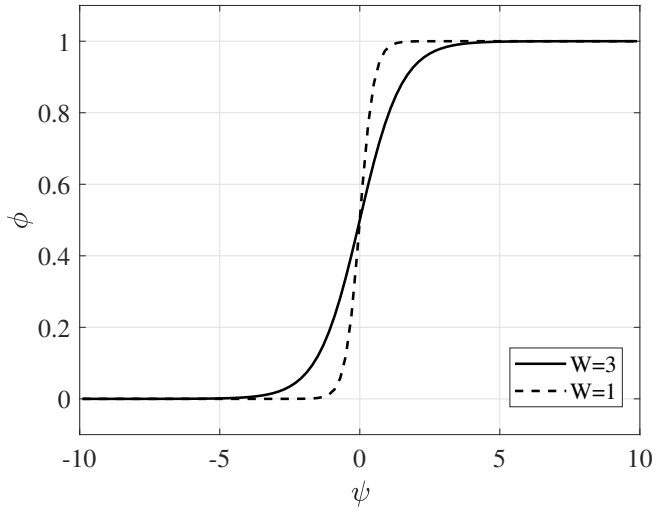


Figure 2.2: Hyperbolic tangent profile of phase field function at equilibrium.

For a interface located at $\boldsymbol{x}_0$, the phase field function has a hyperbolic tangent profile in the equilibrium state:

$$\phi(\boldsymbol{x}) = \frac{1}{2}\left[1 + \tanh\left(\frac{\psi(\boldsymbol{x})}{W/2}\right)\right], \tag{2.26}$$

where $\psi(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}_0$ is a signed distance function of the interface. As shown in Fig. 2.2, the interface profile is smoother by choosing a larger value of W. Mobility M is governed by a non-dimensional Peclet number [54]:

$$\mathrm{Pe} = \frac{U_0 W}{M} \tag{2.27}$$

where U_0 is a typical velocity of the fluid flow, and the value of Peclet number is 10 in this thesis.

The convection-diffusion equation with a source term in conservative phase field model is essentially a one-step conservative level set method [55]. Without the iteration step for reinitialization of the interface profile, the boundedness of ϕ can be preserved by properly choosing the free parameters M and W. It leads to an easy-to-implement method for interface capturing with less computational cost than conservative level set method.

Although the phase field model has a satisfactory capability for interface capturing, the signed distance function ψ, is superior for geometric representation of interface. In order to calculate the interface normal and curvature with a better accuracy, the level set method [56] is introduced as a complementary tool. The time evolution of ψ follows the advection equation:

$$\frac{\partial \psi}{\partial t} + \boldsymbol{u} \cdot \nabla \psi = 0. \tag{2.28}$$

To keep the consistency with phase field function and also to maintain the property of signed distance function, a coupling and re-initialization procedure is performed every

several time steps. To begin with the level set function is converted from phase field function by inverting Eq. (2.26) in a region near the interface:

$$\psi_0(\boldsymbol{x}) = \frac{W}{2}\tanh^{-1}(2\phi(\boldsymbol{x}) - 1) = \frac{W}{4}\ln\left(\frac{\phi(\boldsymbol{x})}{1-\phi(\boldsymbol{x})}\right), \tag{2.29}$$

then a re-initialization equation is solved with enough number of iterations to correct $\psi_0(\boldsymbol{x})$ to a signed distance function,

$$\frac{\partial \psi}{\partial \tau} + S(\psi)\left(|\nabla\psi| - 1\right) = 0, \tag{2.30}$$

where τ is the time step used for iteration and $S(\psi)$ is a smoothed signed function:

$$S(\psi) = \frac{\psi}{\sqrt{\psi^2 + (|\nabla\psi|\,\Delta x)^2}}. \tag{2.31}$$

Finally, the normal vector in Eq. (2.24) and curvature of interface are calculated using the level set function:

$$\boldsymbol{n} = \frac{\nabla\psi}{|\nabla\psi|}, \tag{2.32}$$

$$\begin{aligned}\kappa &= \nabla\cdot\boldsymbol{n} \\ &= \frac{\psi_x^2(\psi_{yy}+\psi_{zz}) + \psi_y^2(\psi_{zz}+\psi_{xx}) + \psi_z^2(\psi_{xx}+\psi_{yy}) - 2(\psi_x\psi_y\psi_{xy} + \psi_y\psi_z\psi_{yz} + \psi_x\psi_z\psi_{xz})}{|\nabla\psi|^3}.\end{aligned} \tag{2.33}$$

2.1.4 One-Fluid Model for Two-Phase Flow

For a two-phase flow system under the framework of one-fluid model, the fluid properties such as density and viscosity coefficient can be evaluated as

$$\rho = \phi\rho_h + (1-\phi)\,\rho_l, \tag{2.34}$$

$$\mu = \phi\mu_h + (1-\phi)\,\mu_l, \tag{2.35}$$

where the physical properties of heavy and light fluids are indicated by the subscripts h and l, respectively.

The external force term in Eq. (2.2) usually consists of two parts for two-phase flow. One is the gravity $\boldsymbol{F}_g = \rho\boldsymbol{g}$, another is the surface tension force $\boldsymbol{F}_{st}$ acting on the interface. A density-scaled continuum surface force (CSF) model [57, 58] is applied in this thesis, $\boldsymbol{F}_{st}$ has the following form:

$$\boldsymbol{F}_{st} = \frac{2\rho}{\rho_h + \rho_l}\sigma\kappa\nabla\phi, \tag{2.36}$$

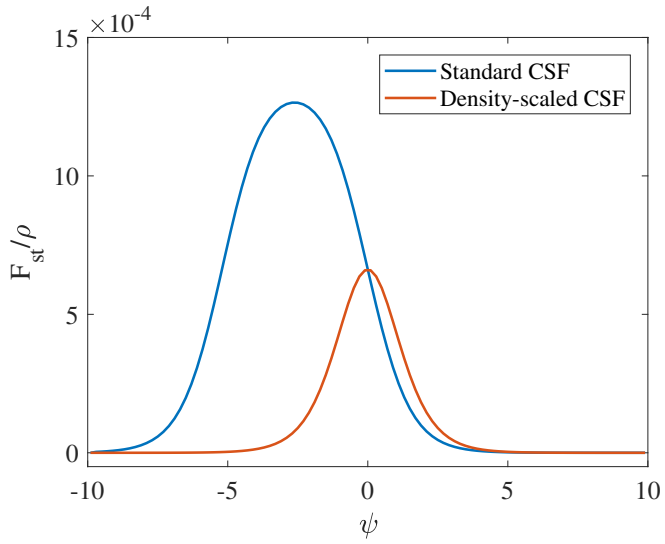


Figure 2.3: Acceleration by surface tension force across two-phase interface.

where σ is the coefficient of surface tension, κ is the curvature of interface. The density scaling coefficient shifts the force to higher density region, resulting in a symmetrical distribution of acceleration across the interface, as shown in Fig. 2.3, where a density ratio $\rho_h : \rho_l = 1000 : 1$ is considered and $\psi = 0$ corresponds to the location of interface.

2.2 Numerical Methods

2.2.1 Spatial Discretization on Staggered Grid

A staggered grid system is used for spatial discretization, as illustrated in Fig. 2.4. It is known that the staggered grid avoids the problem of the decoupling between velocity and pressure, providing higher numerical stability. In a 2-dimensional uniform staggered cell $\left[i-\frac{1}{2}, i+\frac{1}{2}\right] \times \left[j-\frac{1}{2}, j+\frac{1}{2}\right]$, the grid spacing is identical in different directions $\Delta x = \Delta y$. The pressure p, phase field function ϕ and level set function ψ are defined on the center of cell (i, j), surrounded by the velocity components (u, v) on their corresponding cell face. This section presents numerical schemes used to discretize each term of the governing equations, and extension to a 3-dimensional staggered grid is straightforward.

Finite difference method for fluid dynamic equations

The advection term in the momentum equation is crucial for an accurate and robust numerical simulation. Although it is written in non-conservative form in Eq. (2.2), the conservative form will be computed using a standard second-order central difference in the simulation of the singe phase flow, to enable a fair comparison with the results of

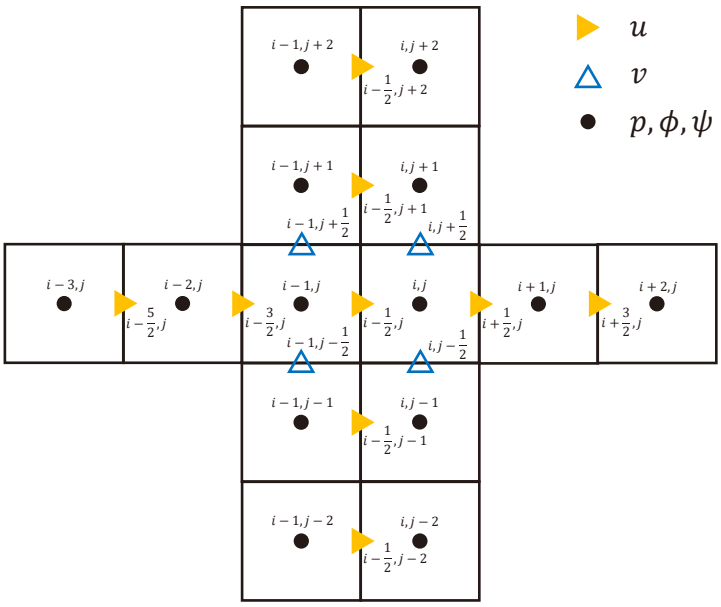


Figure 2.4: Configuration of staggered grid

Toutant [8],

$$\begin{aligned}\nabla \cdot (\boldsymbol{u}u)_{i-\frac{1}{2},j} &= \left(\frac{\partial uu}{\partial x}\right)_{i-\frac{1}{2},j} + \left(\frac{\partial vu}{\partial y}\right)_{i-\frac{1}{2},j} \\ &= \frac{u_{i,j}^2 - u_{i-1,j}^2}{\Delta x} + \frac{\left(v_{i-\frac{1}{2},j+\frac{1}{2}} u_{i-\frac{1}{2},j+\frac{1}{2}}\right)^2 - \left(v_{i-\frac{1}{2},j-\frac{1}{2}} u_{i-\frac{1}{2},j-\frac{1}{2}}\right)^2}{\Delta y},\end{aligned} \tag{2.37}$$

where the undefined velocities at the cell center and cell corner are linearly interpolated from the nearest points, for example:

$$u_{i,j} = \left(u_{i-\frac{1}{2},j} + u_{i+\frac{1}{2},j}\right)/2, \tag{2.38}$$

$$u_{i-\frac{1}{2},j+\frac{1}{2}} = \left(u_{i-\frac{1}{2},j} + u_{i-\frac{1}{2},j+1}\right)/2, \tag{2.39}$$

$$v_{i-\frac{1}{2},j+\frac{1}{2}} = \left(v_{i-1,j+\frac{1}{2}} + v_{i,j+\frac{1}{2}}\right)/2. \tag{2.40}$$

In the simulation of the two-phase flow with a high density and viscosity ratio, especially when the flow becomes violent such as in a breaking wave, the above central difference scheme usually becomes unstable. To enhance numerical stability, the non-conservative form of the advection term is discretized using the third-order WENO (Weighted Essentially Non-Oscillatory) scheme [59],

$$(\boldsymbol{u} \cdot \nabla u)_{i-\frac{1}{2},j} = u_{i-\frac{1}{2},j} \left(\frac{\partial u}{\partial x}\right)^{\text{WENO}}_{i-\frac{1}{2},j} + v_{i-\frac{1}{2},j} \left(\frac{\partial u}{\partial y}\right)^{\text{WENO}}_{i-\frac{1}{2},j}, \tag{2.41}$$

the v component at position $\left(i-\frac{1}{2},j\right)$ is also obtained by linear interpolation:

$$v_{i-\frac{1}{2},j}=\left(v_{i-1,j-\frac{1}{2}}+v_{i,j-\frac{1}{2}}+v_{i-1,j+\frac{1}{2}}+v_{i,j+\frac{1}{2}}\right)/4. \tag{2.42}$$

The third-order WENO scheme requires a 4-point stencil depending on the upwind direction:

$$\left(\frac{\partial u}{\partial x}\right)^{\text{WENO}}_{i-\frac{1}{2},j}=\begin{cases} f\left(u_{i+\frac{1}{2},j},u_{i-\frac{1}{2},j},u_{i-\frac{3}{2},j},u_{i-\frac{5}{2},j}\right), & \text{if } u_{i-\frac{1}{2},j}>0 \\ f\left(u_{i-\frac{3}{2},j},u_{i-\frac{1}{2},j},u_{i+\frac{1}{2},j},u_{i+\frac{3}{2},j}\right), & \text{if } u_{i-\frac{1}{2},j}\leq 0, \end{cases} \tag{2.43}$$

where $f(\cdot)$ is a function to calculate the spatial derivative.

When the density of the fluid is only governed by the phase field model as in Eq. (2.34), the left side of Eq. (2.2) cannot take into account the effect of the density difference near the two-phase interface. If the advection term of the momentum equation is directly solved using the WENO scheme for a non-conservative form, the momentum transport may become inconsistent. This issue was investigated in [32] for density ratios up to 10^6. A density weighted advection scheme is employed to account for the large jump of momentum around the interface. The spatial derivative of velocity in the advection term is replaced by:

$$\frac{\partial u}{\partial x}=\frac{1}{\rho}\left[\frac{\partial\left(\rho u\right)}{\partial x}-u\frac{\partial \rho}{\partial x}\right]. \tag{2.44}$$

Moreover, in the region far from two-phase interface where no density difference exists, the computation on the right side of Eq. (2.44) is equivalent to the left side. The spatial derivatives of momentum ρu and density ρ are computed robustly with the help of the minmod limiter $\Psi(r)=\max\left[0,\min\left(r,1\right)\right]$ to switch between a high-order WENO scheme and a first-order upwind scheme,

$$\varphi_x=\varphi_x^{\text{1st}}+\Psi(r)\left(\varphi_x^{\text{WENO}}-\varphi_x^{\text{1st}}\right), \tag{2.45}$$

where r is the smoothness indicator:

$$r_{i-\frac{1}{2}}=\begin{cases} \frac{\varphi_{i+\frac{1}{2}}-\varphi_{i-\frac{1}{2}}}{\varphi_{i-\frac{1}{2}}-\varphi_{i-\frac{3}{2}}}, & u_{i-\frac{1}{2}}>0 \\ \frac{\varphi_{i-\frac{1}{2}}-\varphi_{i-\frac{3}{2}}}{\varphi_{i+\frac{1}{2}}-\varphi_{i-\frac{1}{2}}}, & u_{i-\frac{1}{2}}\leq 0. \end{cases} \tag{2.46}$$

The viscous stress term of the momentum equation in two-dimensional space can be expanded as

$$\nabla\cdot\left[\mu\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T\right)\right]=\frac{\partial}{\partial x}\begin{bmatrix} 2\mu\frac{\partial u}{\partial x} \\ \mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right) \\ 2\mu\frac{\partial v}{\partial y}\end{bmatrix}, \tag{2.47}$$

and all terms can be discretized by the central difference scheme. Taking the x-direction part as an example, we have

$$\frac{\partial}{\partial x}\left(2\mu\frac{\partial u}{\partial x}\right)_{i-\frac{1}{2},j} = \frac{2\mu_{i,j}\left(\frac{\partial u}{\partial x}\right)_{i,j} - 2\mu_{i-1,j}\left(\frac{\partial u}{\partial x}\right)_{i-1,j}}{\Delta x}, \tag{2.48}$$

$$\frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right]_{i-\frac{1}{2},j} = \frac{\mu_{i-\frac{1}{2},j+\frac{1}{2}}\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)_{i-\frac{1}{2},j+\frac{1}{2}} - \mu_{i-\frac{1}{2},j-\frac{1}{2}}\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)_{i-\frac{1}{2},j-\frac{1}{2}}}{\Delta y}. \tag{2.49}$$

The undefined density and viscosity coefficient at the cell faces and corners are interpolated from the nearest cell-center values in the same fashion as the interpolated velocities in Eq. (2.38)(2.39)(2.40)(2.42).

For the computation of the acoustic part in Section 2.1.2, the central difference scheme is used to discretize the pressure gradient term and velocity divergence term:

$$\left(\frac{\partial p}{\partial x}\right)_{i-\frac{1}{2},j} = \frac{p_{i,j}-p_{i-1,j}}{\Delta x}, \tag{2.50}$$

$$(\nabla\cdot\boldsymbol{u})_{i,j} = \frac{u_{i+\frac{1}{2},j}-u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}}-v_{i,j-\frac{1}{2}}}{\Delta y}. \tag{2.51}$$

The discretization of the surface tension force should be consistent with the pressure gradient term, with the purpose to balance pressure projection and external forces [60, 61],

$$F_{st,(i-\frac{1}{2},j)} = \frac{2\rho_{i-\frac{1}{2},j}}{\rho_h+\rho_l}\sigma\kappa_{i-\frac{1}{2},j}\frac{\phi_{i,j}-\phi_{i-1,j}}{\Delta x}, \tag{2.52}$$

where the curvature at the cell face needs to be interpolated from the cell center $\kappa_{i-\frac{1}{2},j} = \left(\kappa_{i-1,j}+\kappa_{i,j}\right)/2$.

As a supplement, the pressure Poisson equation Eq. (2.13) is solved by using the red/black successive over relaxation (SOR) method, which is suitable for parallel computing [62].

Finite volume method for conservative Allen-Cahn equation

To satisfy the exact mass conservation in two-phase flow simulations, the conservative Allen-Cahn equation is discretized by following the concept of the finite volume method. The advection term in Eq. (2.24) is rewritten in flux form:

$$\nabla\cdot(\boldsymbol{u}\phi)_{i,j} = \frac{\left[(u\phi)_{i+\frac{1}{2},j}-(u\phi)_{i-\frac{1}{2},j}\right]\Delta y + \left[(v\phi)_{i,j+\frac{1}{2}}-(v\phi)_{i,j-\frac{1}{2}}\right]\Delta x}{\Delta x\Delta y}. \tag{2.53}$$

The phase field functions inside the advection flux are reconstructed by a third-order MUSCL (Monotone Upstream-centered Schemes for Conservation Law) scheme [63] with the minmod limiter.

In contrast, a piecewise linear reconstruction is used to calculate the phase field function $\phi_{i-\frac{1}{2},j} = (\phi_{i-1,j} + \phi_{i,j})/2$ for the anti-diffusion flux $\frac{4\phi(1-\phi)}{W}\boldsymbol{n}$. The component of the normal vector on a cell face is calculated using the level set function,

$$n_{x,\left(i-\frac{1}{2},j\right)} = \frac{\psi_{i,j} - \psi_{i-1,j}}{\Delta x} \tag{2.54}$$

with the assertion of the signed distance property $|\nabla\psi| = 1$. The diffusion flux $\nabla\phi$ is discretized using a second-order central difference scheme, which is almost identical to the scheme for the viscous stress term of the momentum equation.

In addition, the numerical methods in [64] is followed and the WENO scheme and Godunov's scheme are used to solve the advection equation and re-initialization equation of the level set function.

2.2.2 Time Integration

A three-stage third-order strong stability-preserving Runge-Kutta (SSP-RK-3) scheme [65] is applied to all temporal derivatives, except for those in the evolving pressure projection process where a first-order Euler scheme is employed. For a general time-marching partial differential equation of a variable q,

$$\frac{\partial q}{\partial t} = L(q), \tag{2.55}$$

where $L(\cdot)$ is an operator relying on q, the SSP-RK-3 scheme uses the following formulation to update q^{n+1} from q^n:

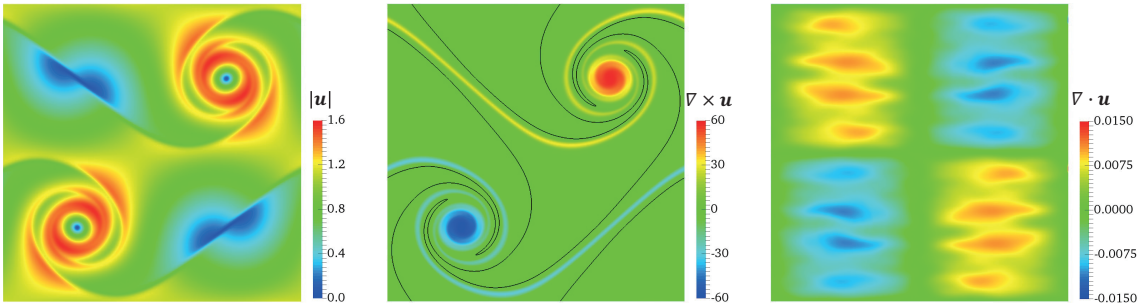
$$q^{(1)} = q^n + L\left(q^n\right)\Delta t, \tag{2.56a}$$

$$q^{(2)} = \frac{3}{4}q^n + \frac{1}{4}q^{(1)} + \frac{1}{4}L\left(q^{(1)}\right)\Delta t, \tag{2.56b}$$

$$q^{n+1} = \frac{1}{3}q^n + \frac{2}{3}q^{(1)} + \frac{2}{3}L\left(q^{(2)}\right)\Delta t. \tag{2.56c}$$

The time step Δt in the present weakly compressible Navier-Stokes solver for two-phase flow simulations is constrained by advection, propagation of the acoustic wave, viscous stress, surface tension, and mobility in the phase field model. Thus, the following conditions should be satisfied [14]:

$$\Delta t = \min\left(\mathrm{CFL}_{\mathrm{adv}}\frac{\Delta x}{|\boldsymbol{u}|_{\max}}, \mathrm{CFL}_{\mathrm{acs}}\frac{\Delta x}{c_s}, s_{\mathrm{visc}}\frac{\Delta x^2}{2N_d\max\left(\frac{\mu_h}{\rho_h}, \frac{\mu_l}{\rho_l}\right)}, s_{\mathrm{sf}}\sqrt{\frac{\rho_h+\rho_l}{2}\frac{\Delta x^3}{2\pi\sigma}}, s_{\mathrm{pf}}\frac{\Delta x^2}{2dM}\right), \tag{2.57}$$



(a) Velocity magnitude (b) Vorticity (c) Velocity divergence

Figure 2.5: Doubly periodic shear layers at $t = 1$ solved through $\boldsymbol{u}^{n+1} = L\left(\boldsymbol{u}^n, p^n\right)$ and $p^{n+1} = L\left(\boldsymbol{u}^n\right)$ without subiteration.

where N_d is the number of spatial dimensions. Safe coefficients $\mathrm{CFL}_{\mathrm{adv}} = \mathrm{CFL}_{\mathrm{acs}} = 0.4$, $s_{\mathrm{visc}} = s_{\mathrm{sf}} = s_{\mathrm{pf}} = 0.1$ are used in the simulations described below.

2.3 Numerical Results

2.3.1 Doubly Periodic Shear Layers

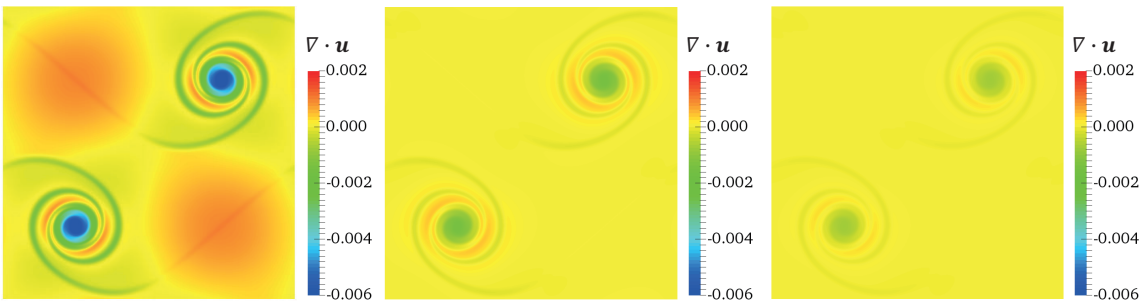
The first validation involves a single-phase flow problem from the reference [8]. In a periodic unit square, the doubly periodic shear layers have initial velocities as follows:

$$u = \begin{cases} \tanh\left[k\left(y - 0.25\right)\right], & y < 0.5 \\ \tanh\left[k\left(0.75 - y\right)\right], & \text{otherwise,} \end{cases} \tag{2.58a}$$

$$v = \delta \sin\left[2\pi\left(x + 0.25\right)\right], \tag{2.58b}$$

where the parameters $k = 80$ and $\delta = 0.05$ are used. To reproduce the results in [8], the same computational settings as in that study are used: density of fluid $\rho = 1$, viscous coefficient $\mu = 10^{-4}$, Mach number $Ma = 0.02$ resulting in a speed of sound $c_s = 50$, time step $\Delta t = 10^{-5}$, and uniform grid with 512×512 cells. A second-order central difference is used to discretize the advection term in the momentum equation, as mentioned in Section 2.2.1. Fig. 2.5 shows the velocity magnitude, vorticity, and velocity divergence field at $t = 1$ when solving the weakly compressible Navier-Stokes equations with a traditional procedure $\boldsymbol{u}^{n+1} = L\left(\boldsymbol{u}^n, p^n\right)$ and $p^{n+1} = L\left(\boldsymbol{u}^n\right)$. The results are almost the same as in [8] when the diffusion coefficient of pressure is 0.

Next, the evolving pressure projection method is applied to this computation. By increasing the number of iterations from 2 to 10 and 20, the magnitude of the velocity divergence decreases accordingly. Moreover, the velocity divergence caused by the acoustic wave in Fig. 2.5(c) is completely removed. Moreover, only the vortices of the shear



(a) 2 iterations (b) 10 iterations (c) 20 iterations

Figure 2.6: Velocity divergence of doubly periodic shear layers at $t = 1$ solved by evolving pressure projection method.

Table 2.1: L2 norm of velocity error at $t = 0.1$ in 2D Taylor-Green vortices.

Iterations	2	5	10	20	50	100
L2 error $[\times 10^{-5}]$	5.59	4.45	3.79	3.32	3.07	3.00

layers appear in the velocity divergence field (Fig. 2.6). The proposed method thus has a stronger effect of damping the sound wave and reducing the compressibility of the fluid.

2.3.2 Taylor-Green Vortices

In order to investigate how the present method affects the solution for the unsteady flows, the two-dimensional Taylor-Green vortices problem in a periodic unit square is computed with different numbers of iterations. Analytical solutions of velocities and pressure are

$$u = \cos(2\pi x)\sin(2\pi y)\, e^{-\frac{8\pi^2}{Re}t}, \tag{2.59a}$$

$$v = -\sin(2\pi x)\cos(2\pi y)\, e^{-\frac{8\pi^2}{Re}t}, \tag{2.59b}$$

$$p = -\frac{1}{4}\left(\cos(4\pi x) + \cos(4\pi y)\right) e^{-\frac{16\pi^2}{Re}t}. \tag{2.59c}$$

The Reynolds number is set to be 10. The speed of sound is given by $c_s = U_0/Ma = 1/0.05 = 20$. Computation is performed on a 128×128 uniform grid. The L2 norm of velocity error against analytical solutions at time $t = 0.1$ is listed in Table 2.1. It shows a clear tendency that the error gradually decreases as the number of iterations increases. 10 iterations have made the error quite small, and the improvement is not significant when the number of iterations is increased to 100.

2.3.3 2D Static Droplet

A two-phase flow problem of two-dimensional static droplet is studied next. In this problem, a circular droplet of diameter $D = 1$ is placed at the center of a 2×2 domain. The density of the liquid droplet is $\rho_h = 1000$, and the density of the surrounding gas is $\rho_l = 1$. Both the droplet and the gas are inviscid, i.e., $\mu_h = \mu_l = 0$. This problem is gravity-free, and only surface tension force is applied. The coefficient of surface tension is $\sigma = 1$. Hence, the exact pressure difference between inside and outside the droplet can be calculated with the Young-Laplace equation

$$\Delta p_{\text{exact}} = \sigma\kappa = \frac{\sigma}{D/2} = 2. \tag{2.60}$$

In this computation, the domain is discretized by a 128×128 uniform grid. The pressure field is initialized as $p_0(\boldsymbol{x}) = \Delta p_{\text{exact}}\phi(\boldsymbol{x})$. The speed of sound is set to $c_s = 5$, and the time step is $\Delta t = 1.25 \times 10^{-3}$. In this simulation and the others reported below, the advection term in the momentum equation is computed using the density weighted advection scheme. First, the proposed evolving pressure projection method is compared with the traditional procedure for solving weakly compressible Navier-Stokes equations, described in the previous section. The numbers of iterations for the evolving pressure projection are 2 and 10. The pressure profiles over the horizontal line crossing the center of the droplet are plotted in Fig. 2.7 after 30 time steps. While the traditional method produces the largest oscillation in pressure, the present method suppresses the oscillation by increasing the number of iterations. Moreover, the computed pressure difference matches the theoretical value.

The velocity fields after 500 time steps are shown in Fig. 2.8. A spurious current near the interface (black line in the figure) has been reported in numerous studies [66]. This current is caused by the pressure not being in balance with surface tension force. In addition to this spurious current, the traditional method gives a chaotic velocity field, which is due to the acoustic wave. In contrast, the evolving pressure projection method removes almost all of the acoustic wave. Moreover, it makes the magnitude of the spurious current comparable with the value obtained by solving the Poisson equation. When attempting to damp the spurious current by introducing the viscous effect, the magnitude of velocity fluctuations doesn't decrease as the results of other studies [36, 38] even after long-time simulations in the viscous time scale. This suggests that further improvements can focus on the exact balance between the surface-tension force and pressure gradient and also the reduction of discretization errors.

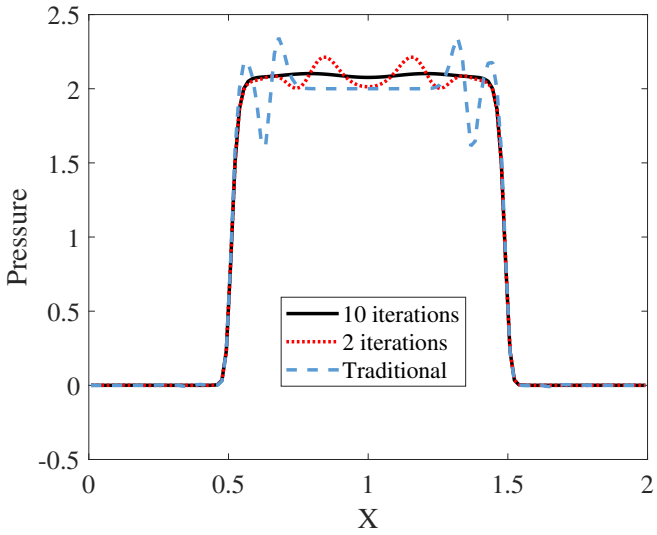
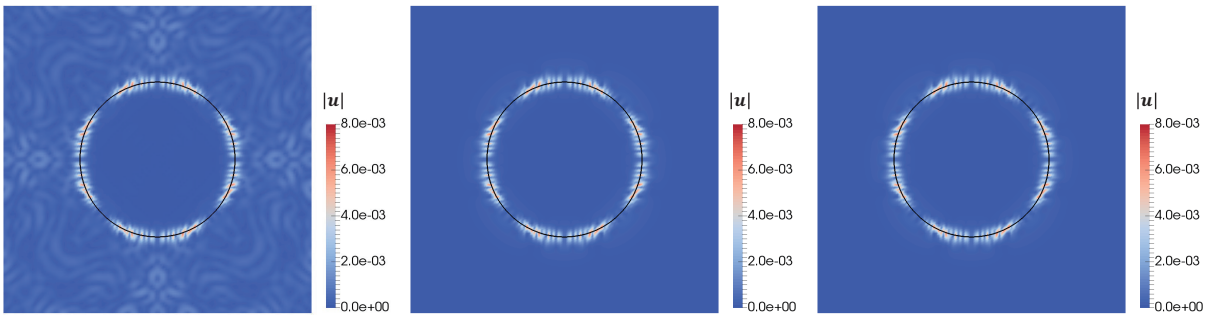


Figure 2.7: Comparison of pressure profiles over the line crossing the center of droplet after 30 time steps.



(a) Traditional (b) 10 iterations (c) Incompressible

Figure 2.8: Comparison of magnitudes of spurious current after 500 time steps in 2D static droplet.

2.3.4 2D Single Rising Bubble

As illustrated in Fig. 2.9, a two-dimensional bubble rising benchmark is simulated to verify the accuracy of the proposed method for two-phase flows. In this simulation, a circular bubble is surrounded by liquid at rest. Two groups of physical properties are examined (Table 2.2). The no-slip conditions are imposed on the top and bottom enclosing walls, while slip conditions are imposed on the left and right sides.

First, the mesh dependency is checked using 128×256 and 256×512 grids. A typical velocity, $U_0 = 0.5$ and Mach number $Ma = 0.1$ are chosen. Thus, the speed of sound in this computation is $c_s = 5$. With 10 iterations for the evolving pressure projection, the shapes of the bubble at $t = 3$ in case 1 and case 2 are shown in Fig. 2.10. While the low- and high-resolution grids both generate the same shape for case 1, the result of the

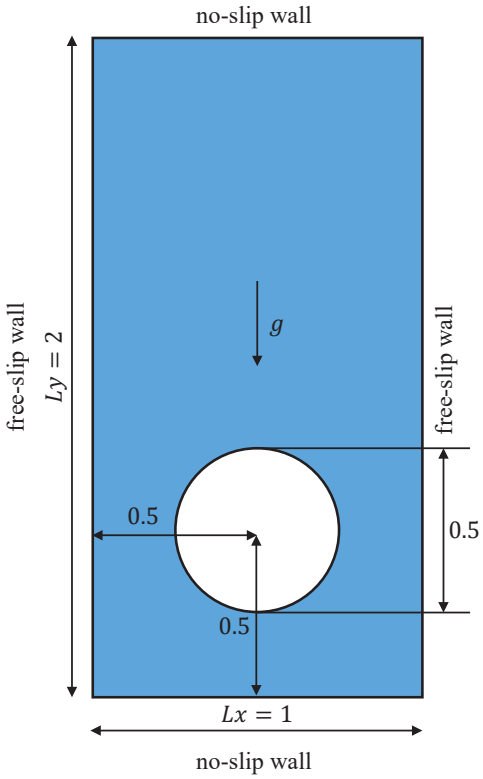


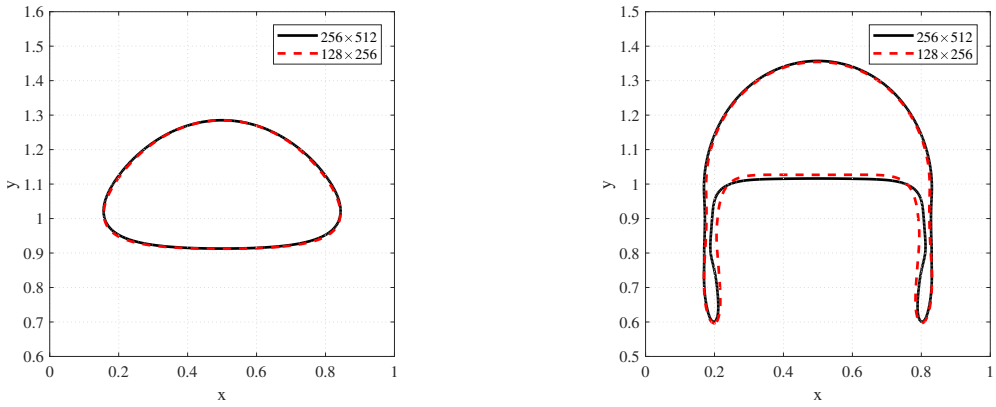
Figure 2.9: Schematic diagram of 2D bubble rising problem.

Table 2.2: Physical properties of 2D bubble rising problem.

Case	ρ_h	ρ_l	μ_h	μ_l	g	σ
1	1000	100	10	1	0.98	24.5
2	1000	1	10	0.1	0.98	1.96

higher resolution grid is in better agreement with the result in [67] for case 2. Therefore, 256×512 grid is used in the subsequent computations.

Initialization of the pressure field is another important issue for weakly compressible two-phase simulations. As we know, the pressure gradient in the static state is balanced with the gravitational force and surface tension force. However, sometimes it is difficult to generate such a pressure field artificially. If the pressure is simply set to $\nabla p = \rho \boldsymbol{g}$ where the density ρ is calculated via the phase field function by Eq. 2.34, and the evolving pressure projection method is not applied, the time variation of rising velocity in case 1 is plotted in Fig. 2.11. The benchmark data are those of Hysing et al.[68] (finite element discretization and level set method) and those of Aland and Voigt [67] (finite element discretization and Cahn-Hilliard model). Because of the initially unbalanced pressure and relatively large Mach number, the acoustic wave causes an obvious oscillation in the bubble rising velocity. In order to eliminate the impact of initial pressure field, the



(a) Case 1 (b) Case 2

Figure 2.10: Bubble shapes at $t = 3$ on 128×256 and 256×512 grids.

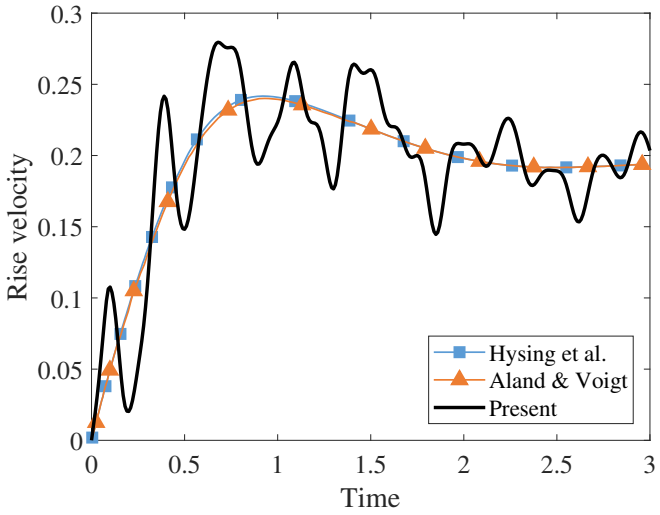
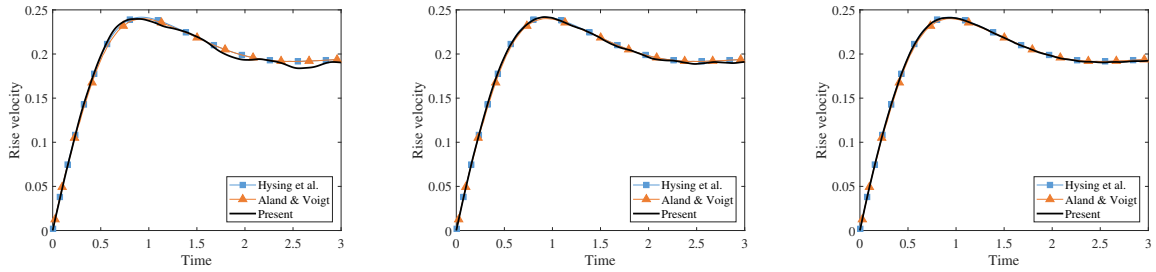


Figure 2.11: Rising velocity over time in case 1 with initial pressure $\nabla p = \rho \boldsymbol{g}$ and no iteration.

pressure Poisson equation is solved for 5 time steps at the beginning of the simulation. Alternatively, the proposed evolving pressure projection method can be used with enough iterations at the beginning.

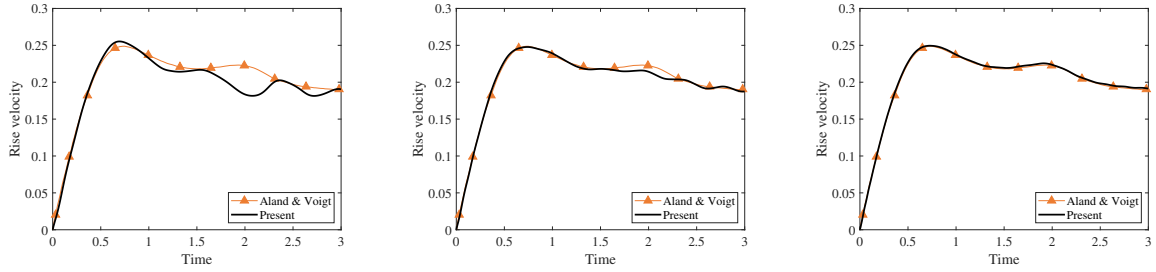
Through above investigations, the final results on the evolution of rising velocity in case 1 and case 2 are plotted in Fig. 2.12 and Fig. 2.13. The velocity oscillation is suppressed and tends to converge as the number of iterations for evolving pressure projection increased. The results obtained using 10 iterations are in good agreement with the Aland and Voigt benchmark [67] in both cases. This verifies that the present method has sufficient accuracy.

The present weakly compressible flow solver is proposed as an alternative to those



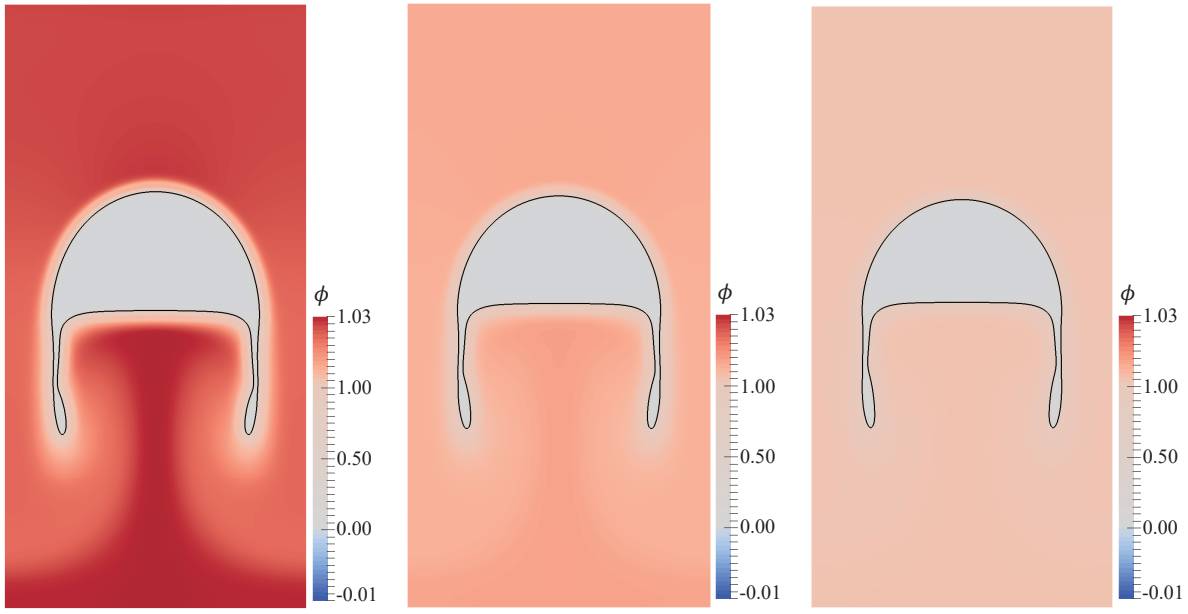
(a) 0 iteration (b) 2 iterations (c) 10 iterations

Figure 2.12: Rising velocity over time in case 1 with different number of iterations.



(a) 0 iteration (b) 2 iterations (c) 10 iterations

Figure 2.13: Rising velocity over time in case 2 with different number of iterations.



(a) 0 iteration (b) 2 iterations (c) 10 iterations

Figure 2.14: Phase field profile emphasized for $\phi < 0$ and $\phi > 1$ in case 2 with different number of iterations.

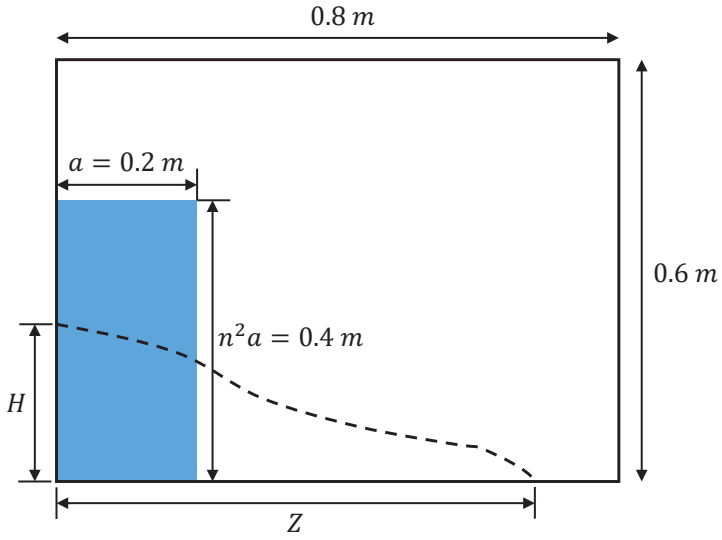


Figure 2.15: Schematic diagram of 2D dam break problem.

incompressible Navier-Stokes solvers where the pressure Poisson equation should be implicitly treated. The purpose is to make the explicit computation for weakly compressible N-S equations as close as possible to the incompressible flow limit. Therefore, the simulation and experimental results of incompressible two-phase flows are used as a comparison of the present method. Compressibility effects still exist in the simulation, as depicted in Fig. 2.14, the region with phase field value $\phi > 1$ and $\phi < 0$ is emphasized. This result shows the compressibility of fluids also decreases with increased number of iterations.

2.3.5 2D Dam Break

A two-dimensional dam break problem is solved to demonstrate the effect of the proposed method on the velocity divergence field, since this sort of simulation shows violent flows and complex two-phase interfaces. Following the experimental settings of Hu and Sueyoshi [69], the initial water column has a width $a = 0.2$ m and height $n^2a = 0.4$ m, located at the left and bottom of a rectangular tank of size 0.8 m $\times$ 0.6 m, as illustrated in Fig. 2.15. The physical properties of the water and air are $\rho_h = 998$ kg/m^3, $\rho_l = 1.2$ kg/m^3, $\mu_h = 1.0 \times 10^{-3}$ Pa·s, $\mu_l = 1.8 \times 10^{-5}$ Pa·s, and surface tension coefficient $\sigma = 0.07275$ N/m. The gravitational acceleration is $g = 9.8$ m/s^2. The computational domain is discretized on a 400 $\times$ 300 grid. The speed of sound is set as $c_s = U_0/Ma = 7/0.05 = 140$ m/s. No-slip conditions are imposed on the enclosing walls.

In addition to the experimental results, the results of an incompressible solver (solve pressure Poisson equation until it converges to $|\nabla \cdot \boldsymbol{u}| \sim O(10^{-6})$) are included as a numerical reference. The field of the phase field function ϕ is depicted in Fig. 2.16. It shows good agreement with the three-dimensional experiment, although the simulation

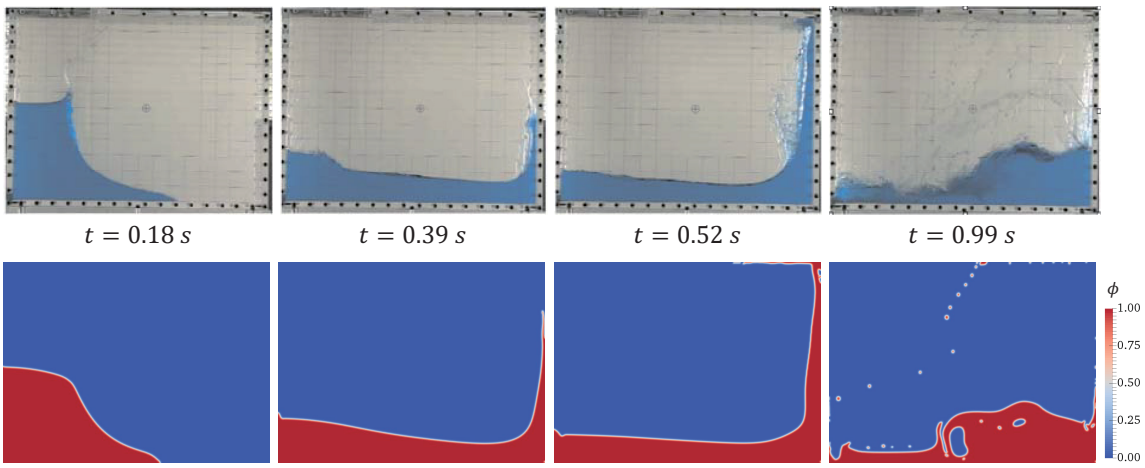


Figure 2.16: Snapshots of water profile in 2D dam break. Experimental [69] (top) and numerical results of incompressible solver (bottom).

is two dimensional. Then, the proposed method with 5, 20, and 100 iterations is used to solve this problem. The velocity divergence fields are shown in Fig. 2.17 where the water surface is represented by the black line. While the interface is in about the same position as the one found by the incompressible solver, the magnitude of velocity divergence is greatly reduced by increasing the number of iterations. The pressure variation is also monitored at the point $(0.8, 0.1)$ on the right wall. Less oscillations of pressure can be observed in Fig.2.18 when the number of iterations increases. It proves that the variables related to compressibility effects will converge with the number of iterations.

In Fig. 2.19, the height and front position of the water column are benchmarked against the experimental data from [70] and [69]. Only the results obtained using 20 iterations are plotted because there is almost no difference among the results of the weakly compressible solver with different iteration numbers and the incompressible solver within the time period of the simulation. The non-dimensionalized time τ, T, height H and front position Z are defined by

$$\tau = t\sqrt{\frac{g}{a}}, \quad T = n\tau, \quad H = \frac{h}{n^2 a}, \quad Z = \frac{z}{a}. \tag{2.61}$$

The simulated height matches the experimental values, whereas the water front advances a little faster in this simulation. Perhaps, this subtle discrepancy can be explained by the neglect of the sidewall resistance in the two-dimensional simulation.

To quantitatively investigate the dependence of the required number of iterations on Mach number, the convergence criterion is set as $|\nabla \cdot \boldsymbol{u}|_{\max} < 0.1$ and test two cases with $Ma = 0.1$ and 0.05. The number of iterations over time is plotted in Fig. 2.20. It is reasonable that several peak values appear when the water accelerates rapidly and hits the right and top walls. Under the restriction of the stability condition Eq. (2.57), a

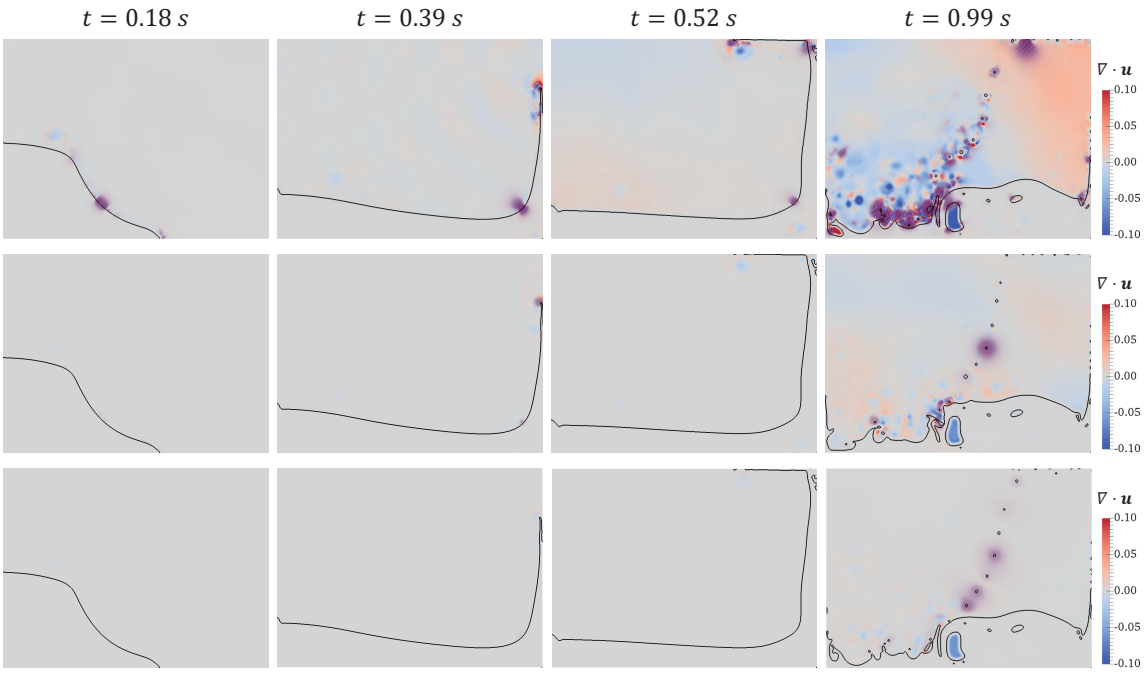
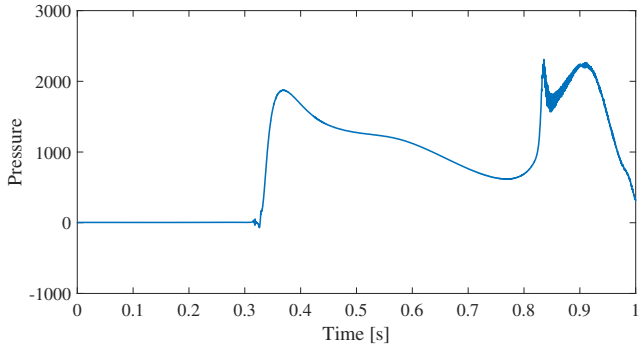


Figure 2.17: Velocity divergence in 2D dam break solved by present method with 5 iterations (top), 20 iterations (middle) and 100 iterations (bottom).

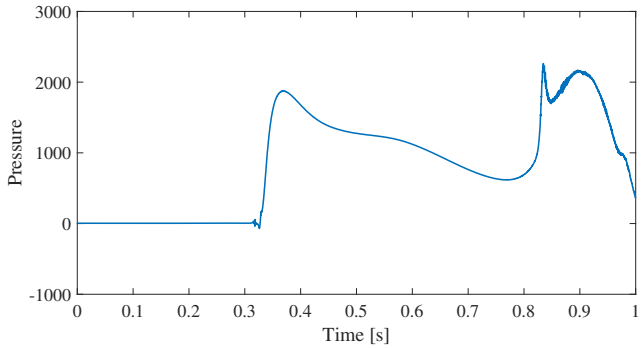
larger time step Δt can be used to shorten the simulation time in the case of a higher Mach number $Ma = 0.1$. However, the required number of iterations becomes much larger if the compressibility error must be small.

Then the order of convergence of the present method is investigated at various situations of two-phase flows, the flow data of this 2D dam break simulation at time $t = 0.18$ s, 0.39 s, 0.52 s and 0.99 s is selected as the initial conditions, corresponding to the snapshots in Fig. 2.16. Only one time step of computation is performed with same Δt by using present method for solving the pressure evolution equation and several classical techniques for solving pressure Poisson equation. The residual of iteration is defined by $\max |p^{*,i+1} - p^{*,i}|$. In all four cases shown in Fig. 2.21, the convergence rate of present method is slower than the iterative methods for Poisson equation. However, it is well known that the implicit computation of Poisson equation cannot take great advantage of massively parallel computing on GPU, even if more sophisticated techniques such as multi-grid method are implemented. In contrast, the present method benefits from the explicit time integration and a local stencil, which make it inherently suitable for GPU computing. Moreover, weakly compressible flow solver allows larger tolerance of velocity divergence, and a few iterations of the present method are enough to suppress the compressibility effects.

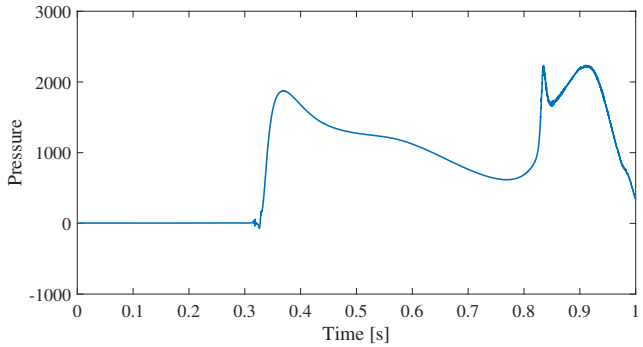
Finally, a test of the computational efficiency is conducted by using this 2D dam break problem. Mach number is set to be 0.05. The time step size Δt in weakly compressible



(a) 5 iteration

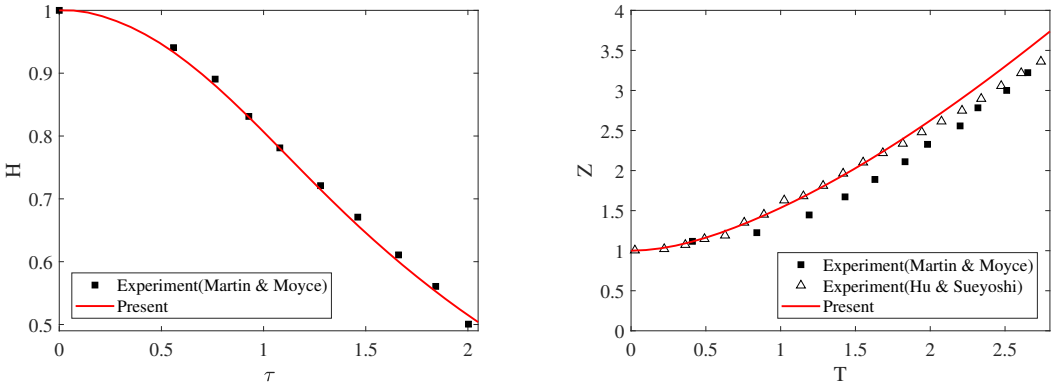


(b) 20 iterations



(c) 100 iterations

Figure 2.18: Time variation of pressure on the right wall of 2D dam break problem with different number of iterations.



(a) Height of water column (b) Front position

Figure 2.19: Time variation of height and front position in 2D dam break.

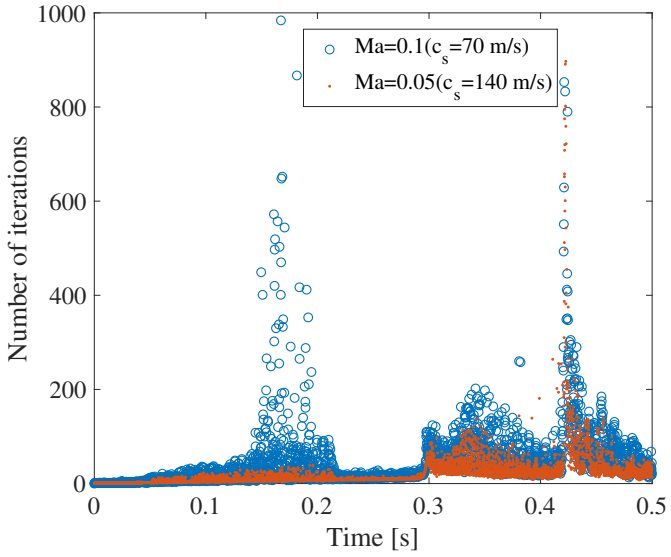
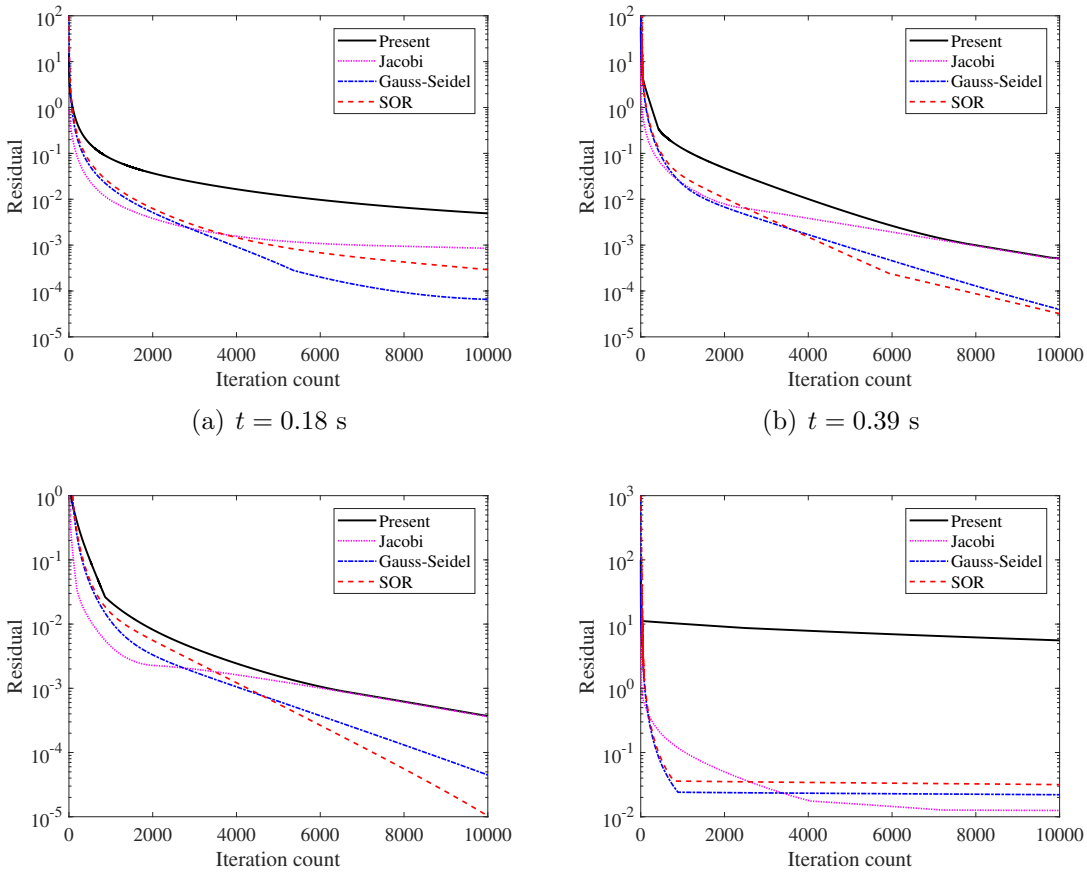


Figure 2.20: Number of iterations to satisfy $|\nabla \cdot \boldsymbol{u}|_{\max} < 0.1$ when $Ma = 0.1$ and 0.05.

flow solver including GPE [8] and the present evolving pressure projection (EPP) methods is governed by the speed of sound, but governed by the mobility of the phase-field model in the incompressible flow solver with SOR method. The time to solution of physical time $t = 1$ s on a Tesla V100 GPU is listed in Table 2.3. The present method takes relatively longer time than GPE when 5 iterations are applied. But the computational efficiency with 100 iterations is still much higher than the incompressible solver with SOR method. Scalability on multiple GPUs of the present method is also promising, but it is remained to be studied in future research.



(a) $t = 0.18$ s (b) $t = 0.39$ s

(c) $t = 0.52$ s (d) $t = 0.99$ s

Figure 2.21: Comparison of the convergence rate of the present method with iterative methods for Poisson equation.

Table 2.3: Comparison of time to solution using weakly compressible and incompressible flow solvers in 2D dam break problem.

Method	GPE	EPP			SOR
		5 iterations	20 iterations	100 iterations	
Δt [$\times 10^{-6}$]	7.14	7.14	7.14	7.14	47.62
Time steps	140000	140000	140000	140000	21000
Computational time [s]	111.67	122.85	162.68	379.83	1653.55

2.3.6 3D Oblique Coalescence of Two Bubbles

The capability of the present solver to accurately produce two-phase interface behavior is verified by simulating bubble dynamics in a real three-dimensional space. The phenomena of oblique coalescence of two bubbles was studied experimentally by Brereton and Korotney [71] and numerically by Annaland et al. [72]. In this simulation, two spherical gas bubbles of diameter $D = 0.01$ m are initially placed in a quiescent liquid, enclosed by a $4D \times 4D \times 8D$ domain. As illustrated in Fig. 2.22, the centers of the bubbles are located at $(2D \times 2D \times 2.5D)$ and $(2.8D \times 2D \times D)$. 20 grids are assigned to the diameter D. The physical properties of the bubble dynamics are determined by the dimensionless Morton number and Eötvös number:

$$Mo = \frac{g\mu_h^4 \left(\rho_h - \rho_l\right)}{\rho_h^2 \sigma^3} = 2 \times 10^{-4}, \tag{2.62a}$$

$$Eo = \frac{\left(\rho_h - \rho_l\right) g d^2}{\sigma} = 16. \tag{2.62b}$$

The viscous coefficients and surface tension coefficient are determined with a density ratio of 100, liquid and gas densities of $\rho_h = 100$ kg/m^3 and $\rho_l = 1$ kg/m^3 and gravitational acceleration of $g = 9.8$ m/s^2. As in the 2D bubble rising problem in Section 2.3.4, the speed of sound is $c_s = U_0/Ma = 0.5/0.1 = 5$ m/s with 10 iterations for evolving pressure projection. All walls have free-slip conditions.

The time evolutions of bubbles shapes are depicted in Fig. 2.23. The leading bubble ascends upwards until the trailing bubble catches up. Meanwhile, the trailing bubble undergoes a large deformation as it rises obliquely, and it eventually coalesces into the leading bubble. The whole process is in good agreement with the results of previous studies [71, 72].

2.3.7 3D Dam Break on a Wet Bed

To demonstrate the stability of the present method as a practical two-phase flow solver, the problem of a three-dimensional dam break on a wet bed is solved. Experimental studies on this problem have been reported by Stansby et al. [73] and Jánosi et al. [74]. Numerical simulations of various dam break problems have also been reported [75]. The water tank considered measures 0.72 m $\times$ 0.12 m $\times$ 0.36 m and is discretized by a $576 \times 96 \times 288$ uniform grid. Enclosed by a virtual baffle plate located at $x = 0.15$ m, a water column with the same width and height is placed at the left side of the tank, as shown in Fig. 2.24. The depth of the water on the flat bed is 0.018 m. Initially the water and air are at rest. The same fluid properties as in the 2D dam break problem in Section 2.3.5 are used in this computation. The speed of sound is $c_s = U_0/Ma =$

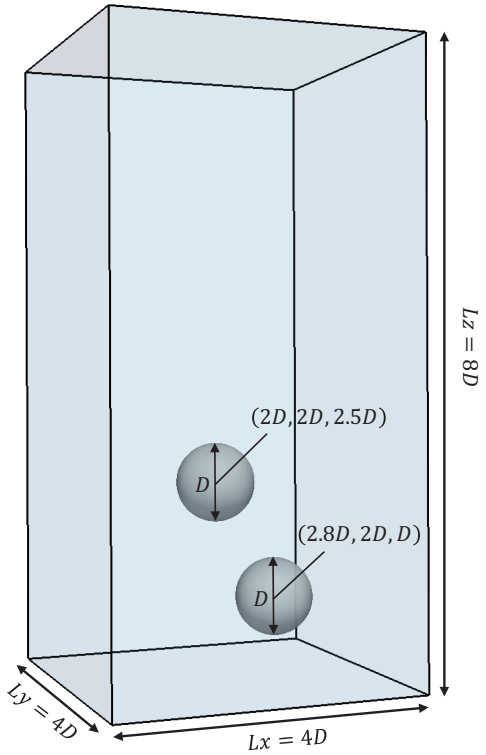


Figure 2.22: Schematic diagram of 3D oblique coalescence of two bubbles problem.

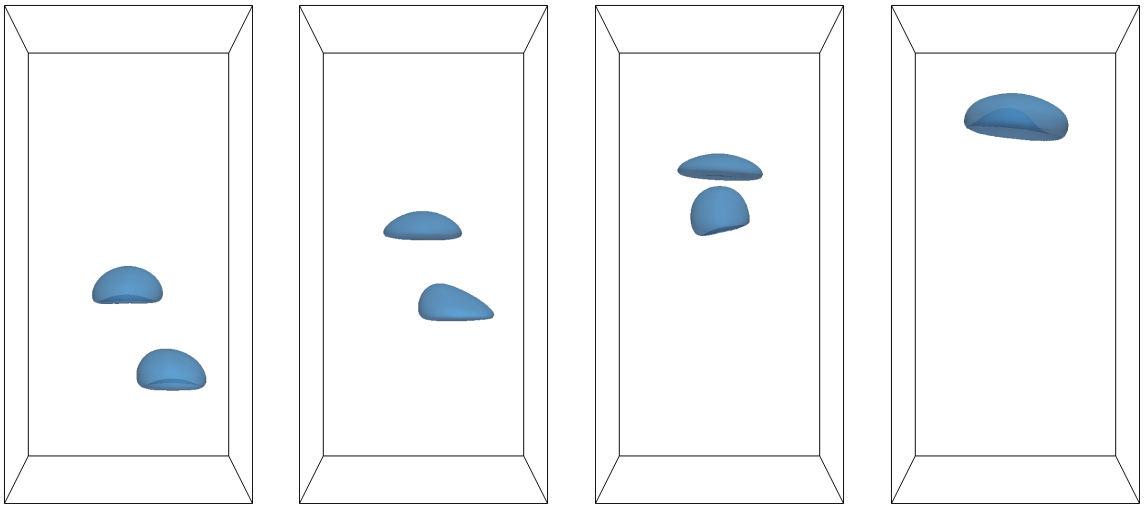


Figure 2.23: Time evolution of bubbles in 3D oblique coalescence of two bubbles.

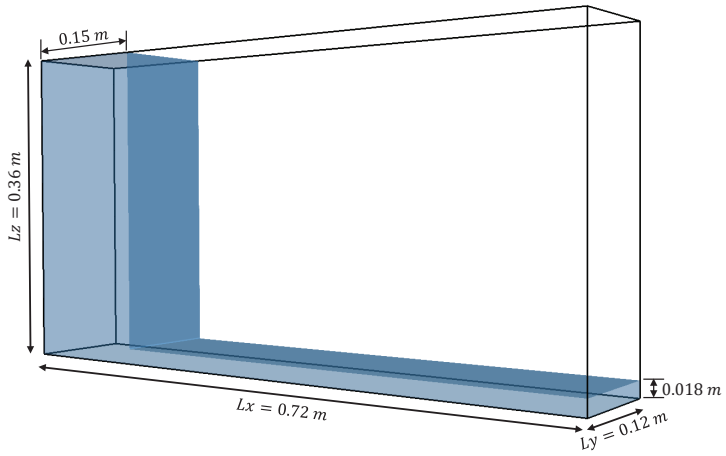


Figure 2.24: Schematic diagram of 3D dam break on a wet bed problem.

$10/0.1 = 100$ m/s and 20 iterations for evolving pressure projection are performed. No-slip conditions are imposed on all walls.

The time evolution of the water surface till 1.2 seconds is depicted in Fig. 2.25. The baffle plate is abruptly removed at the beginning of the simulation, and the water column begins to collapse under gravity. A rolling surface of water over the wet bed appears at $t = 0.2$ s and air bubbles mix into the water at $t = 0.4$ s. After the water crashes against the downstream wall, a lot of droplets splash out at $t = 0.6$ s. As the water body withdraws from the wall, the droplets in the air are integrated into the water. The results obtained by the present method show a similar tendency to those of the Navier-Stokes solver based on the characteristic method [14] and the filtered lattice Boltzmann method [26]. Overall, the computation is very stable.

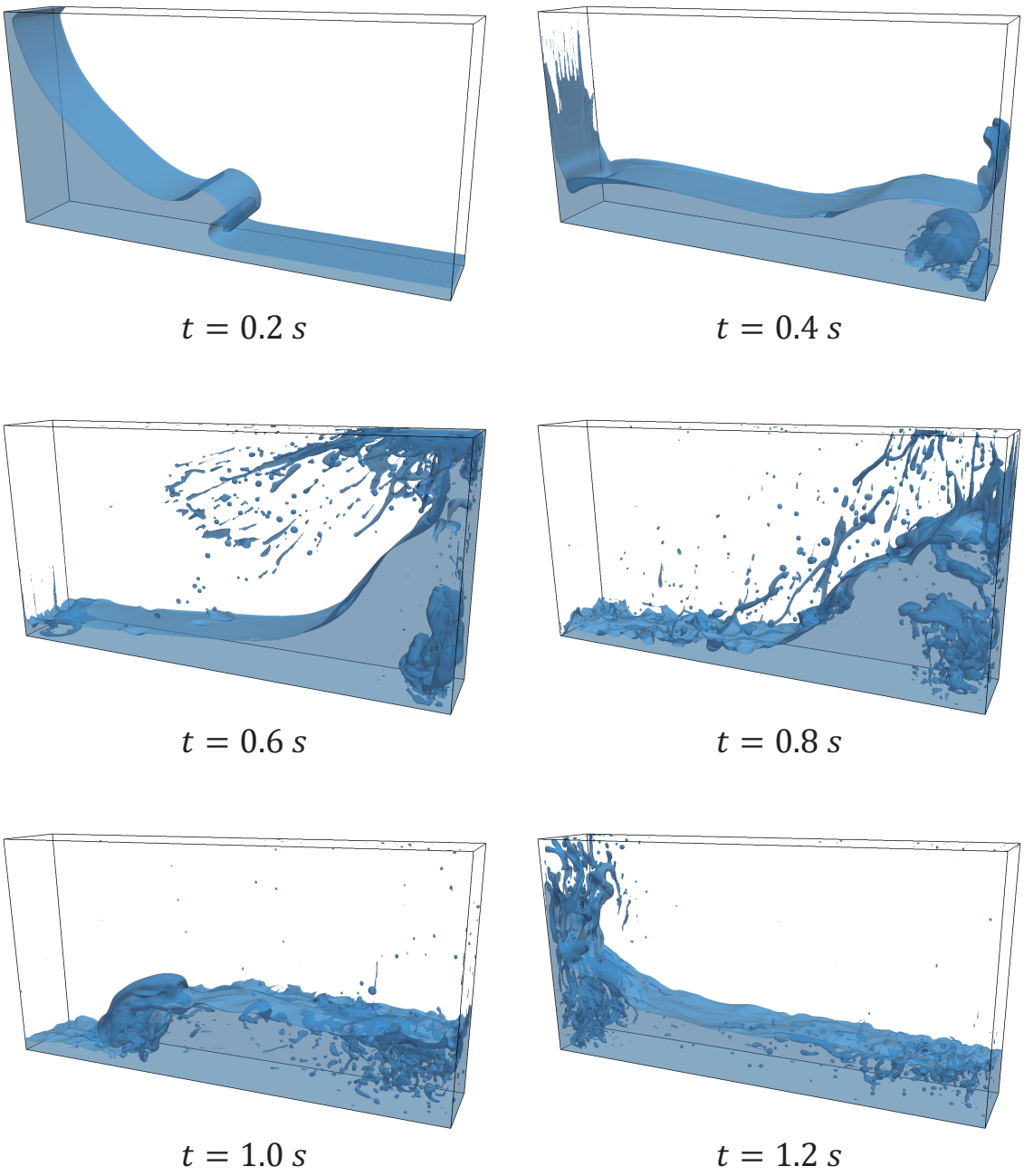


Figure 2.25: Time evolution of water surface in 3D dam break on a wet bed.

Chapter 3

Momentum Conservation for Two-phase Flow

This chapter aims at a robust simulation of violent two-phase flows with high density ratio. First a consistent and conservative formulation for momentum transport within the finite-volume framework is introduced. Then a special treatment is employed to avoid velocity-pressure decoupling on the collocated grid. The significant advantage of a consistent and momentum-conserving solver is demonstrated by the two-phase flow simulations in the presence of high density ratio.

3.1 Mathematical Model

The Navier-Stokes equations in conservative form for weakly compressible fluid under isothermal condition can be written as:

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \nabla \cdot \left[\mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\right] + \boldsymbol{F}, \tag{3.1}$$

$$\frac{\partial p}{\partial t} + \rho c_s^2 \nabla \cdot \boldsymbol{u} = 0, \tag{3.2}$$

where $\boldsymbol{u} = (u, v, w)$ is the velocity vector, ρ the density, p the pressure, μ the dynamic viscous coefficient, and $\boldsymbol{F}$ the external body force. c_s is an artificial speed of sound, which must be large enough to satisfy the low Mach number condition $Ma = |\boldsymbol{u}|_{\max} / c_s \ll 1$. The conservative form of the convection term in the momentum equation is crucial in the present model.

Besides the conservative Allen-Cahn model introduced in Section 2.1.3, the volume of fluid (VOF) method is also adopted for capturing the two-phase interface in this thesis. For immiscible two phases without phase change, the volume fraction of a fluid phase f

is purely advected by the motion of fluid, i.e., the material derivative of f is zero:

$$\frac{\partial f}{\partial t} + \boldsymbol{u} \cdot \nabla f = 0, \tag{3.3}$$

which is equivalent to the flux form

$$\frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) = f \nabla \cdot \boldsymbol{u}. \tag{3.4}$$

Note that the right-hand side of Eq. (3.4) accounts for the change in volume fraction due to fluid compressibility, and it vanishes when the fluid flow is incompressible. The right-hand side term is neglected for two reasons. One is that the compressibility is small enough at low Mach number to consider the fluid as being nearly incompressible. In addition, since the continuity equation is not explicitly included in this model, the VOF equation without the right-hand side also represents conservation of mass. Hence, the VOF equation reduces to

$$\frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) = 0. \tag{3.5}$$

The one-fluid model for the simulation of two-phase flow described in Section 2.1.4 is also used in the present solver. The only difference is that the phase-field function ϕ is replaced by the volume fraction f.

3.2 Numerical Methods

3.2.1 Consistent Transport of Mass and Momentum

A collocated grid system is used for spatial discretization, which means all primitive variables are defined at the center of each cell. The finite volume method (FVM) is used to ensure numerical conservation of mass and momentum when calculating the convective parts of Eq. (3.1) and (3.5). In the present method, the mass transport is first calculated by solving the VOF equation.

In order to compute the advection term of Eq. (3.5) accurately in the presence of steep change of volume fraction near a interface, geometric approach such as piecewise linear interface calculation (PLIC) scheme and algebraic approach including THINC [76] and its extensions to multiple dimensions such as THINC/WLIC [77] and unstructured grid UMTHINC [78] have been proposed and widely studied.

The THINC/WLIC scheme [77], which has both good accuracy and simplicity of implementation, is employed to algebraically evaluate the numerical flux of the volume fraction function. The THINC (tangent of hyperbola for interface capturing) scheme [76] uses a piecewise hyperbolic tangent function to represent the jump of volume fraction

across the interface in one dimension:

$$\Phi_i(x) = \frac{1}{2}\left(1 + \alpha \tanh\left(\beta\left(\frac{x - x_{i-\frac{1}{2}}}{\Delta x} - \widetilde{x}_i\right)\right)\right), \tag{3.6}$$

where $\alpha = 1$ for $f_{i-1} < f_{i+1}$ and $\alpha = -1$ for $f_{i-1} > f_{i+1}$, the parameter β which controls the steepness and thickness of the interface is chosen to be 2.5 here. $\widetilde{x}_i$ represents the distance from the jump center of the interface, and it is determined by solving $\frac{1}{\Delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \Phi_i(x)\, dx = f_i$.

To extend the THINC scheme to multiple dimensions, the WLIC (weighted line interface calculation) method reconstructs the interface by using a weighted average of one-dimensional line interfaces along each axis of space. The weights depend on the interface normal. When calculating the flux of the volume fraction $\mathcal{F}^V$ in a certain axis direction, only the line interface along this axis is reconstructed by the THINC scheme and its flux is a spatial integration of Eq. (3.6). The fluxes of the line interfaces perpendicular to this axis can be simply evaluated by using the 1st-order upwind scheme. For example, the flux in the x-direction can be expressed as

$$\mathcal{F}^V_{x,i+\frac{1}{2}} = \omega_{x,iup}\int_{x_{i+\frac{1}{2}}-u_{i+\frac{1}{2}}\Delta t}^{x_{i+\frac{1}{2}}} \Phi_{iup}(x)\, dx + (1-\omega_{x,iup})\, f_{iup} u_{i+\frac{1}{2}}\Delta t, \tag{3.7}$$

where $iup = i$ for $u_{i+\frac{1}{2}} > 0$, and $iup = i+1$ for $u_{i+\frac{1}{2}} \le 0$. The weight of the line interface along the x-axis is calculated using the interface normal $\boldsymbol{n} = \nabla f / |\nabla f|$:

$$\omega_x = \frac{|n_x|}{|n_x| + |n_y| + |n_z|}, \tag{3.8}$$

where n_x, n_y and n_z are three components of the normal vector in the x, y and z directions respectively.

As illustrated in Fig. 3.1, the volume flux of the heavy phase $\mathcal{F}_f^{V,h}$ and that of the light phase $\mathcal{F}_f^{V,l}$ crossing the cell face during Δt are related as follows:

$$\mathcal{F}_f^{V,l} = u_f \Delta t - \mathcal{F}_f^{V,h}, \tag{3.9}$$

where the subscript f denotes an arbitrary cell face. u_f is a face-centered advection velocity, and its construction will be described in the next subsection. The total mass flux $\mathcal{F}^{\text{Mass}}$ is obtained by summing the volume fluxes of the two-phase fluids multiplied by their corresponding densities:

$$\mathcal{F}_f^{\text{Mass}} = \rho_h \mathcal{F}_f^{V,h} + \rho_l \mathcal{F}_f^{V,l}. \tag{3.10}$$

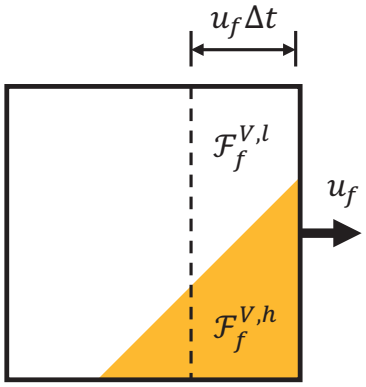


Figure 3.1: Flux of the volume fraction across the cell face during Δt.

Finally, the convection term of the momentum equation is calculated using Gauss's divergence theorem:

$$\begin{aligned}\iiint \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u})\, dV \Delta t &= \sum_f \mathcal{F}_f^{\text{Mass}} \boldsymbol{U}_f A_f \\ &= \sum_f \left(\rho_h \mathcal{F}_f^{V,h} + \rho_l \mathcal{F}_f^{V,l}\right) \boldsymbol{U}_f A_f,\end{aligned} \tag{3.11}$$

where A_f represents the area of the cell face. $\boldsymbol{U}_f$ is the velocity at the cell face that is reconstructed from the cell-centered velocities by a 3rd-order MUSCL scheme [63]. Since the volume flux in the VOF equation is included in the momentum flux, the mass and momentum transports are coupled by the above consistent formulation. Only by this way, the solution of velocity is correct when extracting the velocity from the momentum after solving the transport equations,

$$\boldsymbol{u} = \frac{\rho \boldsymbol{u}}{\rho}. \tag{3.12}$$

For comparison, an inconsistent formulation is also introduced. The convective part of the momentum equation is replaced by a non-conservative form,

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) = 0, \tag{3.13}$$

and solved by the finite difference method with the 3rd-order WENO scheme [59]. Since the time integration is directly implemented on the velocity rather than the momentum, there is no need to worry about the abnormal solution caused by extracting the velocity from the momentum via Eq. (3.12). In this approach, the mass transport and momentum transport are independent at each time step.

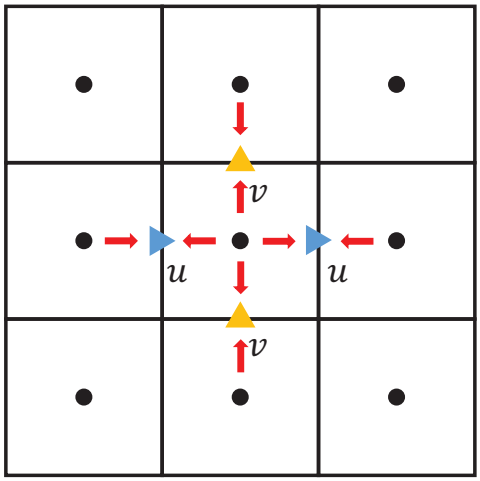


Figure 3.2: Interpolation of cell-centered velocities to faces.

3.2.2 Pressure Projection on Collocated Grid

The evolving pressure projection method described in Section 2.1.2 is also used in this conservative solver. Considering the severe restriction of the time step imposed by the speed of sound, there is a trade-off between the Mach number and the number of iterations in order to accelerate the time-to-solution. In practice, it is found that 20 iterations in every time step are sufficient if the Mach number is smaller than 0.05, which can be realized by dynamically adjusting the speed of sound as the simulation goes on.

To avoid pressure-velocity decoupling when the pressure projection is performed on the collocated grid, the concept of approximate projection [79] is adopted. Imitating the staggered grid system, a face-centered velocity field $\boldsymbol{u}_f$ is constructed by linear interpolation of the corresponding velocity components defined at the cell center, as illustrated in Fig. 3.2. The external forces coming from the surface tension and gravity are first applied to the face-centered velocities,

$$\boldsymbol{u}_f^* = \frac{\boldsymbol{u}_{c-}^* + \boldsymbol{u}_{c+}^*}{2} + \Delta t \left\langle \frac{\boldsymbol{F}}{\rho} \right\rangle_f^{n+1}, \tag{3.14}$$

where $\boldsymbol{u}_{c-}^*$ and $\boldsymbol{u}_{c+}^*$ are the intermediate cell-centered velocities at either side of the face. The density ρ_f on the cell face is also linearly interpolated from the densities at the cell centers. Then, the staggered pressure and velocities are iteratively corrected by using the evolving pressure projection method. Finally, the cell-centered velocity $\boldsymbol{u}_c$ is corrected by averaging the pressure gradient and body forces from the cell faces to the cell center:

$$\boldsymbol{u}_c^{n+1} = \boldsymbol{u}_c^* + \Delta t \left(\left\langle -\frac{\nabla p}{\rho} \right\rangle_{f\to c}^{n+1} + \left\langle \frac{\boldsymbol{F}}{\rho} \right\rangle_{f\to c}^{n+1} \right). \tag{3.15}$$

Additionally, the corrected face-centered velocity will be used as the advection velocity, which also appears in Eq. (3.9), in the next time step.

3.2.3 Time Integration

The numerical solutions of governing equations are advanced from time step t^n to $t^{n+1} = t^n + \Delta t$ by using the fractional-step method, which proceeds as follows:

(1) Compute the consistent transport of mass and momentum by simultaneously solving the convective parts of the VOF equation and the momentum equation based on Eq. (3.6)-(3.11):

$$\frac{\partial \phi}{\partial t} = -\nabla \cdot (\phi \boldsymbol{u}), \tag{3.16a}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} = -\nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}). \tag{3.16b}$$

A three-stage third-order strong-stability-preserving Runge-Kutta (SSP-RK-3) scheme [65] is used for the explicit time integration of the transport equations.

(2) Compute the average density and viscosity with the updated VOF function by Eq. (2.34) and (2.35).

(3) Explicitly solve the viscous part of Eq. (3.1) by using a first-order Euler forward scheme for the time integration:

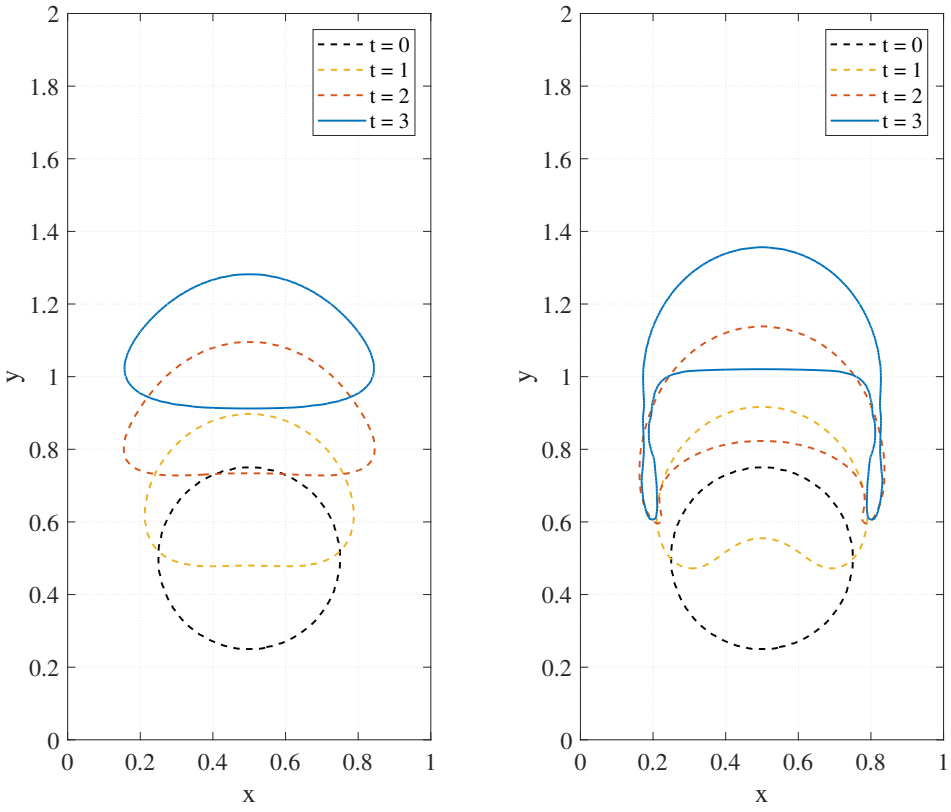
$$\frac{\partial \rho \boldsymbol{u}}{\partial t} = \nabla \cdot \left(\mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\right). \tag{3.17}$$

(4) Construct the face-centered velocity and apply the external forces to the cell face by using Eq. (3.14).

(5) Iteratively correct the face-centered velocity and pressure by using the evolving pressure projection method, i.e., Eq. (2.17)-(2.20).

(6) Approximate projection and forcing for the cell-centered velocity by using Eq. (3.15).

The algorithm described above can be easily switched to an inconsistent formulation by replacing Eq. (3.16b) with Eq. (3.13). In this case, the Runge-Kutta scheme is still used to integrate the non-conservative momentum convection equation in time, although it is no longer coupled with the transport of the volume fraction, Eq. (3.16a). In addition, the restriction imposed on time step Δt is almost same as Eq. (2.57) except for the last term which is associated with the conservative Allen-Cahn equation.



(a) Case 1 (b) Case 2

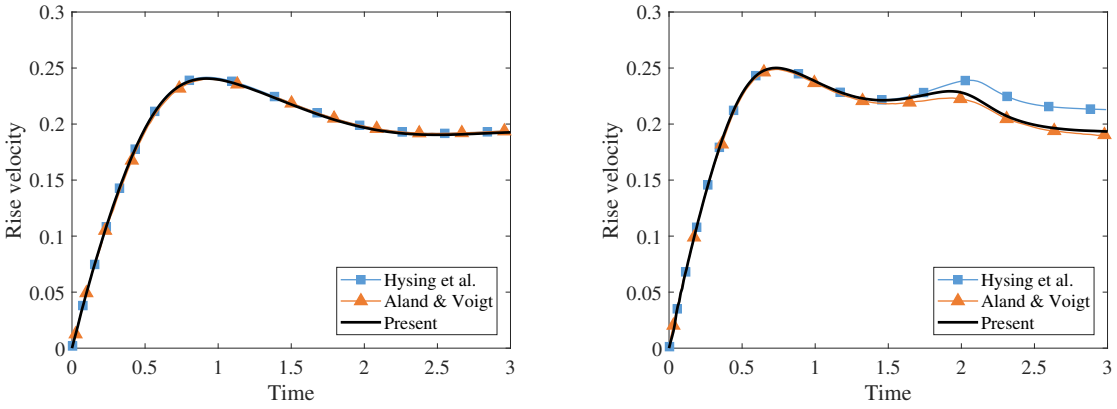
Figure 3.3: Evolution of the shape of bubble during the rising process.

3.3 Numerical Results

3.3.1 Two-dimensional Rising Bubble

For the first validation, a non-violent two-phase flow problem is solved in order to compare the numerical results with other benchmark data. The accuracy of the present method is evaluated in a two-dimensional rising bubble simulation. At the beginning, a circular bubble with diameter 0.5 is surrounded by still liquid in a 1×2 rectangular domain. The center of the bubble is located at $(0.5, 0.5)$. No-slip conditions are imposed on the top and bottom enclosing walls, and slip conditions are imposed on the left and right sides. The computation is performed on a 128×256 uniform grid. The physical properties of fluids are initialized by the same parameters in Table 2.2 for two cases respectively.

The evolution of the shape of the bubble until the ending time $t = 3$ is depicted in Fig. 3.3. Since the present method is mass-conservative, and accurate enough for interface reconstruction, the bubble trail in the case 2 is well captured in the results, which is difficult for the level-set-based method [68]. Next, the rising velocity of the bubble is



(a) Case 1 (b) Case 2

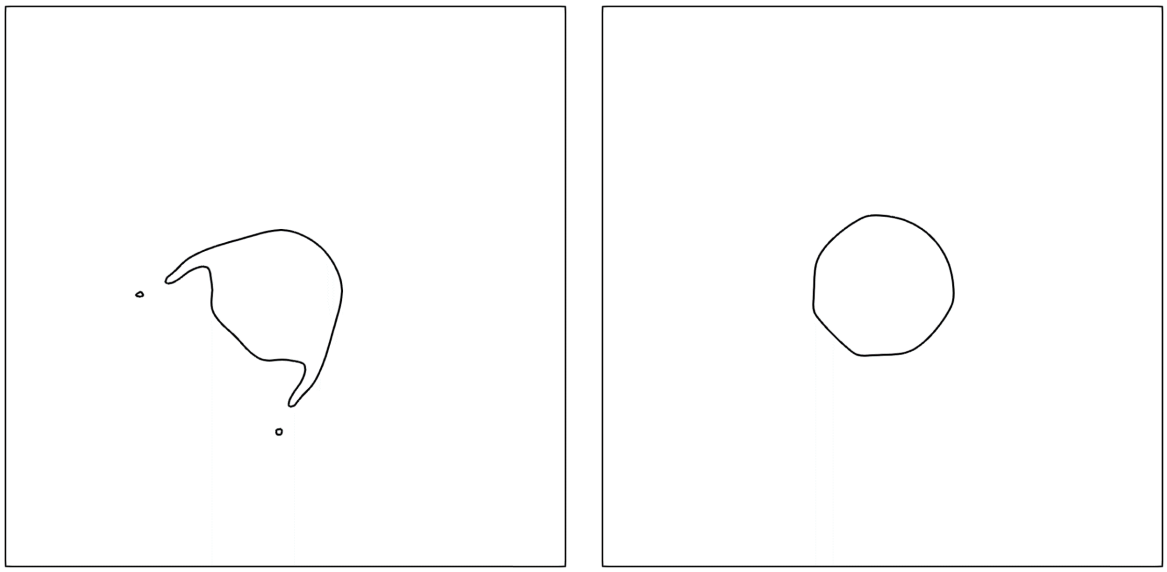
Figure 3.4: Rising velocity over time of the two-dimensional bubble.

compared with the results of Hysing et al. [68] and Aland and Voigt [67]. The latter employs the finite element method for solving the Navier-Stokes and diffusive-interface phase-field system. The rising velocity plotted in Fig. 3.4 shows good agreement in both cases. Different from the previous study in Section 2.3.4, the iteration number is fixed and the speed of sound is adjusted dynamically, as described in Section 3.2.2. There is almost no oscillation in the velocity, which indicates that the proposed approach successfully minimizes the effects of weak compressibility and the acoustic wave. Moreover, as both references solve the incompressible Navier-Stokes equation, these comparative results prove that the present solver can be used to accurately simulate an incompressible two-phase flow.

3.3.2 Transport of a Heavy Droplet

The transport of a heavy droplet is a test that is often conducted in studies on the momentum-conserving or consistent transport methods. Here, a circular droplet of diameter $D = 0.25$ is placed at the center of a periodic unit square domain. The density of the liquid droplet is $\rho_h = 10^6$, and the density of the surrounding gas is $\rho_l = 1$. Both the droplet and the gas are inviscid, i.e., $\mu_h = \mu_l = 0$. This problem is free of the gravity and surface tension force. In a disk area of diameter $D + 2\Delta x$ concentric with the droplet, the velocities are initialized to $u = 1, v = 1$, while the fluid in the other region is at rest. The droplet velocity is initialized in an area slightly larger than the droplet in order to improve the results in the early stage of the simulation [31]. The computation is performed on a 128×128 uniform grid, correspondingly, 32 meshes are assigned to the diameter of the droplet.

The droplet goes through a cycle of motion at $t = 1$ and returns to its original position.



(a) Inconsistent (b) Consistent

Figure 3.5: Droplet shapes after a cycle of motion.

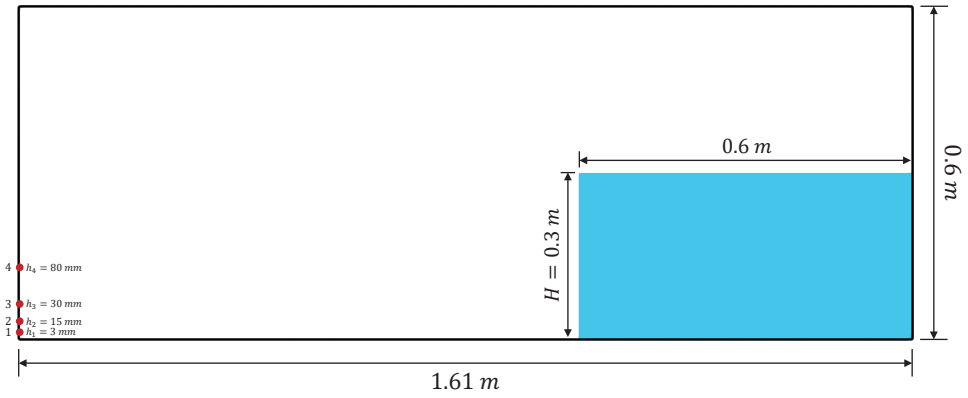


Figure 3.6: Schematic diagram of the collapse of a water column.

Fig. 3.5 compares the droplet shapes obtained by the consistent and inconsistent methods described in section 3.2.1. Although the density ratio 10^6 is very high and this problem is dominated by convection of the heavy fluid, the droplet undergoes unphysical distortion if the mass and momentum transports are inconsistent. In contrast, the consistent method retains the circular shape. There is no significant deformation of the droplet. Hence, it is necessary to use the consistent method when the density ratio between the two phases is high. Or rather, the consistent method is always preferable for the two-phase simulation.

(a) Inconsistent

(b) Consistent

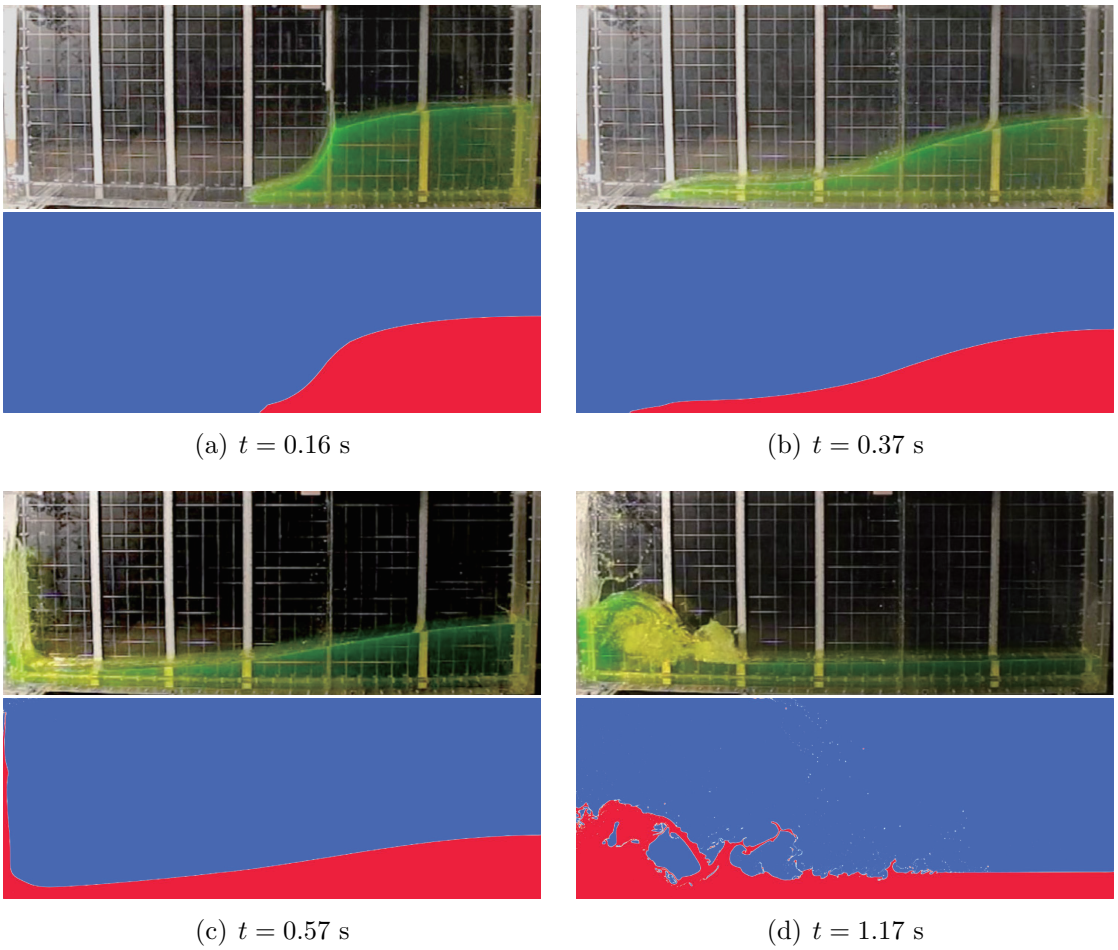
Figure 3.7: Zoom-ins on the front of a collapsing water column at $t = 0.35$ s.

3.3.3 Collapse of Water Column

The two-dimensional collapse of a water column, which is also known as dam break, is simulated to demonstrate the importance of consistent transport on the stability of the two-phase interface at high Reynolds number. The computation is initialized by following the experimental setup of Lobovskỳ et al. [80]. A reasonable simplification to a two-dimensional simulation is made based on the fact that the water surface profile is almost the same in the breadth direction of the tank during the collapse process. The initial water column has a width 0.6 m and height $H = 0.3$ m, located at the right and bottom of a rectangular tank of size 1.61 m $\times$ 0.6 m. A schematic representation of the water and domain is illustrated in Fig. 3.6. On the downstream vertical wall, four pressure monitoring points are selected at the same position as in the experiment: 3 mm, 15 mm, 30 mm and 80 mm. The physical properties of the water and air are $\rho_h = 997$ kg/m^3, $\rho_l = 1.2$ kg/m^3, $\mu_h = 8.8733 \times 10^{-4}$ Pa·s, $\mu_l = 1.8 \times 10^{-5}$ Pa·s, and surface tension coefficient $\sigma = 0.072$ N/m. The gravitational acceleration is $g = 9.8$ m/s^2. Because of the high Reynolds number 3.8×10^6 reported in the experiment, the computational domain is discretized on a relatively high-resolution grid that has 1610 $\times$ 600 cells in order to extract the data more accurately in the flow field. No-slip conditions are imposed on the enclosing walls.

In this simulation, the downstream distance of the domain is sufficiently long to imitate a strong shear flow situation. The interface is unstable when there is a large velocity difference across it with small perturbations between two fluids. This phenomenon is known as Kelvin-Helmholtz instability (KHI). According to an analysis using linear theory [81], gravity and surface tension force play a role in stabilizing the interface. Substituting typical parameters in the case of a collapsing water column into the formula for the linear growth rate of KHI:

$$\gamma = \sqrt{\frac{k^2 \rho_1 \rho_2 \left(U_1 - U_2\right)^2}{\left(\rho_1 + \rho_2\right)^2} - \frac{kg\left(\rho_1 - \rho_2\right)}{\rho_1 + \rho_2} - \frac{k^3 \sigma}{\rho_1 + \rho_2}}, \tag{3.18}$$



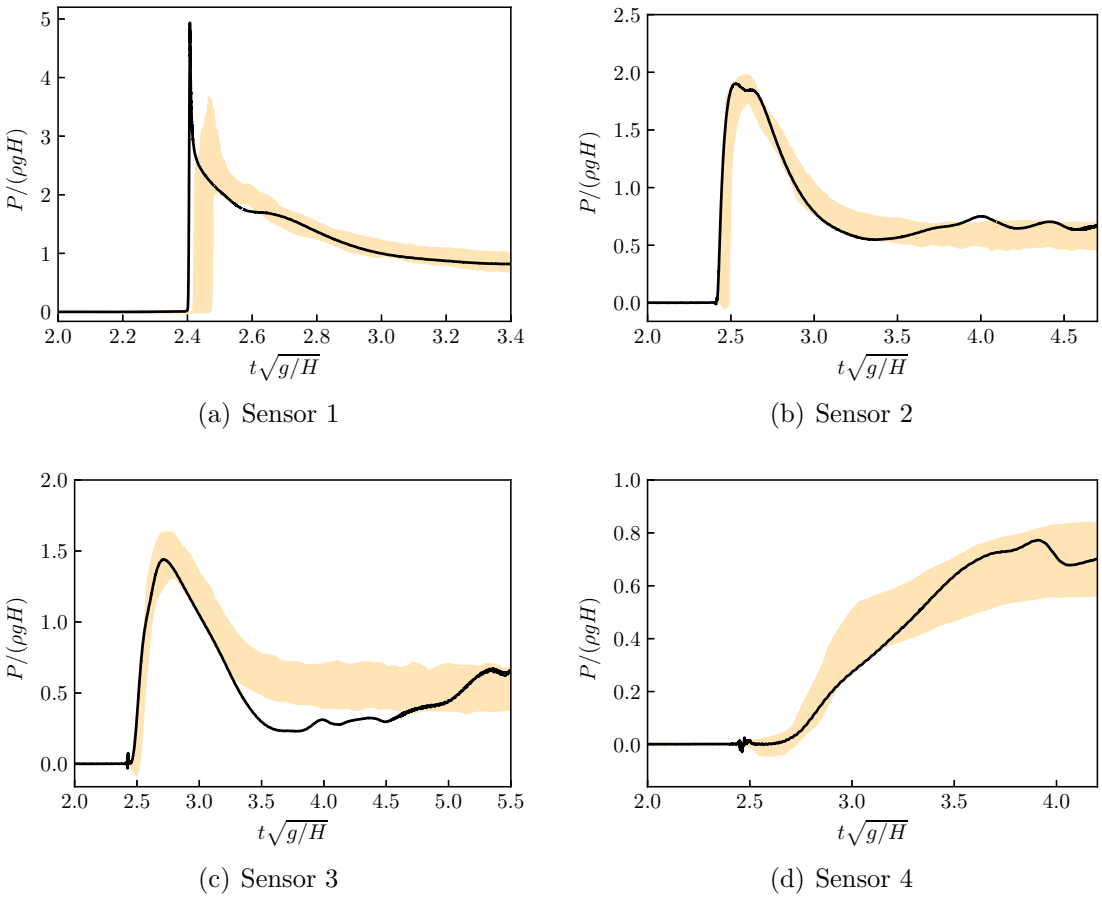
(a) $t = 0.16$ s (b) $t = 0.37$ s

(c) $t = 0.57$ s (d) $t = 1.17$ s

Figure 3.8: Evolution of water surface profile: experiment [80] (top) and numerical results of the consistent method (bottom).

where k is the wave number of the interface perturbation, yields a negative value within the square root. Therefore, the Kelvin-Helmholtz instability shown in Fig. 3.7(a) shouldn't occur. In contrast, the consistent method more realistically reproduces the profile of the water front, which compares favorably to the experiment and another numerical simulation using the SPH method [82]. It reveals that solving the non-conservative momentum equation will result in that if the low-density fluid has a large velocity, it will have a large impact on the movement of the high-density fluid.

The time evolution of the water surface obtained by the consistent method is compared with the experimental snapshots in Fig. 3.8. Except in the early stage of collapse, which is under the influence of the gate release, the surface profiles, water heights and front positions are in good agreement. Fig. 3.9 shows the time variation of the impact pressure at four monitoring points on the downstream wall in comparison with the sensor data in the experiment. Non-dimensional time $t^* = t\sqrt{g/H}$ and pressure $p/(\rho g H)$ are used,



(a) Sensor 1 (b) Sensor 2

(c) Sensor 3 (d) Sensor 4

Figure 3.9: History of impact pressure obtained by the consistent method in comparison with experimental values [80].

where ρ is the density of water and H is the initial height of the water column. The color band in the figure represents the experimental data between the 2.5% and 97.5% percentile levels of 100 tests. In Fig. 3.9(a), the rising point of the impact pressure measured in the numerical simulation is a little ahead of that of the experiment. The peak pressure is also higher. This can be explained by the fact that the water column is not subject to resistance from the side wall in the two-dimensional simulation, and the water front hits the lower corner of the tank earlier with a slightly higher speed. Nevertheless, the simulation is in satisfactory agreement with the impact pressures measured by the four sensors.

3.3.4 Three-dimensional rising bubble

To validate the conservative and consistent solver for realistic three-dimensional two-phase flow simulations, the rising of a single bubble in a liquid under various conditions

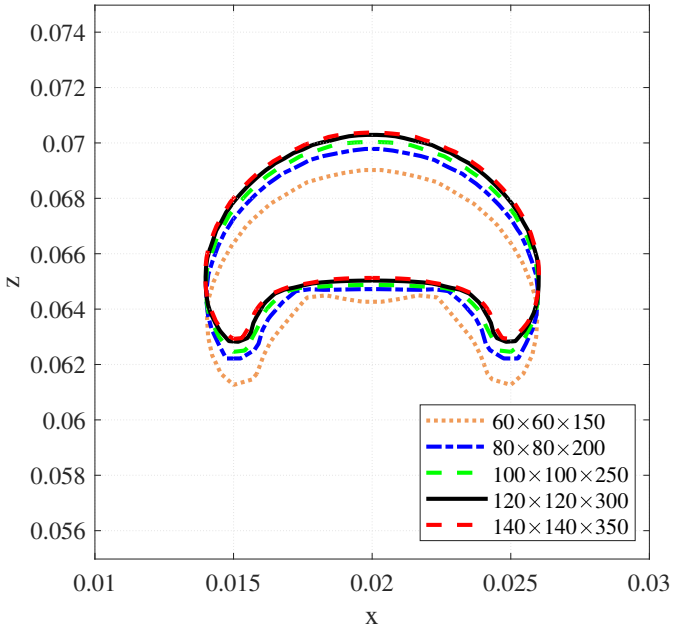


Figure 3.10: Effect of mesh resolution on the simulated bubble shape.

is simulated and the results are compared with the experiment conducted by Bhaga and Weber [83]. The final bubble shapes are categorized based on the non-dimensional Eötvös number, also known as the Bond number, and the Morton number:

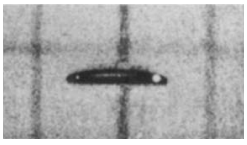
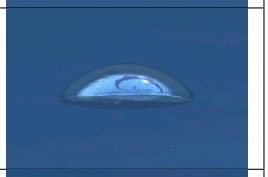
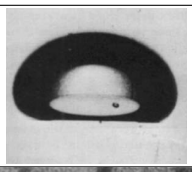
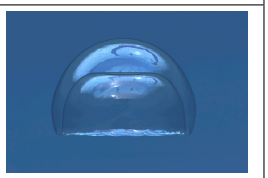
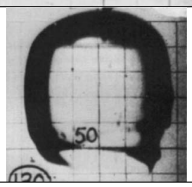
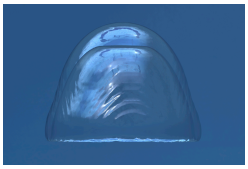
$$Eo = \frac{\rho_h g D^2}{\sigma}, \tag{3.19a}$$

$$Mo = \frac{g\mu_h^4}{\rho_h \sigma^3}. \tag{3.19b}$$

In this simulation, the density ratio and the viscosity ratio are set to 1000 and 100, respectively. The liquid density is $\rho_h = 1000$ and the bubble density is $\rho_l = 1$. The viscosities of the two phases are $\mu_h = 10^{-3}$ and $\mu_l = 10^{-5}$. The bubble diameter is $D = 0.01$. Once the Eötvös and Morton numbers are known, the gravitational acceleration g and the surface tension coefficient σ can be determined accordingly. Reference numerical simulations [72, 84] have shown that a computational domain with lateral dimensions of four times the bubble diameter in the horizontal directions has a negligible boundary effect on the rising bubble. Therefore, the domain is filled with quiescent liquid, measuring $0.04 \times 0.04 \times 0.1$, with the bubble of diameter 0.01 centered at $(0.02, 0.02, 0.02)$. Slip boundary conditions are imposed on the enclosing domain.

Before comparing the results with experiment, the grid independence is examined. The case with $Eo = 116$ and $Mo = 41.1$ is computed on several different uniform grids, where the bubble diameter is resolved by $15 \sim 35$ grid cells. Figure 3.10 shows the contour of the bubble shape on the central slice of the domain at the non-dimensional

Table 3.1: Comparison of the terminal bubble shapes between experiment and simulation under cases A1-A8.

Case	Condition	Experiment	Simulation
A1	$Eo = 8.67$ $Mo = 711$		
A2	$Eo = 17.7$ $Mo = 711$		
A3	$Eo = 32.2$ $Mo = 8.2 \times 10^{-4}$		
A4	$Eo = 243$ $Mo = 266$		
A5	$Eo = 115$ $Mo = 4.63 \times 10^{-3}$		
A6	$Eo = 237$ $Mo = 8.2 \times 10^{-4}$		
A7	$Eo = 339$ $Mo = 43.1$		
A8	$Eo = 641$ $Mo = 43.1$		

time $\tau = t/\sqrt{D/g} = 10$. The present results show that a grid finer than $120 \times 120 \times 300$ had no significant effect on the predicted bubble shape. Therefore, the $120 \times 120 \times 300$ grid with a diameter of 30 cells is used for the following simulations.

The test cases listed in Table 3.1 and Table 3.2 cover almost all typical bubble shapes,

Table 3.2: Comparison of the terminal bubble shapes between experiment and simulation under cases B1-B8.

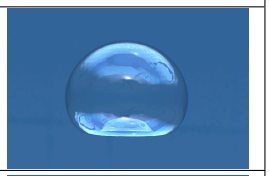
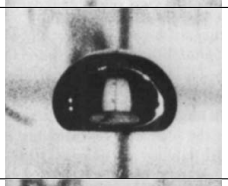
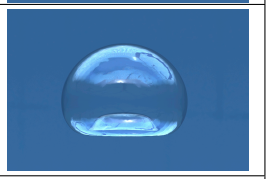
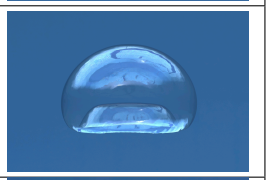
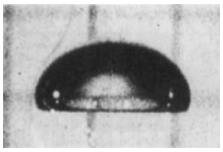
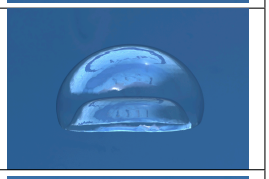
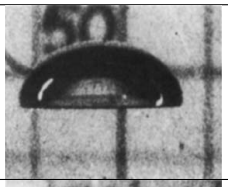
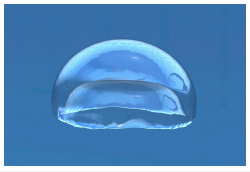
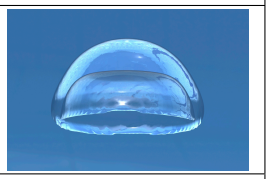
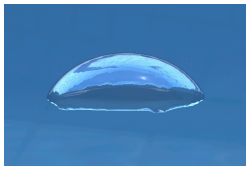
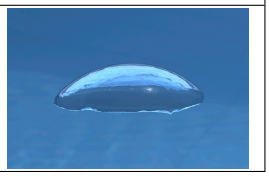
Case	Condition	Experiment	Simulation
B1	$Eo = 116$ $Mo = 848$		
B2	$Eo = 116$ $Mo = 266$		
B3	$Eo = 116$ $Mo = 41.1$		
B4	$Eo = 116$ $Mo = 5.51$		
B5	$Eo = 116$ $Mo = 1.31$		
B6	$Eo = 116$ $Mo = 0.103$		
B7	$Eo = 116$ $Mo = 4.63 \times 10^{-3}$		
B8	$Eo = 116$ $Mo = 8.60 \times 10^{-4}$		

Table 3.3: Physical parameters and fluid properties for jet in cross flow simulations.

Case	ρ_l [kg/m^3]	ρ_g [kg/m^3]	μ_l [kg/(ms)]	μ_g [kg/(ms)]	σ [N/m]	V_l [m/s]	V_g [m/s]	$q = \frac{\rho_l V_l^2}{\rho_g V_g^2}$	$We = \frac{\rho_g V_g^2 d}{\sigma}$	$Re = \frac{\rho_l V_l^2 d}{\mu_l}$
1	118	1.18	0.000307	0.0000186	0.0708	51.45	54.8	88.2	40	15800
2	997	1.18	0.000894	0.0000186	0.0708	17.7	54.8	88.2	40	15800

including spherical, oblate ellipsoidal, oblate ellipsoidal cap, spherical cap, and skirted shapes. The predicted bubble shapes using the consistent solver agree very well with the experiment [83]. It is worth noting that cases A5, A6, B7, and B8 have relatively high Reynolds numbers, and the motion of the bubble does not reach the steady state, making the bubble easy to break up. Therefore, one of the snapshots in the simulation is just selected to match the experiment. These phenomena are also observed in the reference numerical computation [84].

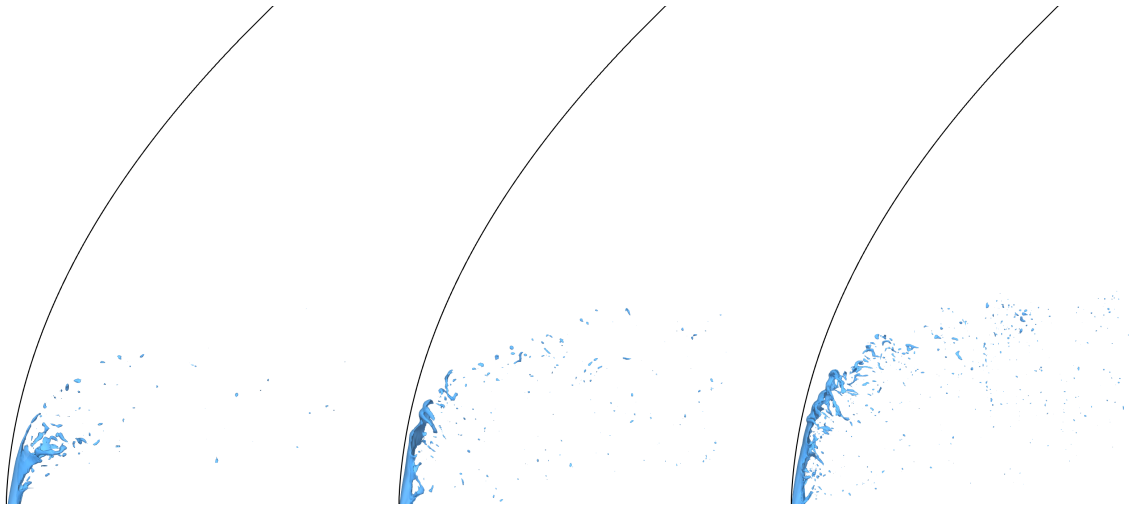
3.3.5 Liquid Jet in Gas Cross-Flow

To demonstrate the stability of the present method as a practical two-phase flow solver, the problem of a three dimensional jet in an air cross-flow is simulated. This is a suitable validation of the importance of consistent transport and momentum conservation when applied to realistic two-phase flows with a high density ratio and high Reynolds number. Experimental studies on this problem have been reported in [85], and a numerical study has been reported in [33].

The geometric and physical parameters follow the computation conditions described in [33]. The domain size measures 2.0 cm $\times$ 1.5 cm $\times$ 3.5 cm and is discretized by the $256 \times 192 \times 448$, $384 \times 288 \times 672$ and $512 \times 384 \times 896$ uniform grids with different resolutions. The liquid nozzle of diameter 0.8 mm is placed on the bottom wall and centered at $(0.2\text{ cm}, 0.75\text{ cm}, 0\text{ cm})$. The air blows in from the left boundary in the positive x direction with a constant velocity V_g, and the liquid is injected from the nozzle along the positive z direction with a constant velocity V_l. The physical properties are listed in Table 3.3. In case 1, the density ratio is 100, while in case 2, the density ratio corresponds to the actual value for water and air. However, the momentum ratio q, Weber number and Reynolds number are the same for both cases, indicating that both cases belong to the multimode breakup regime [85]. No-slip conditions are imposed on the bottom wall $z_{\min}$ except for the nozzle. Outflow conditions are applied to the right boundary $x_{\max}$. Other walls at $y_{\min}$, $y_{\max}$ and $z_{\max}$ have slip conditions.

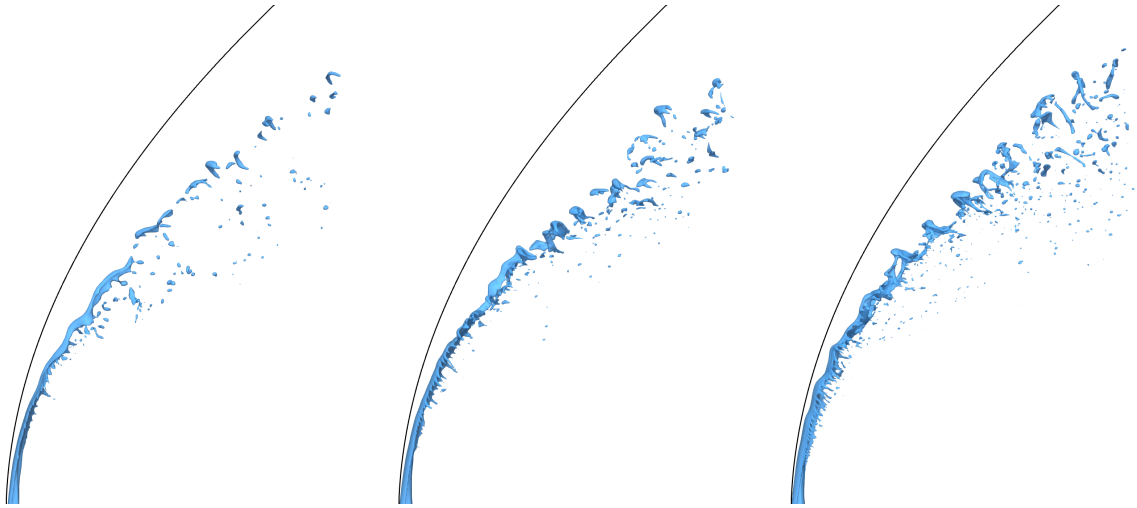
Wu et al. [86] proposed the equation of the liquid column trajectory as

$$y/d = \sqrt{\pi/C_d}\sqrt{q\,(x/d)}, \tag{3.20}$$



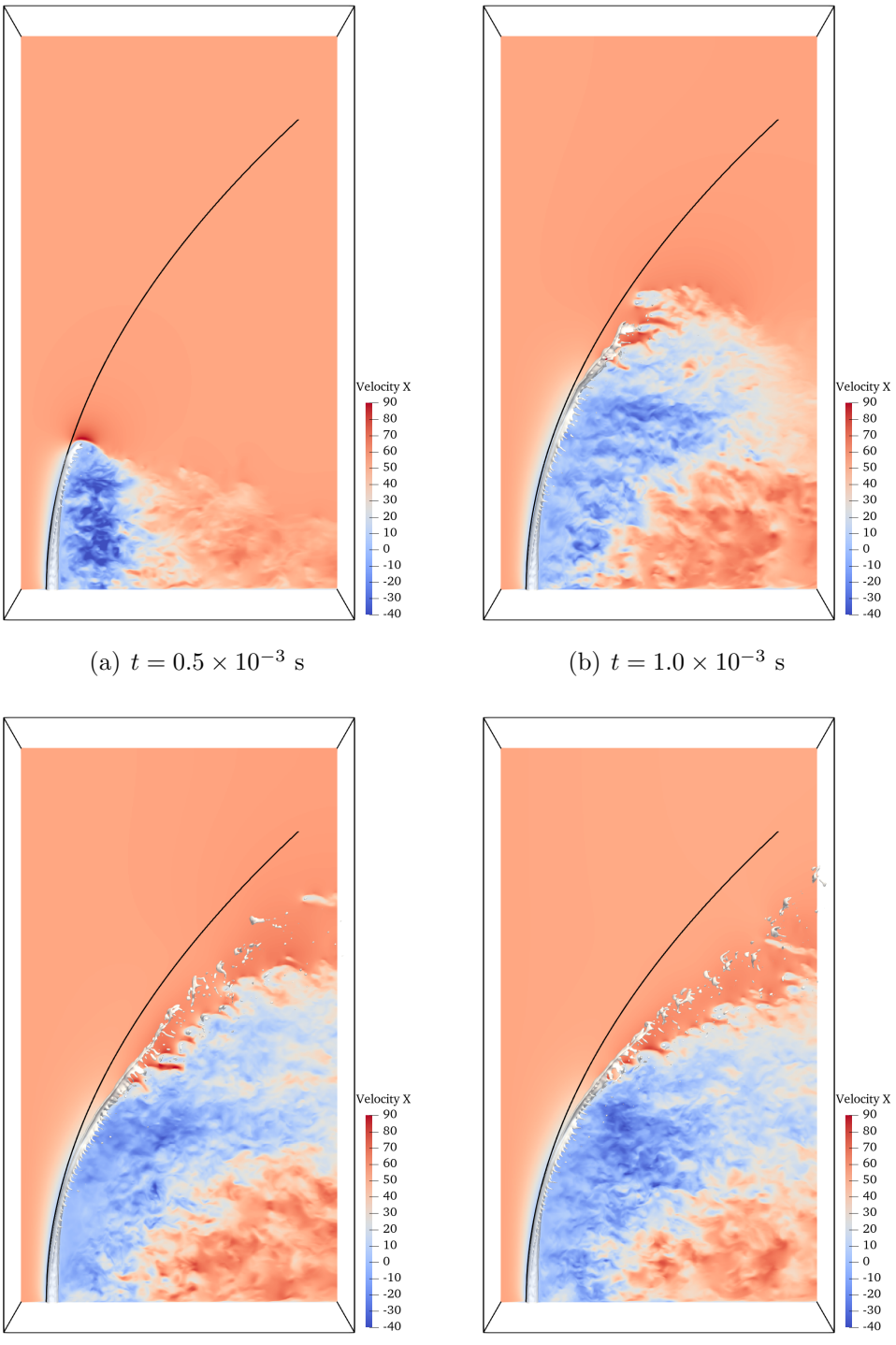
(a) $256 \times 192 \times 448$ (b) $384 \times 288 \times 672$ (c) $512 \times 384 \times 896$

Figure 3.11: Inconsistent solver for case 1 of jet in cross flow simulation at $t = 2.0 \times 10^{-3}$ s.



(a) $256 \times 192 \times 448$ (b) $384 \times 288 \times 672$ (c) $512 \times 384 \times 896$

Figure 3.12: Consistent solver for case 1 of jet in cross flow simulation at $t = 2.0 \times 10^{-3}$ s.



(a) $t = 0.5 \times 10^{-3}$ s

(b) $t = 1.0 \times 10^{-3}$ s

(c) $t = 1.5 \times 10^{-3}$ s

(d) $t = 2.0 \times 10^{-3}$ s

Figure 3.13: Consistent solver for case 2 of jet in cross flow simulation at density ratio corresponding to water and air.

where the drag coefficient fitted by the experiment of Sallam et al. for the multimode breakup regime is $C_d = 4$. Fig. 3.11 and Fig. 3.12 show the liquid surfaces at $t = 2.0 \times 10^{-3}$ s obtained by the inconsistent and consistent methods for case 1. By comparing these results with the experimental results in the literature [85], it is obvious that the consistent method gives more robust simulation results on the interface shape and the trajectory of the jet. The results of the inconsistent method show that the jet column breaks up within a short distance after leaving the nozzle, influenced by the air cross-flow. Moreover, the droplets formed after break up are quickly blown downstream with the air. Even after refining the mesh, the liquid column breakup characteristic obtained by the inconsistent solver does not improve significantly. Such a phenomenon in the numerical simulation is in line with expectations, if the mass and momentum transports are inconsistent, the large momentum stored in the heavy fluid will leak into the light fluid, causing the velocity of the light fluid to increase dramatically. In turn, the light fluid with a high velocity will have a strong non-physical effect on the motion of the heavy fluid.

Finally, the consistent method is used to compute case 2 on the finest uniform grid. The time evolution of the jet interface till 2.0×10^{-3} s and the velocity field on the central section of the computational domain are depicted in Fig. 3.13. The computation remains stable, although the two-phase flow with a high density ratio becomes very turbulent. The trajectory of the liquid column is also in good agreement with other numerical simulations [33, 87] and an experimental study [85]. Considering the complexity of the liquid jet phenomenon, further numerical simulation with much higher grid resolutions are needed to better understand the formation and evolution of the jet structure.

Chapter 4

Multi-GPU Scalability

This chapter first introduces two novel momentum-conserving weakly compressible Navier-Stokes solvers with difference methods for interface capturing and curvature estimation. Then the performance and scalability for the solvers are evaluated on multi-GPU supercomputers. Several numerical results are presented to show the accuracy and fidelity in large-scale two-phase flow simulations.

4.1 Conservative Solvers

4.1.1 Con-CAC-LS

The first conservative solver for two-phase flow simulation introduced here is referred as Con-CAC-LS, which represents the combination of the conservative Navier-Stokes equation, conservative Allen-Cahn equation and level-set method. This solver essentially follows the finite volume framework designed in Chapter 3, except that the Conservative Allen-Cahn model Eq. (2.24) is used to capture the gas-liquid interface. Different with the VOF method, the volume flux crossing a cell face in the conservative Allen-Cahn equation is given by

$$\mathcal{F}_f^V = u_f\phi - M\left(\nabla\phi - \frac{4\phi(1-\phi)}{W}\boldsymbol{n}\right), \tag{4.1}$$

where the subscript f denotes an arbitrary cell face. u_f is a face-centered advection velocity. Then Eq. (4.1) can be substituted into Eq. (3.9) and Eq. (3.10) to evaluate the mass flux.

In the Con-CAC-LS solver, the level-set method is coupled with the conservative Allen-Cahn model by using the approach described in Section 2.1.3. And the level-set function is also used to calculate the interface normal and curvature.

4.1.2 Con-PLIC-HF

The second conservative solver combines the momentum equation Eq. (2.2), PLIC and height function method. Therefore, it is referred as Con-PLIC-HF. In this solver, the motion of the interface between two fluids is governed by the volume of fluid (VOF) equation

$$\frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) = f \nabla \cdot \boldsymbol{u}, \tag{4.2}$$

where f represents the volume fraction of the heavy fluid. Different with the conservative Allen-Cahn equation Eq. 2.24, the above VOF equation is used as a substitute for the continuity equation representing conservation of mass. Hence additional continuity equations for phase 1 and phase 2 need to be solved simultaneously:

$$\frac{\partial f_i \rho_i}{\partial t} + \nabla \cdot (\boldsymbol{u} f_i \rho_i) = 0, \tag{4.3}$$

where the subscript $i \in \{1, 2\}$ indicates the quantities associated with phase i. Similarly, the convective part of Eq. (3.1) is also split into two equations for phase 1 and phase 2, respectively:

$$\frac{\partial f_i \rho_i \boldsymbol{u}}{\partial t} + \nabla \cdot (f_i \rho_i \boldsymbol{u} \otimes \boldsymbol{u}) = 0, \tag{4.4}$$

The average density and momentum of the fluid can be obtained directly by summing the two components as

$$f_1 + f_2 = 1, \tag{4.5a}$$

$$f_1 \rho_1 + f_2 \rho_2 = \rho, \tag{4.5b}$$

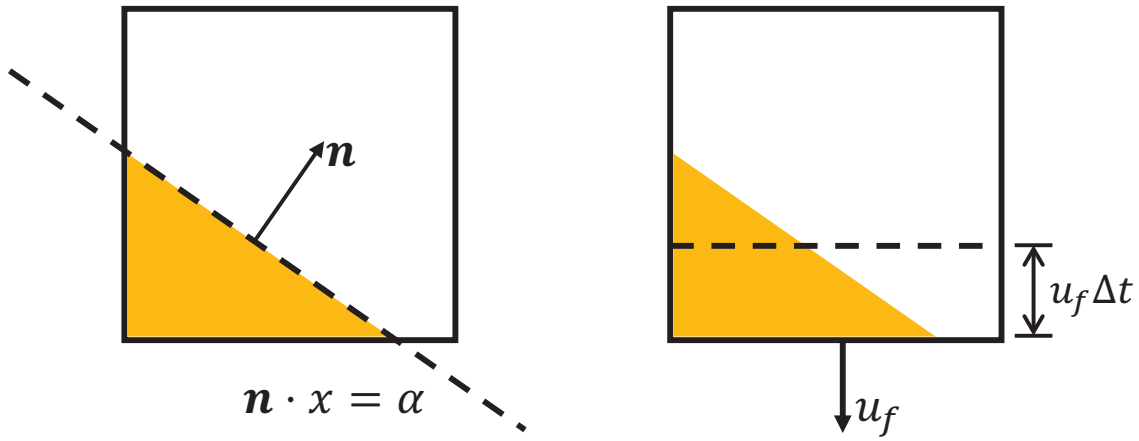
$$f_1 \rho_1 \boldsymbol{u} + f_2 \rho_2 \boldsymbol{u} = \rho \boldsymbol{u}. \tag{4.5c}$$

Fluid viscosity is still evaluated by Eq. (2.35) after replacing the phase-field function in it with the volume fraction.

As one of the geometric VOF methods, the piecewise linear interface calculation (PLIC) method approximates the interface in each interfacial cell as a line in two dimensions or a plane in three dimensions, as illustrated in Fig. 4.1(a). The linear interface is defined by the equation

$$\boldsymbol{n} \cdot \boldsymbol{x} = \alpha, \tag{4.6}$$

where α is a constant that enforces the volume fraction cut by the interface is f. Here, the interface normal $\boldsymbol{n} = \nabla f / |\nabla f|$ is computed by the mixed Youngs-centered (MYC) method [88]. The first step of PLIC method is the interface reconstruction, i.e., to determine α given the volume fraction f and normal vector $\boldsymbol{n}$. Since the orthogonal grid is used for discretization, instead of the root-finding algorithm, analytical expressions can be employed to find α with much cheaper computation [89]. Then, as illustrated in



(a) Interface reconstruction (b) Flux calculation

Figure 4.1: PLIC method.

Fig. 4.1(b), the volume flux across the cell face is calculated by the geometric relation to advect the reconstructed interface, which is the inverse of the previous problem. The analytical method [89] is also used to find the volume fraction f in the rectangular cell given $\boldsymbol{n}$ and α of the linear interface.

The Weymouth & Yue method [90] is used to solve the VOF equation Eq. (4.2), where the multi-dimensional advection of Eq. (3.5) is split into a series of one-dimensional advection problem. Moreover, the velocity divergence term cannot be neglected in one dimension even if the velocity is divergence free in two or three dimensions. Take the two-dimensional case as an example, the Weymouth & Yue advection scheme solves the following two equations in succession:

$$\frac{\partial f}{\partial t} + \frac{\partial (uf)}{\partial x} - f_c \frac{\partial u}{\partial x}, \tag{4.7a}$$

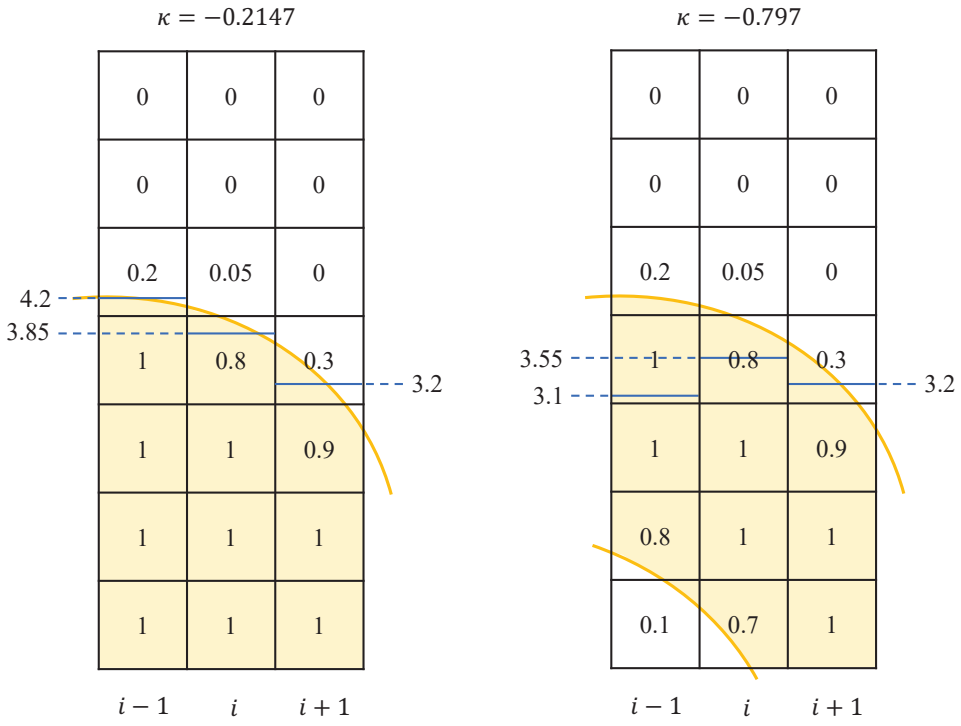
$$\frac{\partial f}{\partial t} + \frac{\partial (vf)}{\partial y} - f_c \frac{\partial v}{\partial y}, \tag{4.7b}$$

where the cell-centered value f_c is treated explicitly by using the f from the previous time step:

$$f_c = \begin{cases} 1, & \text{if } f > 0.5 \\ 0, & \text{else.} \end{cases} \tag{4.8}$$

To enforce the no overfilling or over-emptying condition for the volume fraction, another restriction is imposed on the time step as

$$\Delta t \sum_{d=1}^{N_d} \left| \frac{u_d}{\Delta x_d} \right| < \frac{1}{2}. \tag{4.9}$$



(a) Fully-resolved interface (b) Under-resolved interface

Figure 4.2: Height function for curvature estimation.

In order to make the transport of mass and momentum consistent with that of volume fraction, Eq. (4.3) and Eq. (4.4) are also solved by the dimensional-splitting method. The density and momentum at the cell face are reconstructed by the Bell–Collela–Glaz (BCG) second-order upwind scheme [38, 91]. To couple the transport equations, the numerical fluxes of mass and momentum are evaluated through multiplying the reconstructed value by the flux of volume fraction $\mathcal{F}^V$ which is computed by the PLIC method:

$$\mathcal{F}_{f_i \boldsymbol{Y}_i} = \boldsymbol{Y}_{i,\text{adv}} \mathcal{F}_{f_i}, \tag{4.10}$$

where $\boldsymbol{Y}_i = (\rho_i, \rho_i \boldsymbol{u})$ and $\boldsymbol{Y}_{i,\text{adv}}$ represents the reconstructed quantities at the cell face by the BCG scheme. $\mathcal{F}_{f_i \boldsymbol{Y}_i}$ denotes the amount of mass and momentum transported across the cell face during the time step. As illustrated in Fig. 4.1(b), the volume flux of the phase 1 $\mathcal{F}_{f_1}$ and that of the phase 2 $\mathcal{F}_{f_2}$ are related as follows:

$$\mathcal{F}_{f_1} = u_f \Delta t - \mathcal{F}_{f_2}. \tag{4.11}$$

In the case of geometric VOF method, it is straightforward to apply the height function method [92] for the estimation of interface curvature. As illustrated in Fig. 4.2(a), depending on the orientation of interface normal, the height function is obtained by summing up the volume fraction in a column or row. In the case of $|n_y| > |n_x|$ in two

Table 4.1: Comparison between Con-CAC-LS and Con-PLIC-HF solvers.

Solver	Con-CAC-LS	Con-PLIC-HF
Interface capturing	Conservative Allen-Cahn model	PLIC-VOF method
Curvature	Level-set function	Height function
Transport variables	$\phi, \psi, \rho\boldsymbol{u}$	$f, f_i\rho_i, f_i\rho_i\boldsymbol{u}$
Convection term	3rd order MUSCL scheme	2nd order BCG scheme
Time integration	3rd order Runge-Kutta method	2nd order direction-splitting method
Viscous term	2nd order central difference scheme (space) 1st order forward Euler method (time)	
Surface tension	Density-scaled CSF model	
Pressure term	Evolving pressure projection method	

dimensions, the height function H is calculated by

$$H_{i+s} = \sum_{t=-3}^{3} f_{i+s,j+t}, \ \ s = -1, 0, 1. \tag{4.12}$$

And the curvature can only be validly calculated at the interfacial cell which means $3 \leq H_i < 4$. To estimate the curvature at this cell, a 3×7 stencil around this cell is required. Then, the curvature is obtained by

$$\kappa = \frac{H_{xx}}{\left(1 + H_x^2\right)^{3/2}}. \tag{4.13}$$

In the case of three dimensions, the formulation is

$$\kappa = \frac{H_{xx} + H_{yy} + H_{xx}H_y^2 + H_{yy}H_x^2 - 2H_{xy}H_xH_y}{\left(1 + H_x^2 + H_y^2\right)^{3/2}}. \tag{4.14}$$

A situation where the interface is under-resolved by the low mesh resolution is often encountered in practice, since the stencil for calculating the height function is quite long. As illustrated in Fig. 4.2(b), although the central cell contains the interface with the same geometry as that of Fig. 4.2(a), the estimated curvature has a very large error compared to the correct value. Therefore, the values of volume fraction should be modified when the under-resolved interface is detected. The approach to enforce a local monotonic variation of volume fraction [93] is employed in this work. An improved height function technique is adopted here with a filtered discretization of the partial derivatives of the height function [93], which has a better accuracy than the standard central difference scheme.

Table 4.1 summarizes the main features of two solvers. Except for the differences in the interface capturing, curvature estimation and transport equations, Con-CAC-LS solver use a 3rd order RK for time integration, while the Con-PLIC-HF is solved by a direction-split forward Euler method. The viscous term is treated explicitly, and a density scaled continuum surface force model is used to calculate the surface tension in both two solvers.

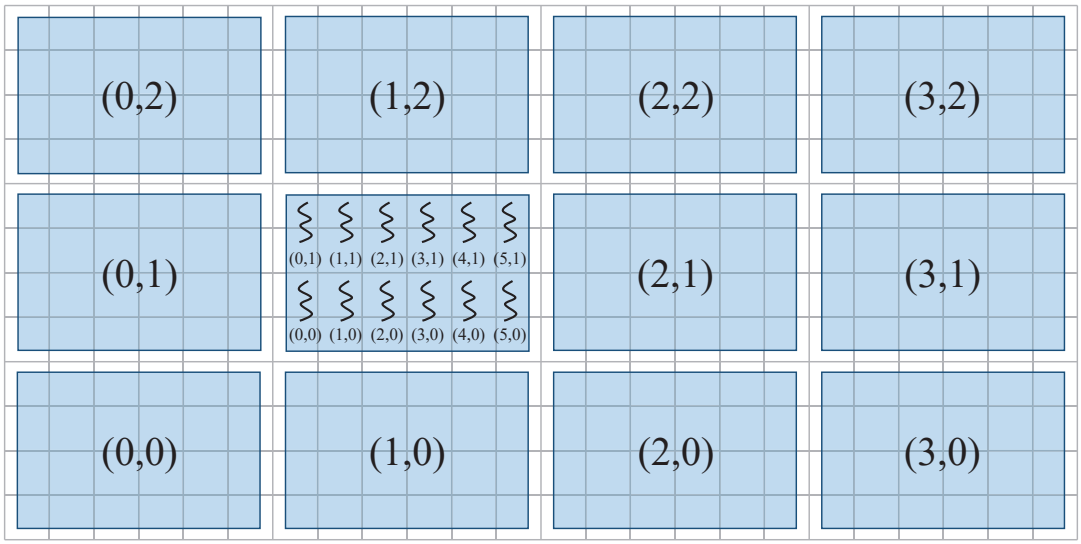


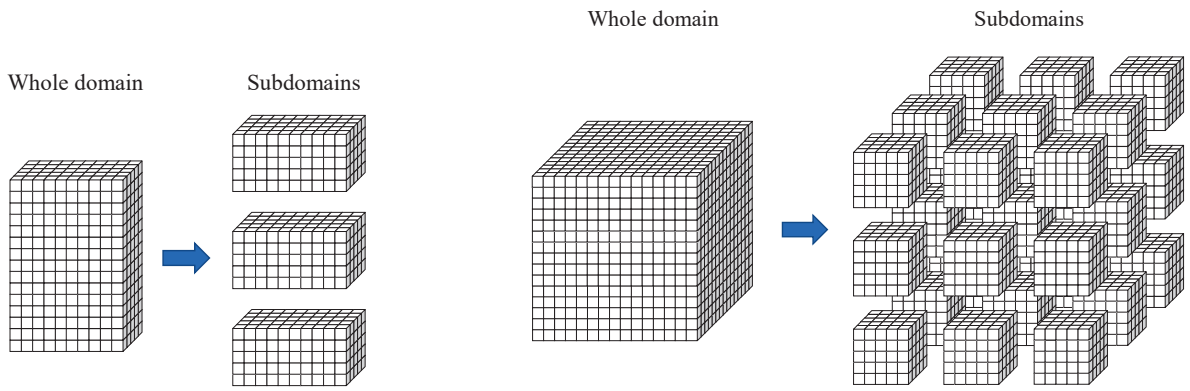
Figure 4.3: Thread hierarchy in CUDA programming.

4.2 Performance and Scalablity on GPUs

4.2.1 Basics of GPU Computing

In recent years, the use of GPU as computational accelerator has attracted wide attention. Powerful computing performance and high bandwidth for data transfer are the main features of GPU. At the same time, general programming environments such as OpenCL and CUDA (Compute Unified Device Architecture) have emerged, which greatly promote the application of GPU in many fields including scientific computing. In the field of high performance computing, multi-node GPU clusters have appeared in a large number of TOP 500, Graph 500 and Green 500 lists.

Stencil-based numerical methods such as the finite volume method naturally fit the framework of CUDA which is the programming interface on GPU. As shown in Fig. 4.3, to perform parallel computing on the background grid, blocks are allocated to take charge of their corresponding region of the grid. A certain number of threads inside every block will execute calculating commands simultaneously. Since thousands of CUDA cores are available in a GPU, it is flexible to arrange blocks and threads. Index of a thread is defined in a vector $threadIdx$ with three components, so the block composed of threads can be one-dimensional, two-dimensional or three-dimensional, and the size of each dimension of each thread block is defined by the variable $blockDim$. Although the number of threads in a thread block is limited, a kernel function in CUDA can be executed by multiple blocks of the same size. The structure composed of blocks is called grid. The grid of blocks can also be one-dimensional, two-dimensional or three-dimensional. The index of block in the grid is defined by vector $blockIdx$. Fig. 4.3 illustrates an example for block



(a) 1D partition (b) 3D partition

Figure 4.4: Domain partition for multi-GPU computation.

index in 2D grid and thread index in 2D block. Finally, when one thread is assigned for one grid point, the index of grid points (i, j, k) to be computed can be obtained by

$$\begin{aligned} i &= blockIdx.x * blockDim.x + threadIdx.x; \\ j &= blockIdx.y * blockDim.y + threadIdx.y; \\ k &= blockIdx.z * blockDim.z + threadIdx.z. \end{aligned} \tag{4.15}$$

Besides the thread hierarchy, there is memory hierarchy in CUDA programming. Every thread has its own local memory (registers), while shared memory is accessible by all threads in one block. And the global memory of GPU is visible to to all blocks and threads. More details about code optimization such as coalesced access of global memory and calculating occupancy are explained in [94].

The idea for parallel computing on multiple GPUs is to decompose the whole computational domain into several subdomains, each of them is computed by one GPU respectively, as shown in Fig. 4.4. However, in practical problems, the subdomains are not completely independent of each other. The common situation is that the data at boundaries of adjacent subdomain need to be accessed. A common approach for data communication between GPUs is the message passing interface (MPI). While only pointers to CPU (host) memory is accepted in a regular MPI implementation, the buffers for communication allocated in the device memory of GPU can be passed by using CUDA-aware MPI. CUDA-aware MPI not only simplifies the programming but also has the potential to accelerate the communication if GPUDirect P2P or RDMA is supported.

An effective approach to improve the performance of multi-GPU computation is hiding the communication time [47], which is realized by overlapping the kernel execution for inner cells and the data transfer for outer cells of the subdomain. With the help of CUDA stream technique, the outer cells that account for only a small fraction of the

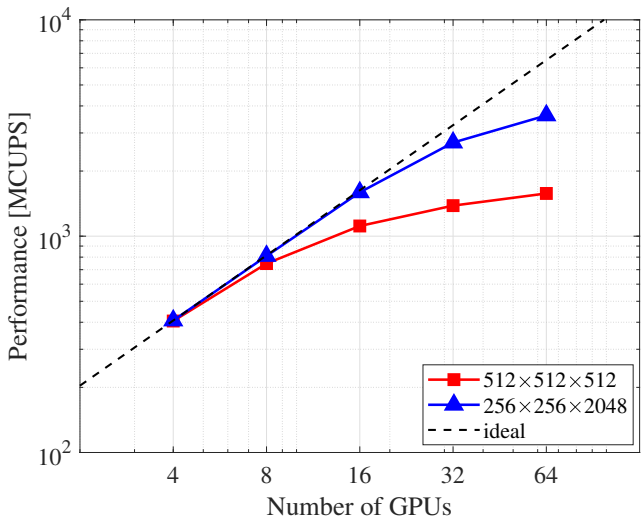


Figure 4.5: Strong scaling of Con-CAC-LS with 1D domain partition (overlapping).

total number of grids are first computed by the CUDA stream with the higher priority. And the computing kernel for inner cells is launched with a low-priority stream. Once the computation of the high-priority stream is completed, the communication process for outer cells is performed simultaneously with the inner computation. The overlapping technique has been implemented for the program with 1D domain partition which corresponds to Fig. 4.4(a). Currently, the 3D domain partition program corresponding to Fig. 4.4(b) does not support overlapping communication and computation due to the complexity of programming.

In this work, the NVIDIA Tesla V100 SXM2 on the FLOW Type II supercomputer is used to perform all of the computation. NVLink and InfiniBand EDR are deployed in this supercomputer for the intra-node and inter-node connection, respectively. The HPC SDK 21.2 is employed to compile the MPI+CUDA program.

4.2.2 Performance of Con-CAC-LS

In this section, the strong scaling, which concerns the speedup for a fixed problem size with respect to the number of processors, is evaluated for the present multi-GPU solvers. The performance is measured by the mega cells update per second (MCUPS). In the case of 1D domain partition with the overlapping technique, the amount of computation of each subdomain keeps decreasing as the number of GPUs increases, but the amount of data communication does not change. Hence the communication cannot be hidden anymore. As shown in Fig. 4.5, the computational performance does not increase proportionally with the number of GPUs. Moreover, different domain shapes with the same total number of meshes also affect the performance of 1D domain partition. The

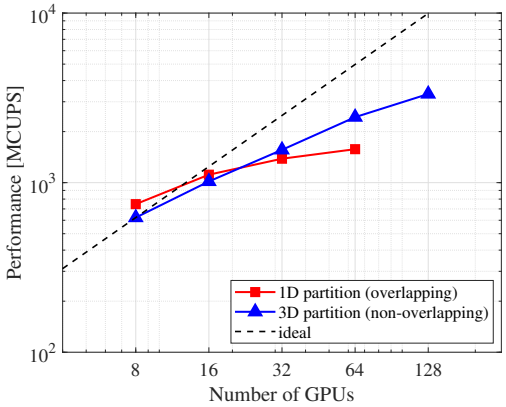


Figure 4.6: Comparison between 1D domain partition (overlapping) and 3D domain partition (non-overlapping) for the strong scaling of Con-CAC-LS on a $512 \times 512 \times 512$ grid.

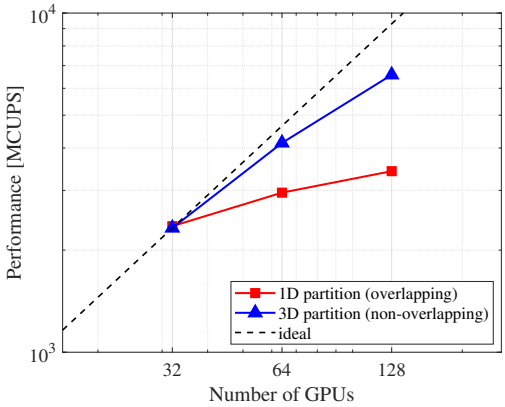


Figure 4.7: Comparison between 1D domain partition (overlapping) and 3D domain partition (non-overlapping) for the strong scaling of Con-CAC-LS on a $1024 \times 1024 \times 1024$ grid.

results show that strong scaling from 4 to 16 GPUs achieves even ideal performance in the rectangular domain, but behaves much worse in the cubic domain.

Fig. 4.6 and Fig. 4.7 show the comparison between 1D domain partition and 3D domain partition for the strong scaling of Con-CAC-LS in the cubic domain with different resolutions. Although the overlapping technique is not implemented in the program with 3D domain partition, the 3D partition demonstrates better scalability and higher performance on a large number of GPUs. This significant advantage comes from the variation of the surface area-to-volume ratio with the number of GPUs at different partitioning methods. When the computational domain is partitioned in multiple dimensions at the same time, each subdomain not only decreases in computation with smaller volume, but also decreases in data communication with smaller surface area instead of remaining constant as in the one-dimensional partition.

4.2.3 Performance of Con-PLIC-HF

For the dimensional-splitting method used in the transport part of the Con-PLIC-HF, the pattern of data communication is optimized accordingly. When the transport equation is solved in a certain dimension, data communication is performed only in the direction of this dimension. The strong scaling of Con-PLIC-HF with 3D domain partition in Fig. 4.8 and Fig. 4.9. Compared to Con-CAC-LS, Con-PLIC-HF has almost identical scalability but with higher overall performance. Although Con-PLIC-HF solves more transport equations, on the one hand, the analytical method greatly simplifies the computation of the interface geometry reconstruction, and on the other hand, it does not

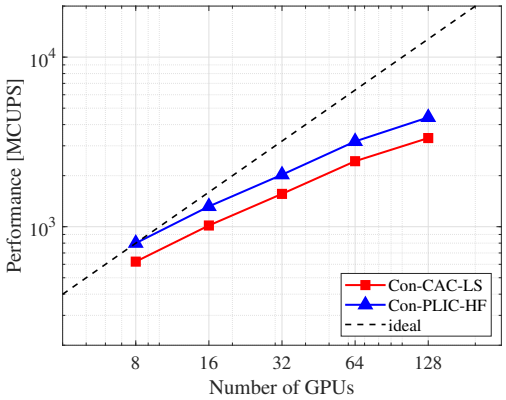


Figure 4.8: Comparison between Con-PLIC-HF and Con-CAC-LS with 3D domain partition (non-overlapping) for the strong scaling on a $512 \times 512 \times 512$ grid.

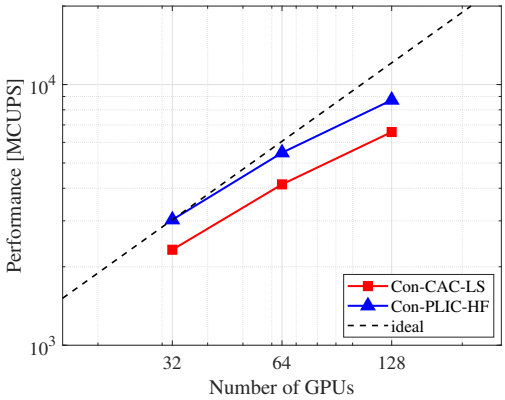


Figure 4.9: Comparison between Con-PLIC-HF and Con-CAC-LS with 3D domain partition (non-overlapping) for the strong scaling on a $1024 \times 1024 \times 1024$ grid.

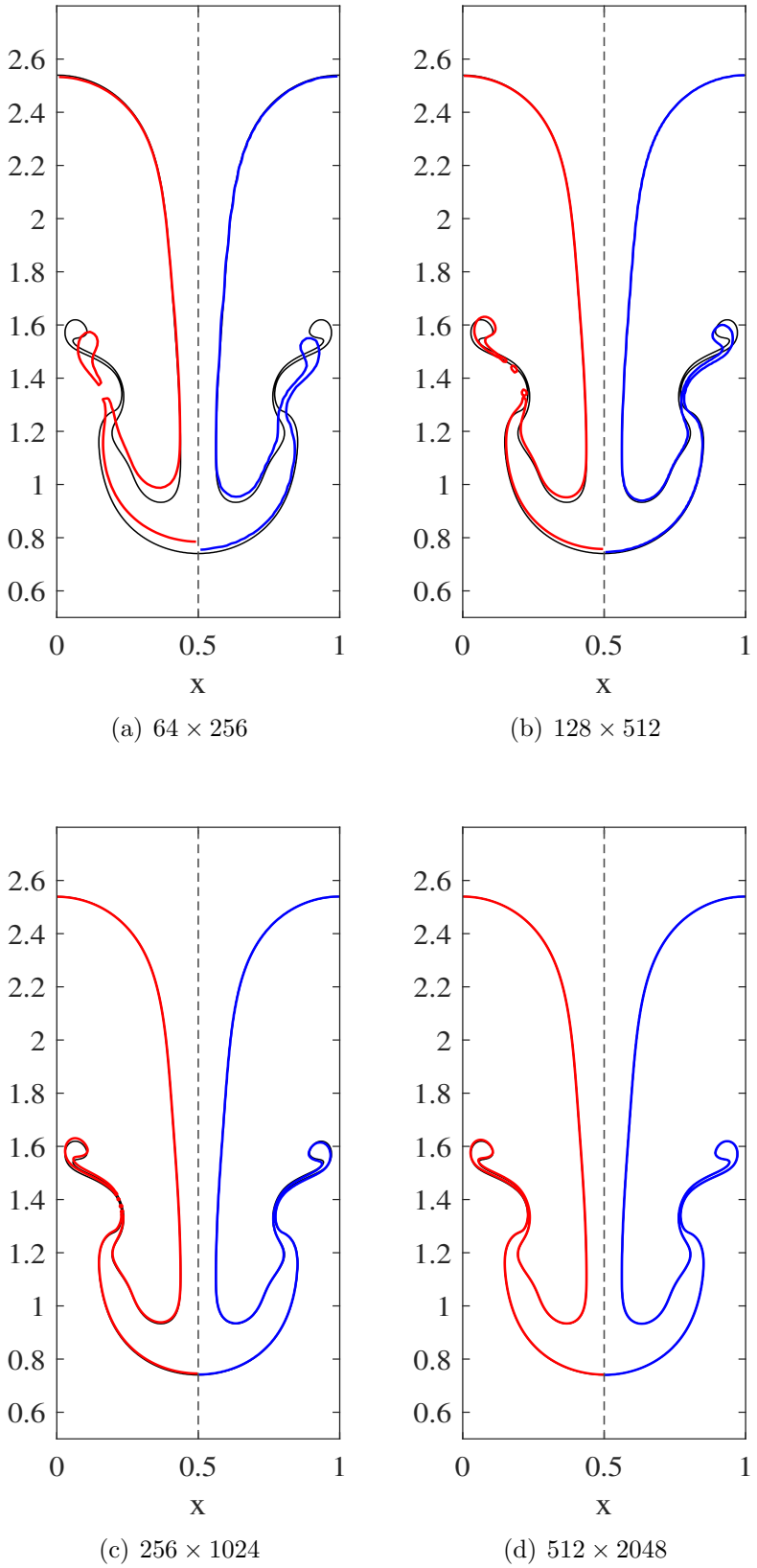
use the higher-order Runge-Kutta time integration, which multiplies the computational effort, as Con-CAC-LS does.

4.3 Numerical Results

4.3.1 Rayleigh-Taylor Instability

The first numerical result is to discuss the mesh dependency of the two-phase flow simulation. A classical two-phase flow phenomenon known as Rayleigh-Taylor instability is simulated. The computation conditions described in a reference paper [95] is followed here. Initially the heavy fluid with density $\rho_h = 1.225$ is placed on top of the light fluid with density $\rho_l = 0.1694$. The viscosity of both fluids is $\mu_h = \mu_l = 0.00313$. The two-phase interface is horizontally positioned at the center of a 1×4 rectangular domain and perturbed with a cosine wave of amplitude 0.05. This problem is free of the surface tension force, and only the gravity $g = 9.81$ is applied. At the beginning, the fluid in the whole domain is at rest. Slip conditions are imposed on the enclosing walls.

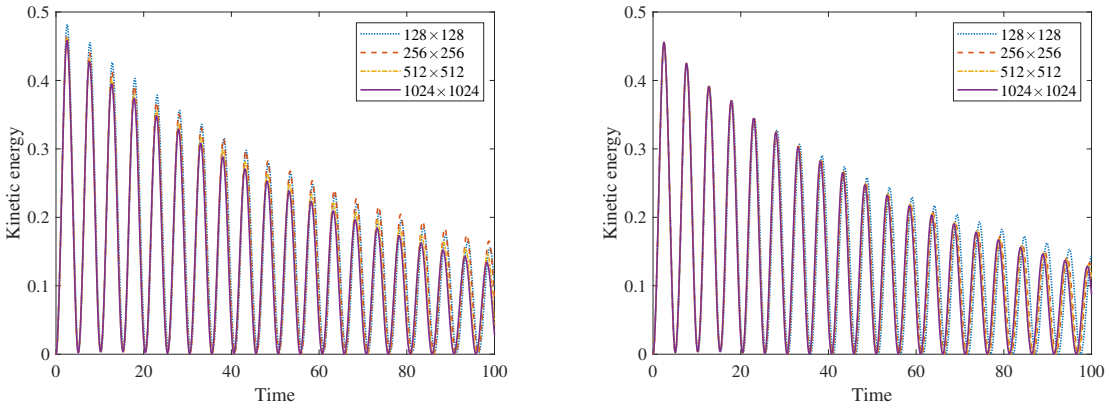
The interface location of the Rayleigh-Taylor instability at $t = 0.9$ is depicted in Fig. 4.10 with four different mesh resolutions including 64×256, 128×512, 256×1024 and 512×2048. Because Con-PLIC-HF produces identical interfaces on the 256×1024 and 512×2048 grids, it is regarded as the mesh-independent result and choose this interface profile as the reference solution. However, Con-CAC-LS only becomes consistent with the reference solution on the 512×2048 grid. The results indicate that Con-PLIC-HF has a better mesh convergence than Con-CAC-LS and is more accurate when the mesh



(a) 64×256 (b) 128×512

(c) 256×1024 (d) 512×2048

Figure 4.10: Interface location of the Rayleigh-Taylor instability at $t = 0.9$ predicted by Con-CAC-LS (red line), Con-PLIC-HF (blue line) and the reference solution (black line) with different mesh resolutions.



(a) Con-CAC-LS (b) Con-PLIC-HF

Figure 4.11: Comparison of the evolution of total kinetic energy in the drop oscillation.

resolution is low.

4.3.2 Drop Oscillation with Surface Tension Force

Next, a drop oscillation problem driven by surface tension force is investigated. This problem is widely used to verify the accuracy of surface force models. The same computational setting as in [96, 97] is used, where a 20^2 square region is discretized by the 128×128, 256×256, 512×512, 1024×1024 uniform grid. The initial drop interface is an ellipse defined by the equation,

$$(x-10)^2/9 + (y-10)^2/4 = 1. \tag{4.16}$$

The density and viscosity of the drop are 1 and 0.01 respectively, and the fluid outside the drop has density 0.01 and viscosity 5×10^{-5}. This problem is gravity-free, only surface tension force is applied. The surface tension coefficient is $\sigma = 1$. At the beginning, the fluid in the whole domain is at rest. Slip conditions are imposed on the enclosing walls.

The evolution of the total kinetic energy $\frac{1}{2}\int \rho |\boldsymbol{u}|^2 dV$ of the computational domain with the two solvers at difference mesh resolutions is plotted in Fig. 4.11. The frequencies of the oscillation obtained by both solvers are basically the same and are close to the values reported in the references [96, 97]. However, the curvature inside a narrow band of interface estimated by the Con-CAC-LS solver has a lot of overshoot and undershoot, as shown in Fig. 4.12. In contrast to this, the curvature calculated by the Con-PLIC-HF solver at the interfacial cells varies smoothly and coincides with the shape of the ellipse. The numerical results indicate the Con-PLIC-HF solver has better mesh convergence and accuracy compared to the Con-CAC-LS solver as illustrated in Fig. 4.11 when the effect of surface tension exists.

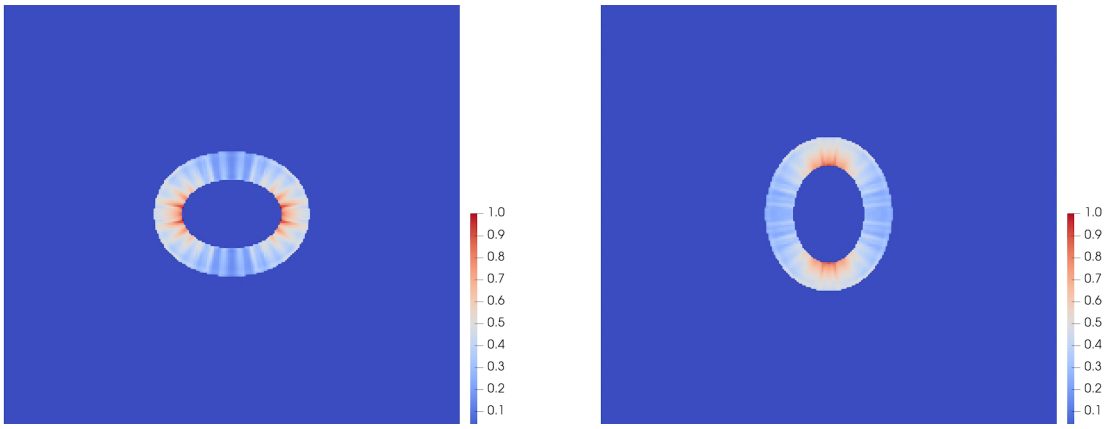


Figure 4.12: Interface curvature estimated by Con-CAC-LS solver.

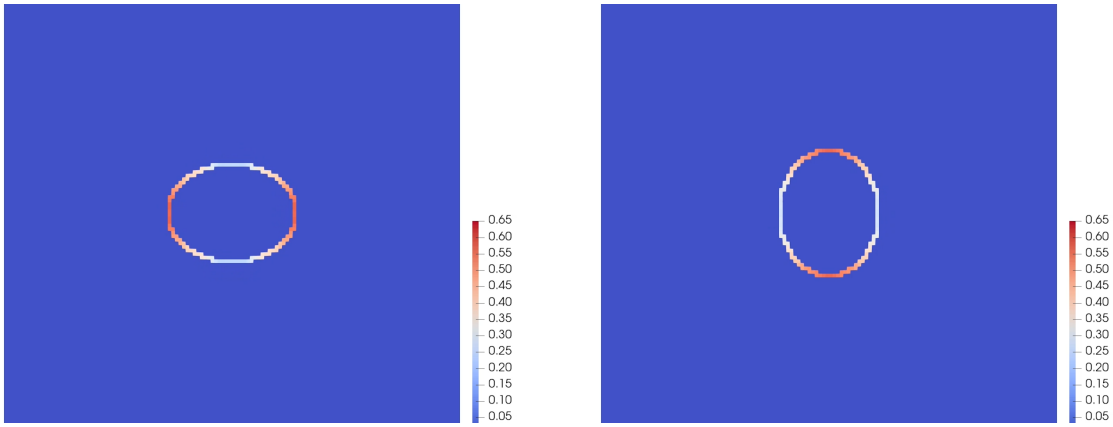
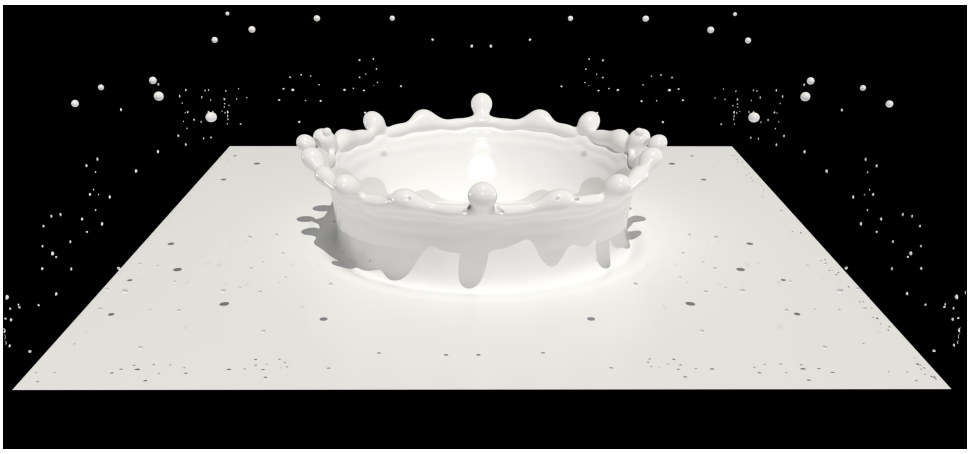


Figure 4.13: Interface curvature estimated by Con-PLIC-HF solver.

4.3.3 Drop Impacting on a Thin Liquid Film

To demonstrate the accuracy of the present methods in resolving three-dimensional interface topology, the problem of a drop impacting on a thin liquid film is solved and a classical phenomenon known as milk crown is reproduced. This problem has been extensively studied by numerical simulation, experiment and theoretical analysis [**yarin2006drop**]. The water tank considered measures 60 mm $\times$ 60 mm $\times$ 20 mm. Considering the symmetry of this computation, only a quarter of the water tank is calculated and discretized by a $768 \times 768 \times 512$ uniform grid. The depth of the thin liquid film is 0.82 mm. Initially the liquid film and air are at rest. A droplet of diameter $D = 7.04$ mm is falling with the speed of $V = 1.61$ m/s. As a consequence, the diameter of the drop is resolved by about 90 grid cells. The physical properties of the liquid and gas are $\rho_h = 1000$ kg/m^3, $\rho_l = 1$ kg/m^3, $\mu_h = 1.0 \times 10^{-3}$ Pa·s, $\mu_l = 1.0 \times 10^{-5}$ Pa·s, and surface tension coefficient $\sigma = 0.07275$ N/m. The Weber number is $We = \rho_h DV^2/\sigma = 250$ in this case. No-slip

(a) $t = 2$ ms



(b) $t = 16$ ms

(c) $t = 32$ ms

Figure 4.14: Results of 3D drop impacting on liquid film by Con-CAC-LS solver.

(a) $t = 2$ ms

(b) $t = 16$ ms

(c) $t = 32$ ms

Figure 4.15: Results of 3D drop impacting on liquid film by Con-PLIC-HF solver.

conditions are imposed on all walls. This computation is performed on 18 V100 GPUs.

The time evolution of the liquid surface is depicted in Fig. 4.14 and Fig. 4.15 for two solvers, respectively. By comparing the results, Con-PLIC-HF solver can resolve the fingering structure of the interface very well. On the other hand, Con-CAC-LS solver produces a lot of splashing drops and the free rim also breaks up. But these phenomena should not occur at this low Weber number case. As for the computational efficiency, it takes 6 hours 40 minutes for Con-PLIC-HF solver and 12 hours 30 minutes for Con-CAC-LS solver to complete the simulation.

4.3.4 Dam Break on a Wet Bed

To demonstrate the stability of the present solver as a practical two-phase flow solver, the problem of a three-dimensional dam break on a wet bed is simulated. Experimental studies on this problem have been reported by Stansby et al. [73]. The low-resolution simulation is also reported in Section 2.3.7 and my previous work [98]. The water tank considered measures 0.72 m $\times$ 0.12 m $\times$ 0.36 m and is discretized by a $1536 \times 256 \times 768$ uniform grid. Enclosed by a virtual baffle plate located at $x = 0.15$ m, a water column with the same width and height is placed at the left side of the tank, as shown in Fig. 4.16(a). The depth of the water on the flat bed is 0.018 m. Initially the water and air are at rest. The physical properties of the water and air are $\rho_h = 998$ kg/m^3, $\rho_l = 1.2$ kg/m^3, $\mu_h = 1.0 \times 10^{-3}$ Pa·s, $\mu_l = 1.8 \times 10^{-5}$ Pa·s, and surface tension coefficient $\sigma = 0.07275$ N/m. The gravitational acceleration is $g = 9.8$ m/s^2. No-slip conditions are imposed on all walls.

The time evolution of the water surface is depicted in Fig. 4.16 and Fig. 4.17. The virtual baffle plate is abruptly removed at the beginning of the simulation, and the water column begins to collapse under gravity. A rolling surface of water over the wet bed appears at $t = 0.2s$ and air bubbles mix into the water at $t = 0.4s$. After the water crashes against the downstream wall, a lot of droplets splash out at $t = 0.6s$. In this simulation, the high mesh resolution makes it possible to reproduce small-scale bubbles and droplets realistically.

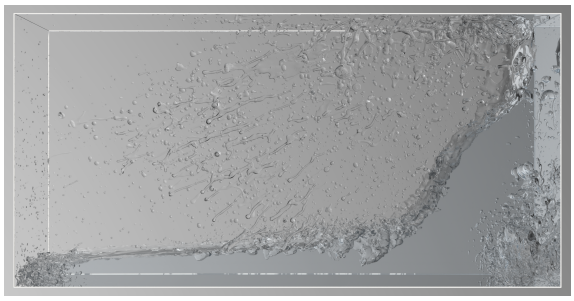
4.3.5 Liquid Jet in Gas Cross-flow

The last simulation presented in this chapter is a three-dimensional liquid jet in gas cross-flow. This problem has been studied in a existing literature [87] and also used by us to verify the importance of consistent transport and momentum conservation [99]. The jet column predicted by a non-conservative method will break up within a short distance after leaving the nozzle, affected by the gas cross-flow. And the droplets formed after

(a) $t = 0s$

(b) $t = 0.2s$

(c) $t = 0.4s$

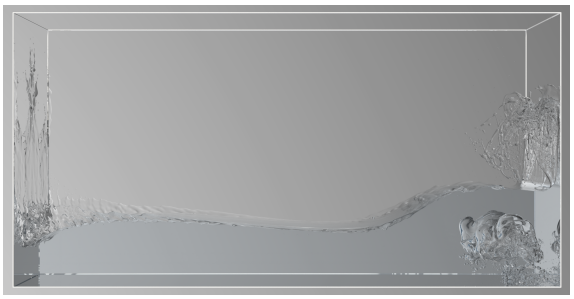


(d) $t = 0.6s$

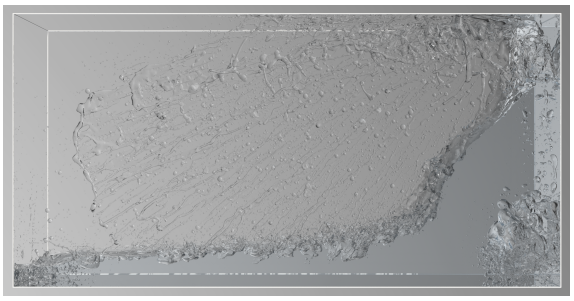
Figure 4.16: Time evolution of water surface in 3D dam break on a wet bed simulated by Con-CAC-LS.

(a) $t = 0s$

(b) $t = 0.2s$

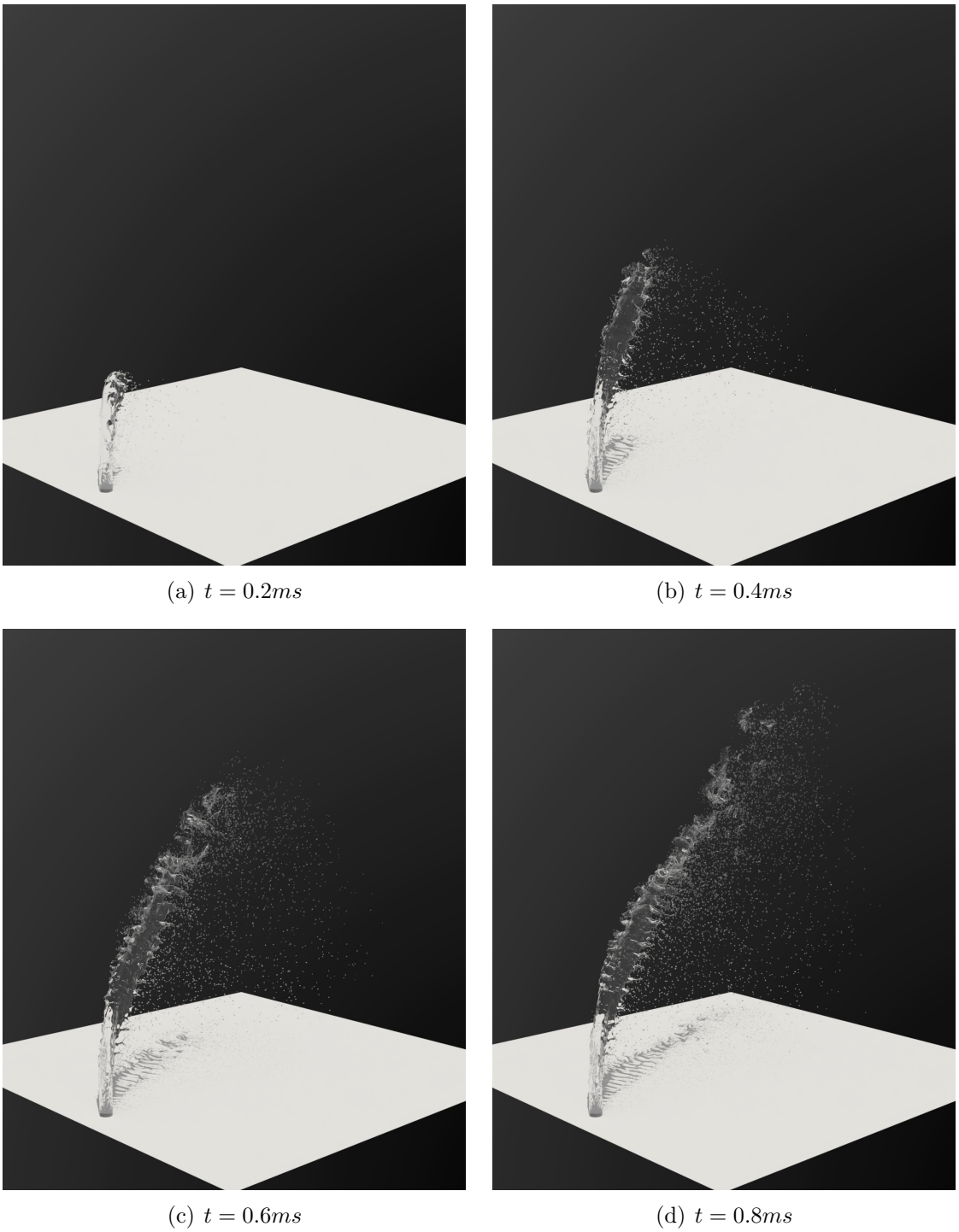


(c) $t = 0.4s$



(d) $t = 0.6s$

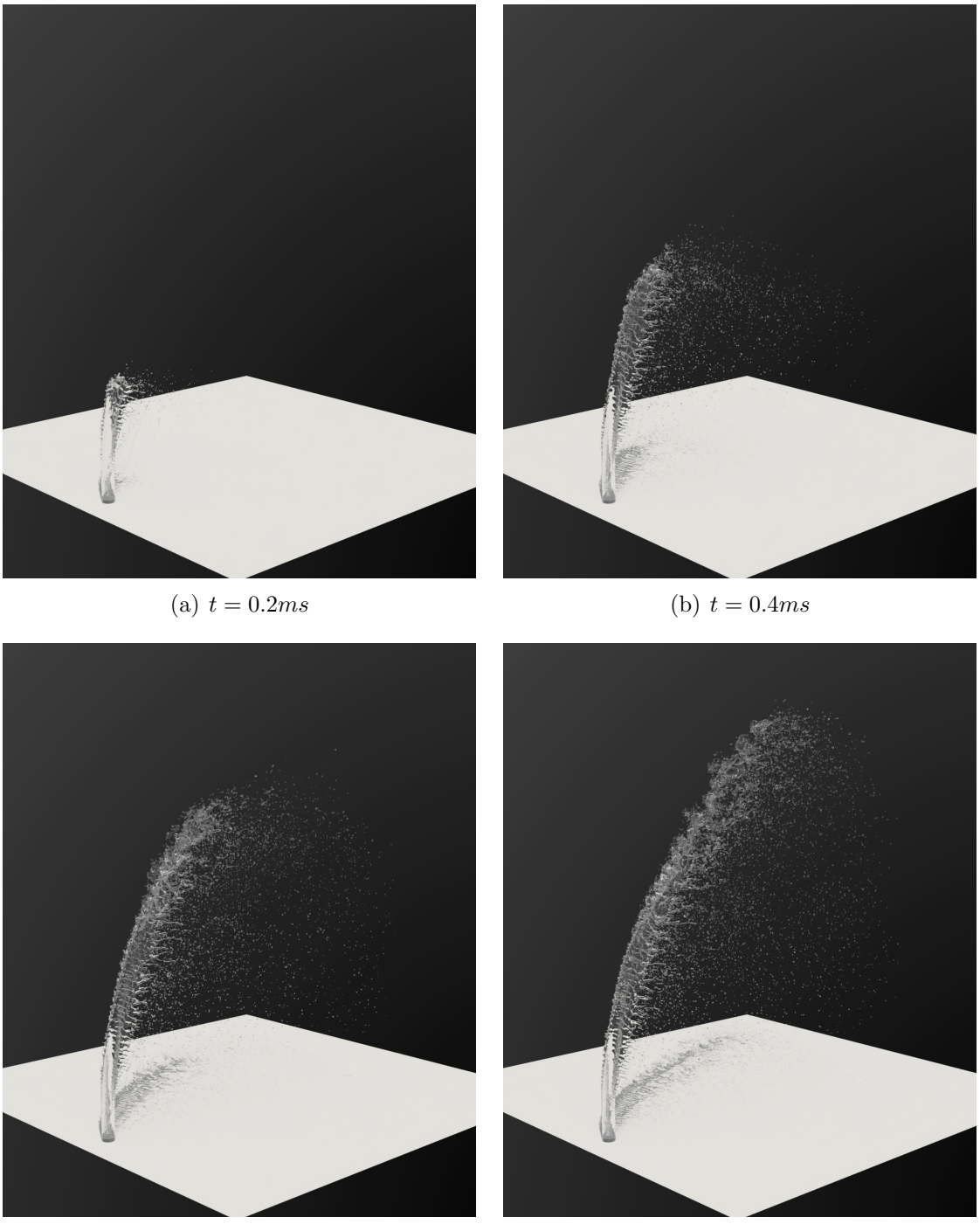
Figure 4.17: Time evolution of water surface in 3D dam break on a wet bed simulated by Con-PLIC-HF.



(a) $t = 0.2ms$ (b) $t = 0.4ms$

(c) $t = 0.6ms$ (d) $t = 0.8ms$

Figure 4.18: Time evolution of liquid jet surface in gas cross-flow simulated by Con-CAC-LS.



(a) $t = 0.2ms$ (b) $t = 0.4ms$

(c) $t = 0.6ms$ (d) $t = 0.8ms$

Figure 4.19: Time evolution of liquid jet surface in gas cross-flow simulated by ConPLIC-HF.

breaking up are quickly blown to the downstream region with the gas. On the contrary, the conservative solver can produce more accurate simulation results on the interface shape and trajectory of the jet.

The computational domain size measures 2.0 cm $\times$ 1.5 cm $\times$ 3.5 cm and is discretized by a $1024 \times 768 \times 1792$ uniform grid. The liquid nozzle of diameter 0.8 mm is placed on the bottom wall and centered at $(0.2 \text{ cm}, 0.75 \text{ cm}, 0 \text{ cm})$. The gas blows in from the left boundary in the positive x direction with a constant velocity V_g, and the liquid is injected from the nozzle along the positive z direction with a constant velocity V_l. The physical properties are same as the Case 2 in Table 3.3, which corresponds to the actual value for water and air. No-slip conditions are imposed on the bottom wall $z_{\min}$ except for the nozzle. The outflow conditions are applied to the right boundary $x_{\max}$. Other walls at $y_{\min}$, $y_{\max}$ and $z_{\max}$ have the slip conditions. This computation is performed on 42 V100 GPUs.

Numerical results for the time evolution of the liquid jet are depicted in Fig. 4.18 and Fig. 4.19. It can be seen from the figure that Con-PLIC-HF maintains the water jet pattern and trajectory better. In this large-scale simulation with the Con-PLIC-HF solver, a great number of liquid drops with different sizes are successfully resolved. The trajectory of the liquid column is also in good agreement with other numerical simulations and an experimental study [85].

Chapter 5

Spatially Sparse Grid Structure on GPU

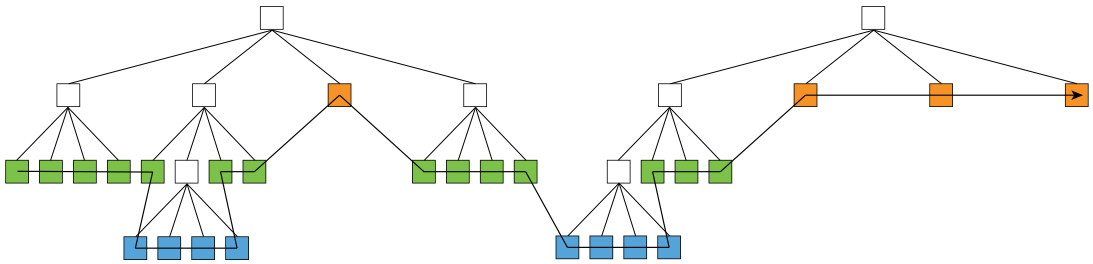
This chapter briefly introduces the implementation of complex data structure for sparse volumetric representation of the computational domain with GPU computing. The data structures are designed to have a flexible topology and efficient data access for fluid simulations.

5.1 Adaptive Mesh Refinement

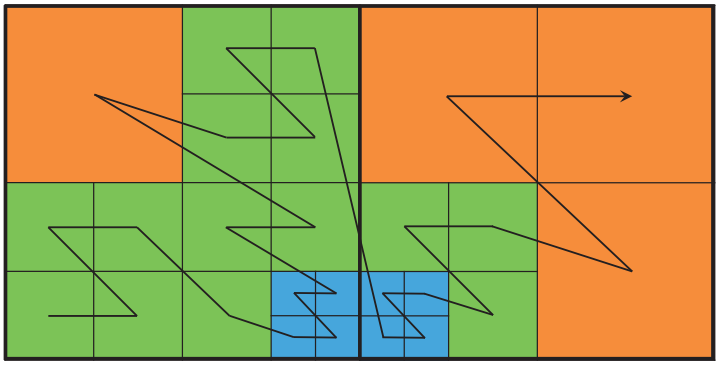
5.1.1 Data Structure and Algorithms in AMR

Along with the progress of computer technology like GPU, the study on implementation of complicated algorithms on advanced architecture of hardware is demanded. In the simulation of fluid flow, wide range of spatial scales are often encountered, including shocks in compressible flows, interfaces between immiscible liquids, turbulence intermittency, boundary layers and vorticity generation near solid boundaries. A straightforward solution is to adjust the mesh resolution to follow the evolution of flow structure, which is the so-called adaptive mesh refinement. In this thesis, a block-structured cell-centered AMR algorithm is implemented on GPU. The overall design and several important techniques are introduced here.

The smallest entity used to assign computational task is called a block, which is a cube composed of three dimensional cells. The computational domain is decomposed into blocks of different sizes in AMR. A data structure based on the forest of octrees (quadtrees in 2D) is implemented, as shown in Fig. 5.1(a). A block in a lower hierarchical level of octree can be subdivided into eight (four in 2D) smaller blocks of equally size in a higher level in the refining process. Conversely, Eight children blocks in a higher level will be



(a) Forest of quadtrees



(b) Space filling curve

Figure 5.1: Tree-Based Block-Structured AMR.

merged to one parent block in a lower level in the coarsening process. As a consequence, the effective blocks are actually the leaves in the forest of octrees. They are stored in the order following a space filling curve manner [100]. The Morton curve which is one of space filling curves can be constructed by depth first search (DFS) of the whole forest of octrees. The computational grid corresponding to the octrees is illustrated in Fig. 5.1(b), where the 2:1 balance between neighboring blocks is satisfied, i.e., the maximum allowable difference of level is 1 between neighboring blocks. The approach to enforce the 2:1 balancing condition is briefly introduced here. When a block meets the refinement criterion and is going to be refined, all neighboring blocks that have a lower depth in the octree are refined. In contrast, when a block doesn't meet the refinement criterion and is going to be coarsened, the coarsening process shouldn't executed if any of its neighboring block has a higher depth.

In actual implementation, the structures of octrees and the information of computational blocks are managed separately, while some informations of the latter are generated from the former such as the neighbor list of each block. However, simulation data is never stored in the structure of octrees. Same as the Daino framework [101], the tree structure is set on host memory and managed by CPU, All the data of grid blocks are stored on device memory and the stencil computation is performed by GPU.

The search of neighbors is realized through an ingenious design of the binary index of

<table>
<tr><td colspan="2" rowspan="2">X: 0
Y: 1</td><td>10
11</td><td>11
11</td><td colspan="2" rowspan="2">0
1</td><td rowspan="2">1
1</td></tr>
<tr><td>10
10</td><td>11
10</td></tr>
<tr><td>00
01</td><td>01
01</td><td>10
01</td><td colspan="2">11
01</td><td>00
01</td><td>01
01</td><td rowspan="3">1
0</td></tr>
<tr><td rowspan="2">00
00</td><td rowspan="2">01
00</td><td rowspan="2">10
00</td><td>110
001</td><td>111
001</td><td>000
001</td><td>001
001</td><td rowspan="2">01
00</td></tr>
<tr><td>110
000</td><td>111
000</td><td>000
000</td><td>001
000</td></tr>
</table>

Figure 5.2: Binary index of leaf nodes of trees in AMR.

nodes of trees, as illustrated in Fig. 5.2. The binary index is composed of two in 2D or three in 3D unsigned integers, and each corresponds to a spatial dimension. Considering a parent node with index *my_node_id*, it has 8 children nodes in three dimensions. For the ith child where $i \in [0, 7]$, the binary index *child_node_id* is calculated by

$$\begin{aligned} child\_node\_id.x &= (my\_node\_id.x << 1)|(i\&1); \\ child\_node\_id.y &= (my\_node\_id.y << 1)|((i >> 1)\&1); \\ child\_node\_id.z &= (my\_node\_id.z << 1)|((i >> 2)\&1). \end{aligned} \tag{5.1}$$

According to this pattern, for example, the binary index of a neighbor node at same level in the positive direction of x axis is obtained by adding 1 to $my\_node\_id.x$.

5.1.2 Interpolation for Cell-Centered Data

As described in previous subsection, when a block is to be refined, every cell in this block will be subdivided into 4 in 2D or 8 in 3D smaller cells. Data in these fresh cells should be set artificially. In this thesis, volumetric formulation for the cell-centered data is adopted, hence a linear interpolation scheme that satisfies the conservation law can be constructed as follows:

$$f^f = f^c + (\boldsymbol{r}_f - \boldsymbol{r}_c) \cdot \nabla f^c, \tag{5.2}$$

where the superscripts r and c represent the fine cell and coarse cell, respectively. $\boldsymbol{r}_f$ and $\boldsymbol{r}_f$ are the positional vectors of cell centers, as illustrated in Fig. 5.3. ∇f^c is the local gradient of data in coarse cells, which can be calculated by the central difference scheme. This interpolation scheme works as distributing the data in coarse cell to fine cells according to the gradient.

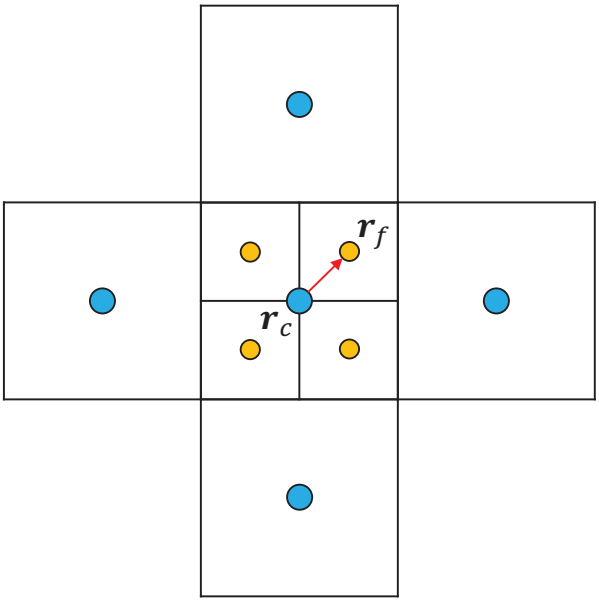
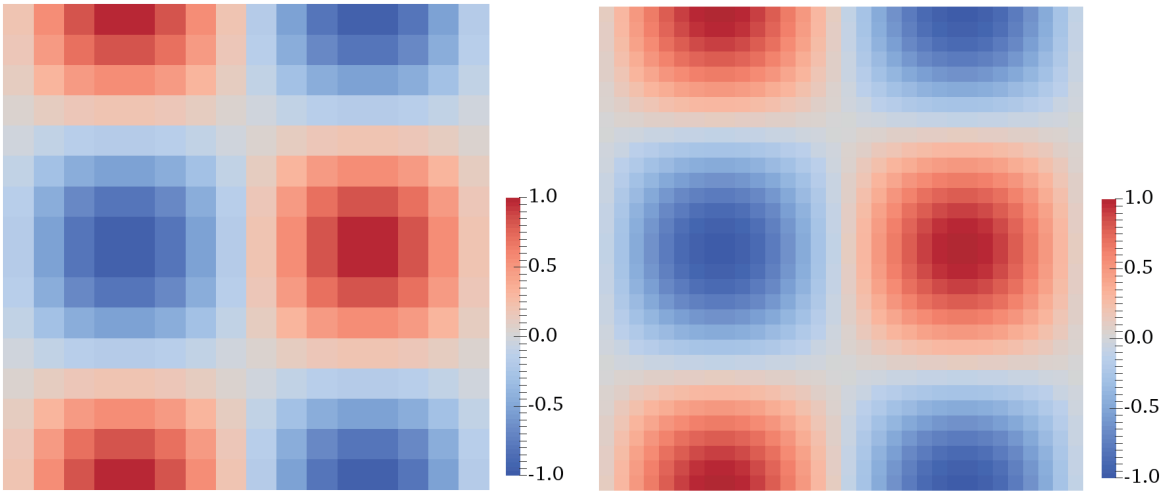


Figure 5.3: Interpolation for cell-centered data in mesh refinement.



(a) Before interpolation (b) After interpolation

Figure 5.4: Interpolation for a smooth periodic profile from 16×16 grid to 32×32 grid.

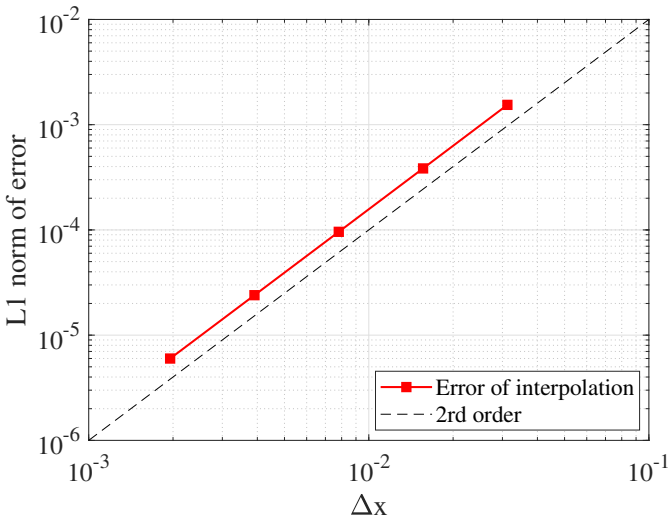


Figure 5.5: L1 error of present interpolation scheme.

To verify the effectiveness and accuracy of this interpolation scheme, a refinement test is shown here. In a periodic unit square, the data to be interpolated is initialized by the following profile:

$$f(x, y) = \sin(2\pi x)\cos(2\pi y). \tag{5.3}$$

where $x, y \in [0, 1]$. If the initial mesh has a size of $N \times N$, the size of resulting mesh after interpolation will be $2N \times 2N$. An example of interpolating the data in 16×16 grid to 32×32 grid is depicted in Fig. 5.4. The profile of data becomes much smoother after interpolation. Based on the difference between the values obtained by interpolation and the analytical values given by Eq. (5.3), the L1 error of present interpolation scheme can be plotted as function of mesh resolution, as shown in Fig. 5.5.

When a block is to be coarsened, the data located at center of a coarse cell should be calculated by its corresponding fine cells. According to the conservation law in volumetric formulation, the identity is expressed as:

$$f^c(\boldsymbol{r}_c)\Delta V^c = \sum_{i=0}^{n-1} f^f(\boldsymbol{r}_f)\Delta V^f, \tag{5.4}$$

where ΔV^c and ΔV^f are the volumes of a coarse cell and a fine cell. $n = 4$ in 2D and 8 in 3D. $\Delta V^c = n\Delta V^f$ is another identity. Therefore, f^c can be easily constructed by taking the average of f^f in fine cells:

$$f^c(\boldsymbol{r}_c) = \frac{1}{n}\sum_{i=0}^{n-1} f^f(\boldsymbol{r}_f). \tag{5.5}$$

In the flow simulation with immersed object, two criteria are used to judge whether a block is to be refined or coarsened. The first criterion is the distance to the solid object,

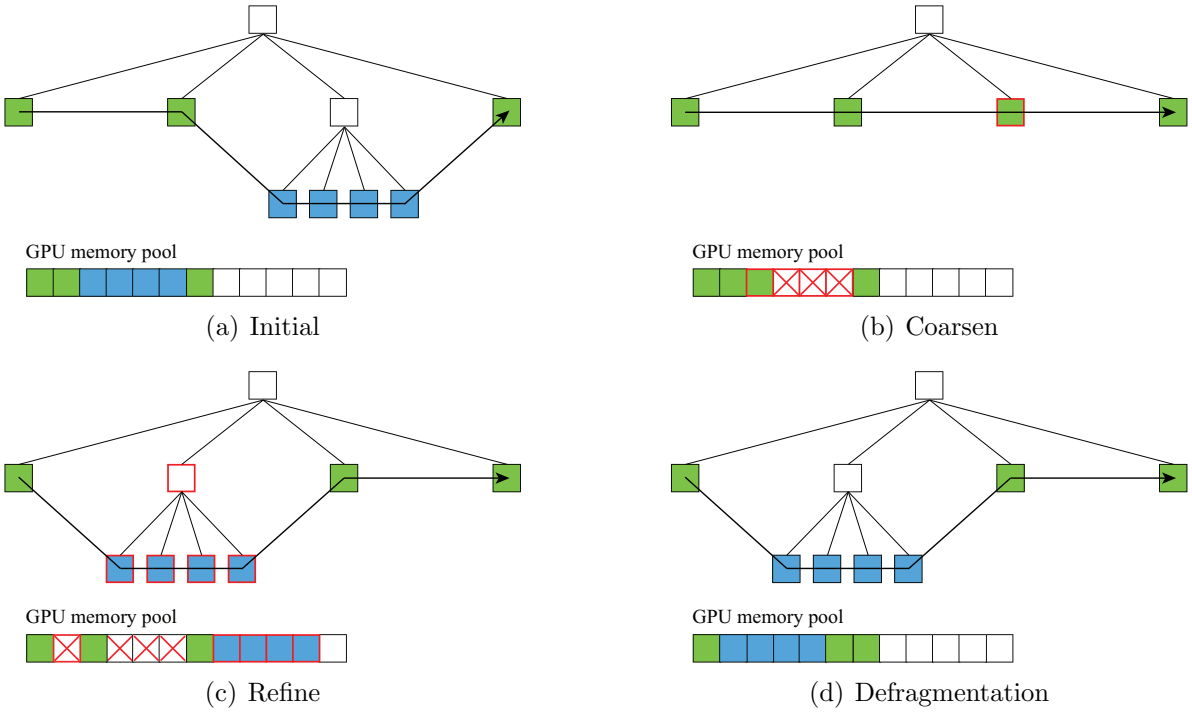


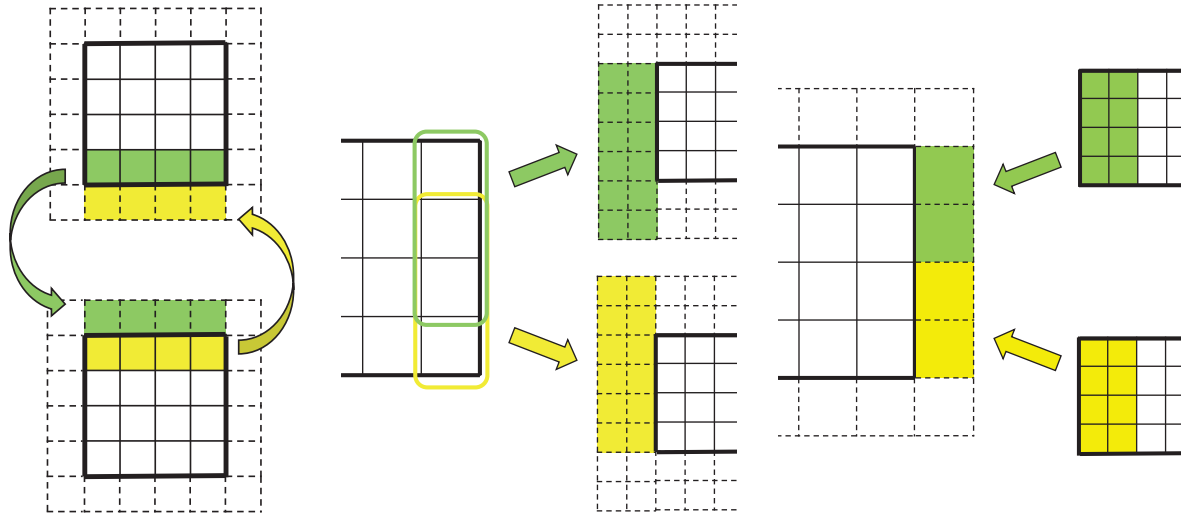
Figure 5.6: Management of GPU memory during mesh adaptation.

which is given by the level set function ψ, while the second criterion is based on the norm of the local vorticity vector $|\nabla \times \boldsymbol{u}|/|\boldsymbol{u}|_{\max}$.

5.1.3 GPU-Based AMR Implementation

In present block-structured AMR, one block may have $8 \times 8 \times 8$ inner cells and 2 outer layers of halo cells. And the simulation data of cells belonging to the same block is continuous in global memory. It is reasonable to assign one CUDA block of threads to process one AMR block. Hence the amount of CUDA blocks is equal to the number of leaves in the forest of octrees. The difficulty of GPU-based AMR implementation mainly lies in the management of memory when mesh adaptation occurs. New memory space is required when a block is refined to multiple children blocks, while other memory spaces may be idle due to children blocks are merged into a block. The strategy for memory management is briefly explained in this section.

As shown in Fig. 5.6(a), this tree begins with 7 leaves, with 3 leaves in level 1 and 4 leaves in level 2. The data in each block is arranged in order of space filling curve in the memory pool. When those 4 blocks in higher level are coarsened to 1 block in lower level, data of this new block will occupy the memory space of its first child block, shown in Fig. 5.6(b), while the memory spaces of another 3 blocks become unused. In Fig. 5.6(c), another block in level 1 is refined to 4 blocks in level 2. Because there is not always a



(a) Copy (b) Explosion (c) Coalescence

Figure 5.7: Communications for distribution functions of LBM at the boundaries of blocks.

continuous space in the middle of the memory pool to insert data in 4 new blocks, the data of blocks generated by refinement is always stored in the end of memory pool, A larger memory space can be reallocated if necessary. However, after many refinements and coarsening, the memory pool will be flooded with idle space and become fragmented. Hence a defragmentation operation is required to organize the memory pool, as shown in Fig. 5.6(d). This is achieved by storing all blocks in order of space filling curve in a newly allocated memory pool, then release the space of previous one.

To effectively fetch the data from neighboring blocks used for stencil computation, three kinds of communication at the boundaries of blocks are performed in every computational step. Similar to the approach described in [45], the halo region is constructed for each block with two layers of cells. For the sake of brevity, only communication across face of neighboring blocks is explained here, more details on the data transfer across edge and corner of the cubic block can be found in [102]. Copy process (see Fig. 5.7(a)) is used for the communication between neighboring blocks at same level. Boundary data in a block are directly sent to the halo layer of its neighbor. Explosion process (see Fig. 5.7(b)), i.e., the coarse-to-fine communication, sends necessary parts of the coarse block to the halo layers of finer neighbors. Some coarse cells must be sent to more than one fine neighbor. The distribution functions are homogeneously distributed to corresponding fine cells. In contrast to copy and explosion processes, coalescence process (see Fig. 5.7(c)) reads data from the finer block and writes data into the halo region of the coarse block. An averaging operation is performed for distribution functions of finer cells when

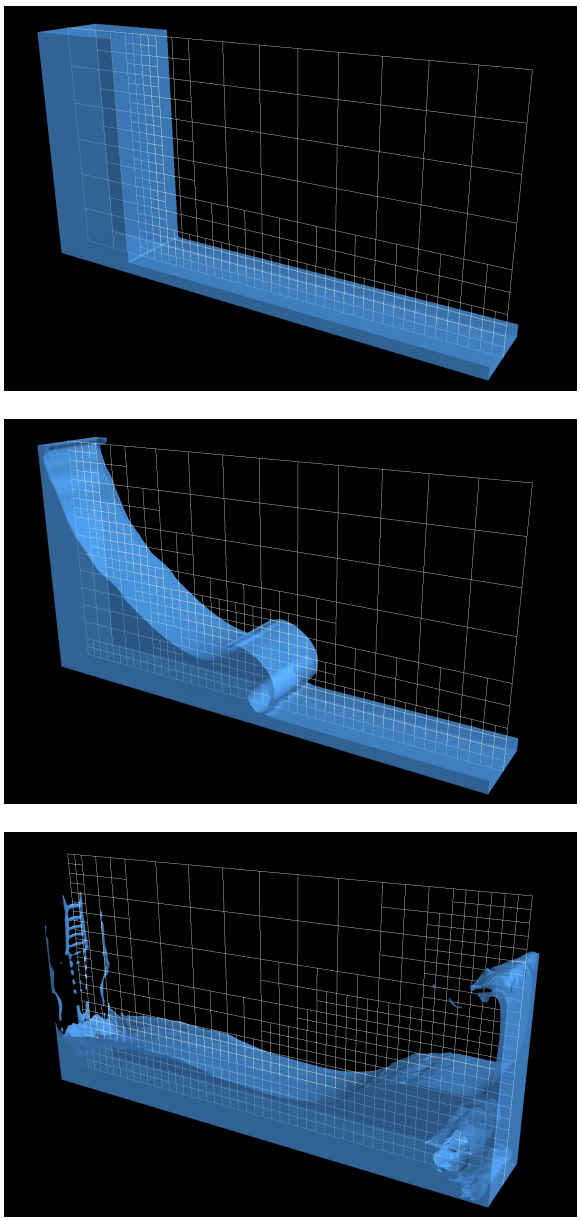


Figure 5.8: Dam break simulation with adaptive mesh refinement.

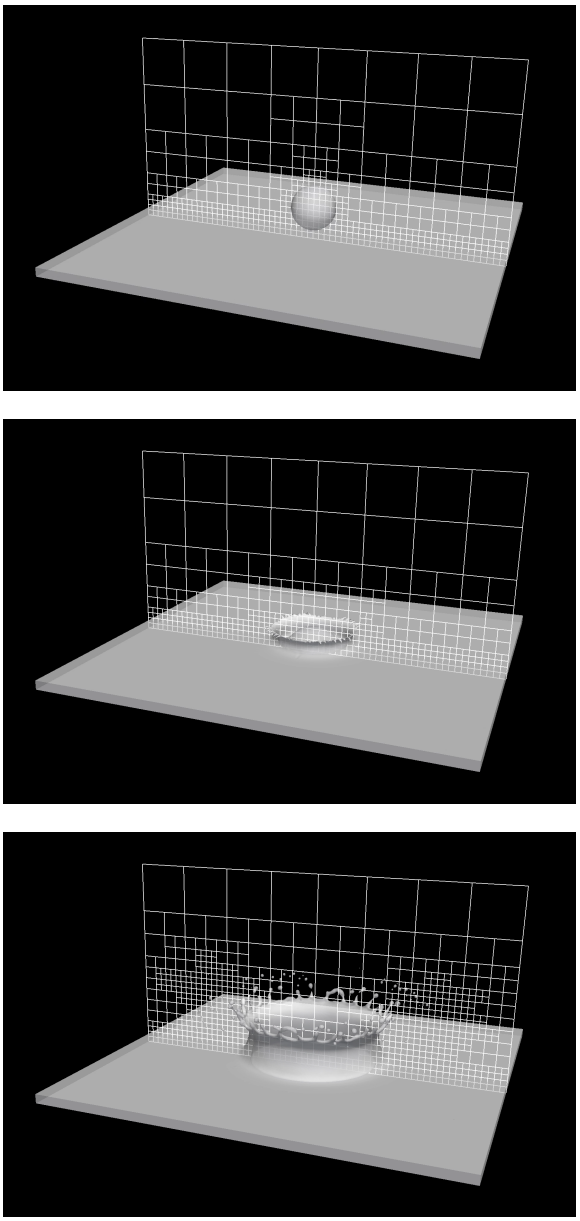


Figure 5.9: Drop splashing simulation with adaptive mesh refinement.

merged to a coarse cell.

5.2 Numerical Simulation with AMR

Two numerical examples of two-phase flow simulations by using the conservative solver with adaptive mesh refinement are presented in this section. Currently, AMR computation is only supported on single GPU.

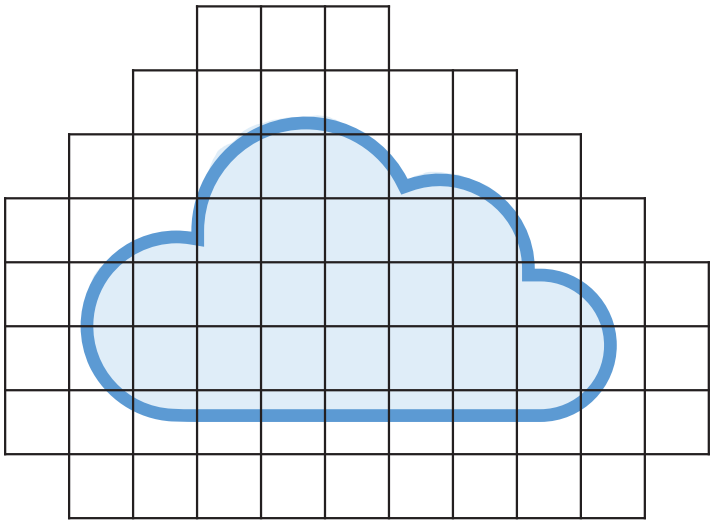


Figure 5.10: Sparse grid for the region of interest.

The fist example is the dam break on a wet bed. The same problem setting as Section 4.3.4. To accommodate the geometry of the computational domain which measures 0.72 m $\times$ 0.12 m $\times$ 0.36 m, $6 \times 1 \times 3$ tree roots are assigned to the forest of octrees. The minimum and maximum level (depth) of octrees are 1 and 3, respectively. It means the computational domain is discretized by a minimum of $96 \times 16 \times 48$ meshes and a maximum of $384 \times 64 \times 192$ meshes. The refinement criterion is set as the distance to the two-phase interface. The interface evolution along with mesh adaptation is depicted in Fig. 5.8.

The second example is the drop splashing on a thin liquid film. Similar to the problem setting of Section 4.3.3, the domain size is slight smaller with 40 mm $\times$ 40 mm $\times$ 20 mm. Therefore, the forest contains $2 \times 2 \times 1$ octrees. The minimum and maximum level of each octree are 2 and 5, respectively. And the level 2 corresponds to a $64 \times 64 \times 32$ grid and the level 5 corresponds to a $512 \times 512 \times 256$ grid. The numerical result is depicted in Fig. 5.9.

5.3 Sparse Grid Based on Spatial Hash

Another type of spatially sparse volumetric data structure is the sparse grid represented by VDB [103], which is wide used in the field of computer graphics for the simulation and rendering of visual effects. Since the octree is generally deep with many levels, the traversal of trees and dereferencing pointers are costly. In contrast, the VDB data structure uses a shallow tree with 4 levels and large branching factor instead of the 8 branched limited by the octree. Every leaf node in VDB is a block consisting of $8 \times 8 \times 8$

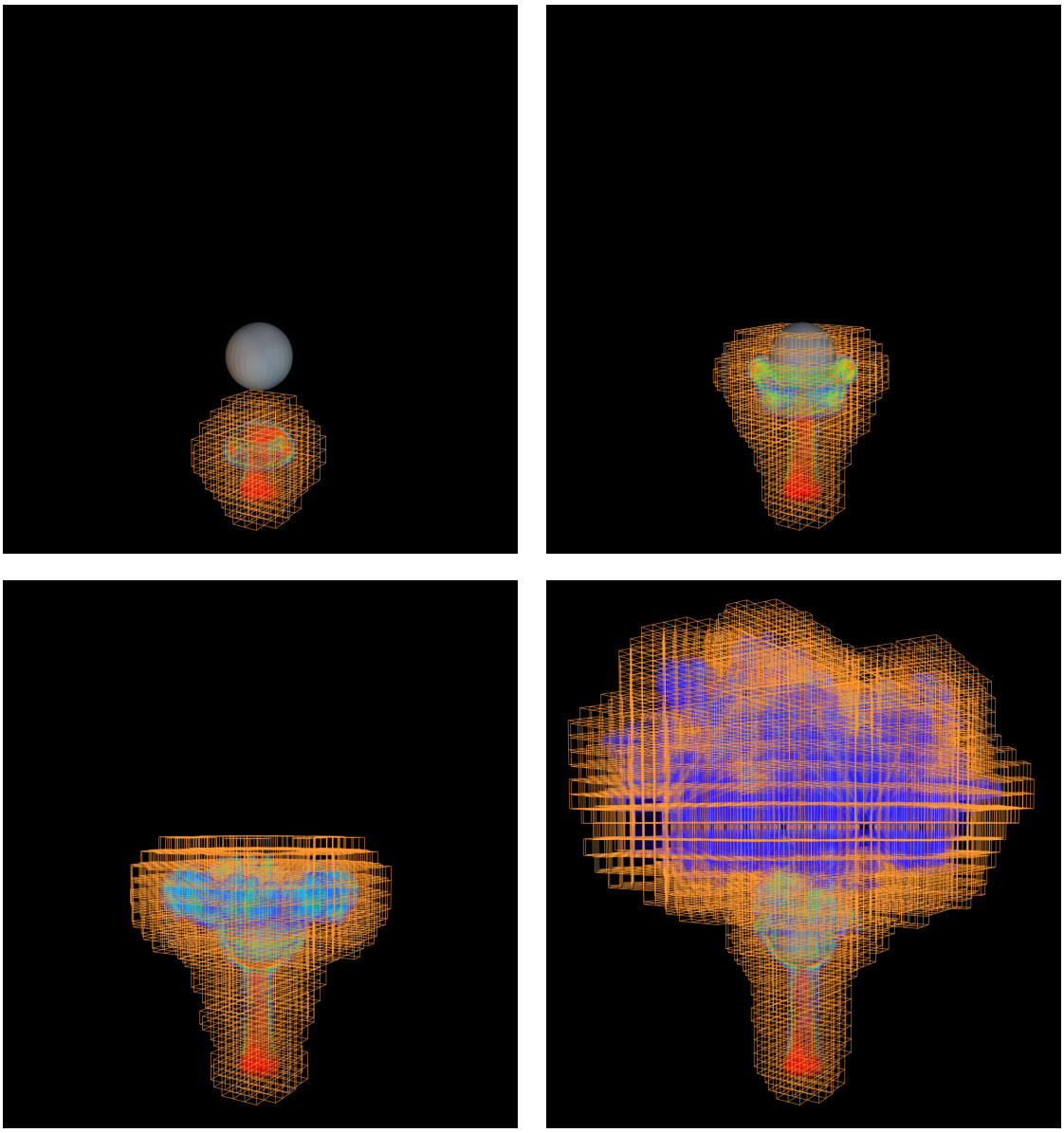


Figure 5.11: Blocks of sparse grid in smoke simulation.

cells which is named as voxel in VDB. Different with the octree based adaptive grid, all of the blocks have the same size. A SPGrid data structure is proposed to model an adaptive octree grid as a pyramid of sparse uniform grids to avoid the overhead of data access associated with octree structure [104]. Based on the VDB framework, SIMD-accelerated [105] and GPU-accelerated [106] fluid simulations have been presented.

In this thesis, a sparse grid based on spatial hash is implemented on GPU for fluid simulation. Each block is stored as an entry in the hash table. As illustrated in Fig. 5.10, the blocks of a sparse grid are only allocated in the region of interest. The data in a cell can be access with indices just like the dense uniform grid but actually realized by a

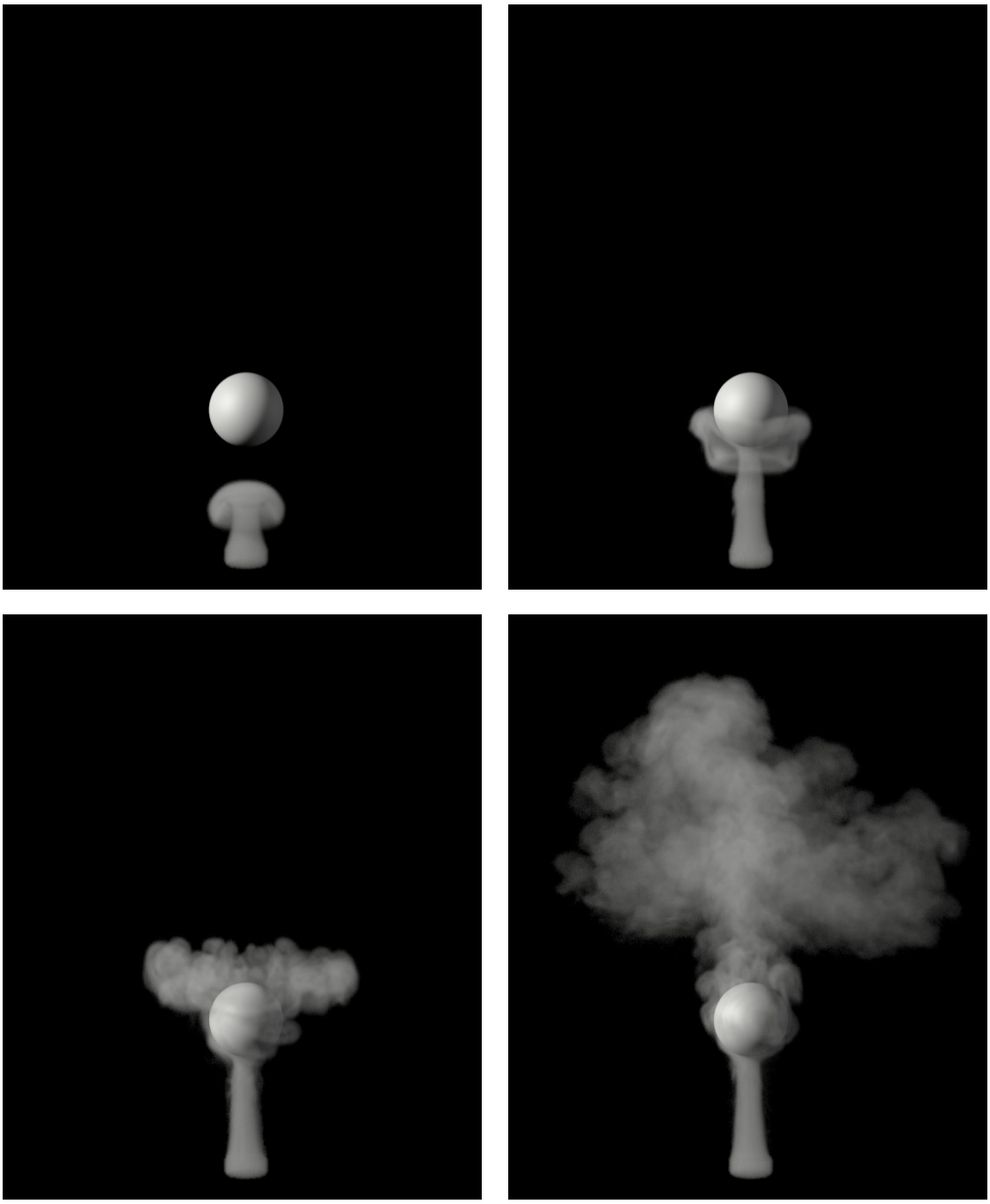


Figure 5.12: Smoke simulation with sparse grid.

hash query. There is no tree structure and spatially varying resolution, so that the stencil computation can be easily conducted without any interpolations. More details of GPU implementation and optimization for the hash table can be found in the references [107, 108]. The hash table is maintained dynamically by inserting and removing the blocks according to the topology change of the sparse grid.

A numerical example of smoke simulation by using the sparse grid on GPU is presented here. Besides solving the Navier-Stokes equations for fluid flow, the advection-diffusion equations are also solved to transport the smoke concentration s and temperature T:

$$\frac{Ds}{Dt} = k_s \nabla \cdot \nabla s, \tag{5.6a}$$

$$\frac{DT}{Dt} = k_T \nabla \cdot \nabla T, \tag{5.6b}$$

where k_s and k_T are non-negative diffusion constants. The buoyancy $\boldsymbol{b}$ related to the smoke is evaluated by the Boussinesq approximation [109] as

$$\boldsymbol{b} = \left[\alpha s - \beta \left(T - T_{\text{amb}}\right)\right] \boldsymbol{g}, \tag{5.7}$$

where α and β and constant coefficients, T_{amb} is the ambient temperature and $\boldsymbol{g}$ is the gravity acceleration.

A smoke flow past sphere is simulated where the emitter of smoke at the bottom is a cylinder with diameter 0.1 m and height 0.03 m. The sphere has a diameter of 0.2 m. The mesh resolution is 0.005 m. The dynamical topology of sparse grid in the simulation is depicted in Fig. 5.11. And the volume rendering for smoke is shown in Fig. 5.12. There are more than 10 thousand of blocks and 5 million cells in the last snapshot.

Chapter 6

Conclusions

6.1 Summary

This thesis presents a novel evolving pressure projection method to solve weakly compressible Navier-Stokes equations. This iteration of the pressure evolution and the projection for the velocity correction can effectively eliminate the acoustic wave affecting traditional simulations of a weakly compressible fluid and thus suppresses the pressure and velocity oscillations. The numerical stability is enhanced by employing a staggered grid system with finite-difference discretization. For a two-phase flow simulation with the one-fluid model, this solver is completed by using the conservative Allen-Cahn equation to capture the interface, which is solved with the finite volume method to exactly guarantee mass conservation. Moreover, the two-phase flow with a high density ratio is handled by adopting a density-weighted advection scheme which improves the consistency of the mass and momentum transport in the finite-difference method. In all benchmark tests for single and two-phase flows, including shear layers, decaying vortices, droplet, bubbles, and dam break, the capability of the proposed method to damp the acoustic wave and suppress oscillations is validated. As demonstrated by the simulation of 2D dam break, the proposed weakly compressible Navier-Stokes solver offers 10x speedup in comparison to an incompressible solver employing SOR method. According to the benchmark tests, using 20 iterations to solve the pressure evolution equation results in a negligible loss in accuracy.

On the basis of developing the evolving pressure projection method for solving weakly compressible Navier-Stokes equations, a conservative and consistent momentum transport method is proposed for simulating incompressible two-phase flows. To achieve consistent transport of mass and momentum, this method employs a collocated grid system on which the finite volume discretization is straightforward to simultaneously solve the VOF and momentum equation in conservative form. A simple consistent formulation can be

constructed by calculating the momentum flux with the help of the volume flux in the VOF equation. The new implementation of the evolving pressure projection method also effectively couples the pressure and velocity on the collocated grid, thereby enhancing the numerical stability of the present solver. The proposed method's ability to preserve momentum conservation at an arbitrary high density ratio is substantiated through various benchmark tests for two-phase flows, encompassing rising bubble, transport of a heavy droplet, oscillating drop, collapsing water column, and jet in a cross flow. The conservative and consistent method ensures accurate and physically realistic results by conserving momentum and accounting for the interactions between the phases. This is particularly important for simulating violent two-phase flows with high density ratios, where momentum transfer between the phases can have a significant impact on the behavior of the complex interfacial dynamics. While non-conservative inconsistent solver is not applicable to two-phase flow with density ratio higher than 1000, the proposed momentum-conserving solver remain stable at a density ratio of 10^6. In addition to its simplicity and efficiency in calculating pressure evolution equations, the momentum-conserving weakly compressible Navier-Stokes solver serves as a robust and high-fidelity tool for two-phase flow simulations.

Then, the multi-GPU computation for simulating two-phase flows is conducted with the original momentum-conserving weakly compressible Navier-Stokes solvers which are fully explicit and inherently parallel. Con-CAC-LS that involves conservative Allen-Cahn equation and level-set method and Con-PLIC-HF that involves PLIC and height function method are proposed under the principle of consistent transport and momentum conservation. The PLIC-VOF method exhibits an average numerical error reduction of 64.7% compared to the conservative Allen-Cahn model for two-phase interface capturing. For multi-GPU computation, the strong scaling with two types of domain partition is evaluated and a better scalability is exhibited by the 3D partition. The technique for overlapping communication and computation is implemented and discussed. The parallel efficiency of the proposed solver achieves 97.8% on 16 GPUs and 55.6% on 64 GPUs in the strong scaling of 1D partition, while it achieves 90.4% on 64 GPUs and 72% on 128 GPUs in the strong scaling of 3D partition method. The scaling test also shows that Con-PLIC-HF has a overall higher performance. After demonstrating the superior mesh convergence of Con-PLIC-HF through simulations of the Rayleigh-Taylor instability and the droplet oscillation, this solver is used for large-scale simulations of milk crown, dam break and liquid jet atomization problems. In these two-phase flow computations with high density ratio and high Reynolds number, high-fidelity results are presented with detailed reproduction of the complex interface structures. The simulations have shown the excellent applicability of the present solver.

Finally, the sparse data structure including adaptive mesh refinement and sparse

grid are introduced in this thesis. The GPU based algorithms are briefly explained. The numerical examples including two-phase flows and smoke simulations are presented. Both AMR and sparse grid can effectively reduce the amount of grid cells for storage and computation, hence minimize the need for computing resources. AMR is more suitable for interfacial flows because it only refines the mesh resolution near the interface and other region has a coarse mesh. Sparse grid allocates a uniform grid to the region of interest. It is free of any spatial interpolation and the traversal of complex tree structure. These features make the sparse grid a popular choice in the field of computer graphics.

In future research, a further extension of the momentum-conserving two-phase flow solver to large-scale multi-GPU simulation with adaptive mesh refinement [101, 110] and sparse grid can be expected. The solver can also be applied to study more complex multiphase flows such as phase change and viscoelastic fluids.

6.2 Originality

- This thesis proposes an accurate and efficient weakly compressible N-S stokes solver based on the evolving pressure projection method;
- The momentum-conserving solvers for simulating high density contrast two-phase flow is proposed. The choices of interface capturing and curvature estimation methods are also discussed;
- The present work evaluates the multi-GPU performance and scalability of the original solvers and demonstrated the high-fidelity large-scale simulation of gas-liquid two-phase flows.

The proposed solvers offer several advantages. First, weakly compressible Navier-Stokes solvers are easier to implement and potentially faster since they avoid the need to implicitly solve large systems of equations. Second, the evolving pressure projection method prevents pressure and velocity oscillations in the simulation of two-phase flows, improving the stability and accuracy of the solvers. Third, the present solver ensures consistent and conservative mass and momentum transport, which is crucial for simulating violent two-phase flows with high density ratios. Lastly, the methods are designed to scale on multi-GPU clusters, making them well-suited for large-scale simulations.

However, there are also several disadvantages to consider. The time step size is restricted by the speed of sound, requiring more computation steps to complete a simulation. Additionally, the methods will need further testing against a wider range of problems and conditions, such as non-isothermal flow and aerodynamics. The drawbacks of individual solvers, like Con-CAC-LS, have been discussed in the corresponding section.

Appendix A

Comparison of Interface Capturing Methods

This appendix compares the three interface capturing methods for two-phase flow used in this thesis, including the conservative Allen-Cahn model (CAC) introduced in Section 2.1.3, the THINC/WLIC method introduced in Section 3.2.1 and the PLIC method introduced in Section 4.1.2. These three methods belong to the categories of diffuse-interface method, algebraic VOF, and geometric VOF, respectively [19]. The famous Rider and Kothe benchmarks for the interface capturing method are calculated in this appendix [111, 112]. Same time step size is used by three method with CFL = 0.1 for a fair comparison.

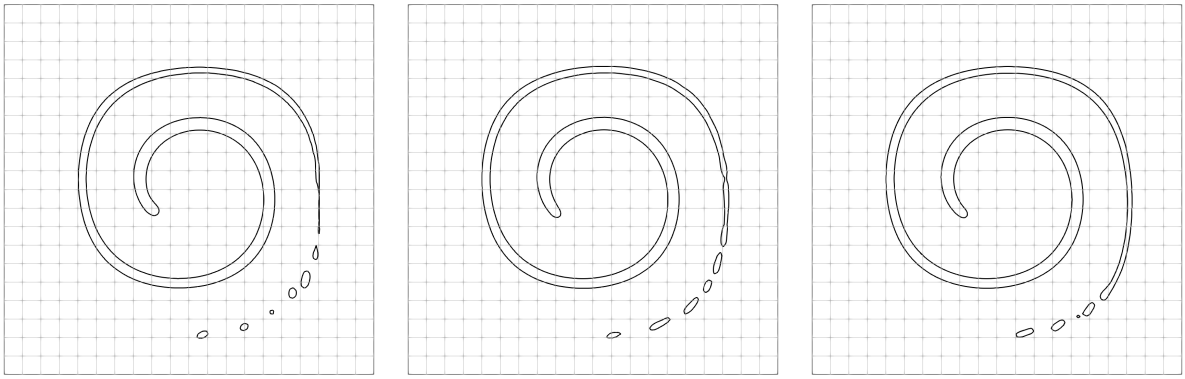
A.1 2D Single Vortex

The first test is a two-dimensional single vortex problem. A circle of radius 0.15 is centered at (0.5, 0.75) in a unit domain. The time-dependent velocity field is given by

$$u = 2\sin^2(\pi x)\sin(\pi y)\cos(\pi y)\cos\left(\frac{\pi t}{T}\right), \tag{A.1a}$$

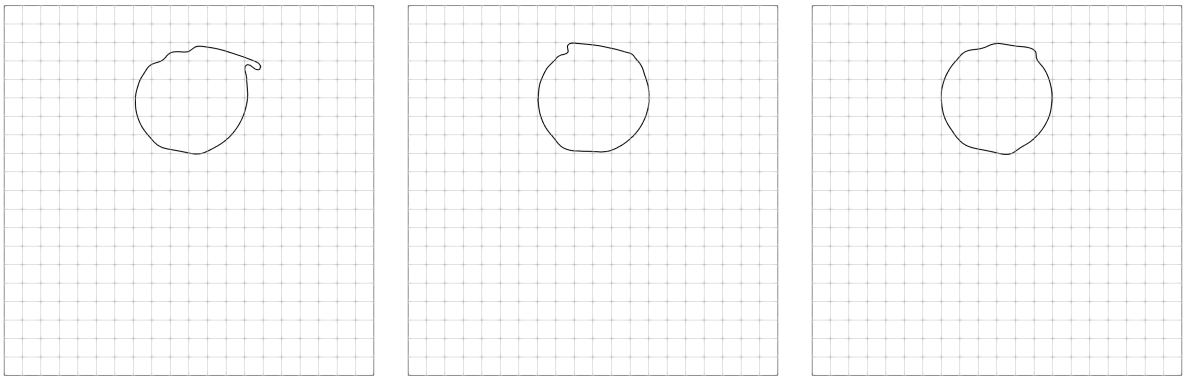
$$v = -2\sin(\pi x)\cos(\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right), \tag{A.1b}$$

where the period $T = 8$. The results of the interface evolution on 128×128 grid calculated by the CAC, THINC/WLIC and PLIC methods are depicted in Fig. A.1 and Fig. A.2. The results on 256×256 grid are depicted in Fig. A.3 and Fig. A.4. The circle is stretched to a thin and long filament at $t = 0.5T$, then it reverses to its initial location at $t = T$.



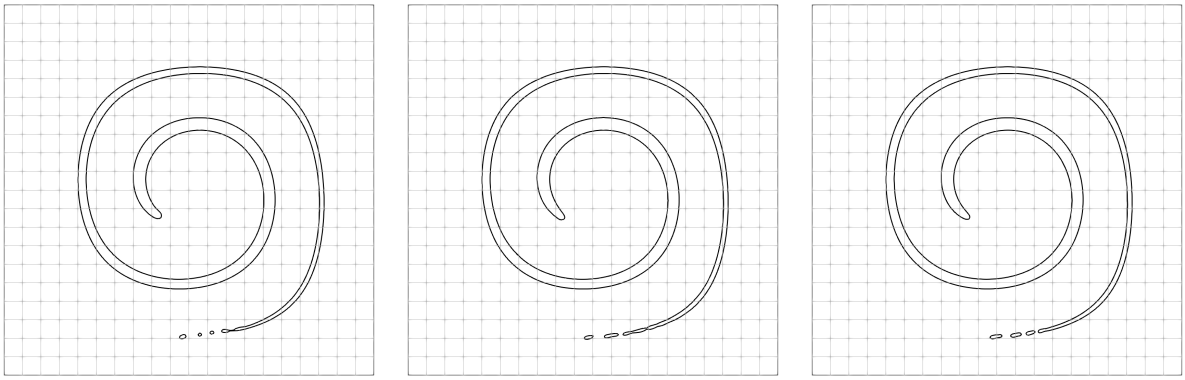
(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.1: Interface in 2D single vortex on 128×128 grid at $t = 0.5T$.



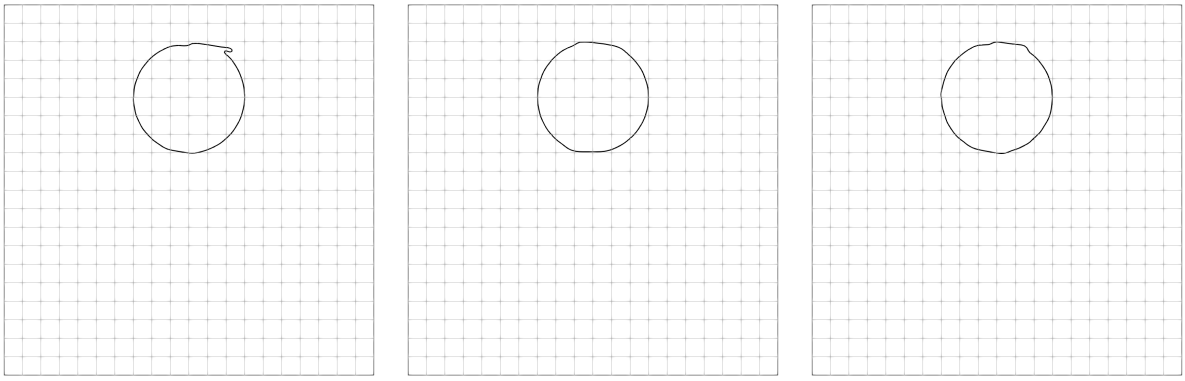
(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.2: Interface in 2D single vortex on 128×128 grid at $t = T$.



(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.3: Interface in 2D single vortex on 256×256 grid at $t = 0.5T$.



(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.4: Interface in 2D single vortex on 256×256 grid at $t = T$.

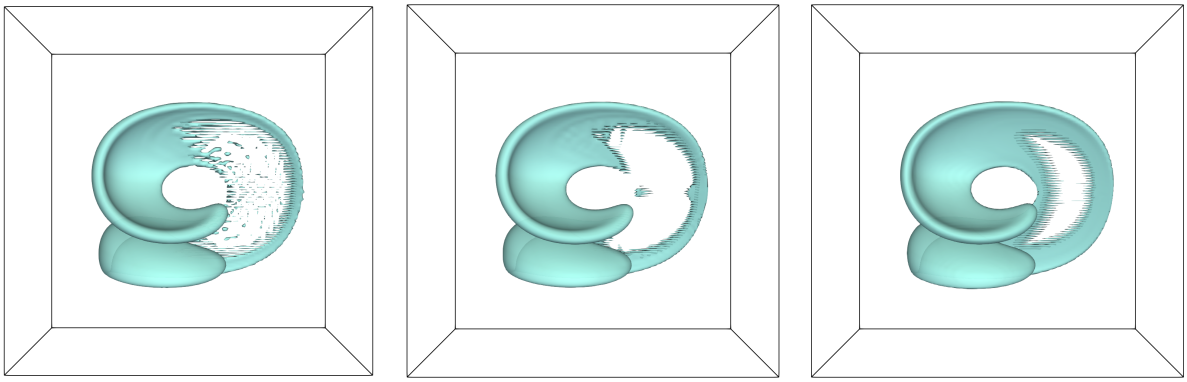
Table A.1: Numerical errors and convergence orders for the single vortex test.

Δx	CAC		THINC/WLIC		PLIC	
	L1 error	Order	L1 error	Order	L1 error	Order
1/64	5.49×10^{-1}	-	1.77×10^{-1}	-	2.55×10^{-1}	-
1/128	1.27×10^{-1}	2.11	5.26×10^{-2}	1.75	3.28×10^{-2}	2.96
1/256	2.74×10^{-2}	2.22	1.73×10^{-2}	1.61	8.58×10^{-3}	1.93
1/512	3.90×10^{-3}	2.81	8.33×10^{-3}	1.05	1.47×10^{-3}	2.54

To quantify the numerical errors of the interface capturing methods used in this thesis, the L1 error is defined as

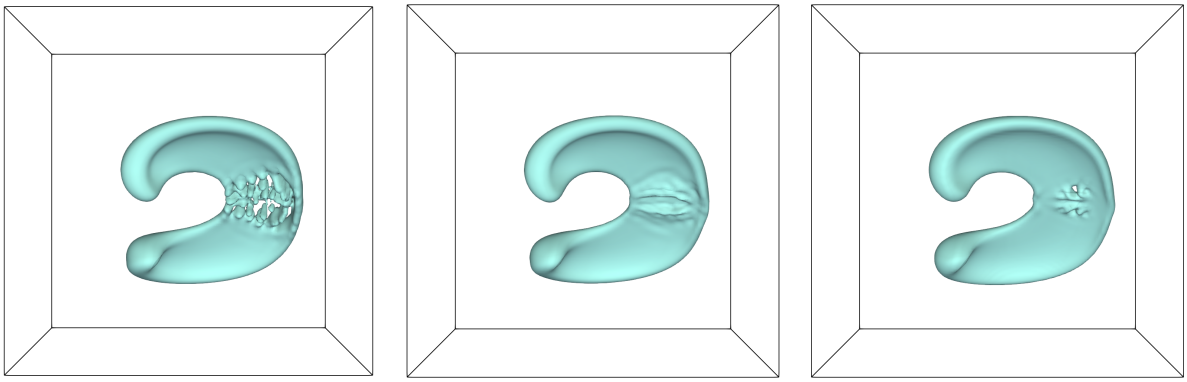
$$\text{L1 error} = \frac{\sum_{i,j} |\phi_{i,j} - \phi_{\text{exact}}|}{\sum_{i,j} |\phi_{\text{exact}}|}, \tag{A.2}$$

where the exact solution ϕ_{exact} is the phase-field and VOF functions at the beginning. And the final solution of the phase-field and VOF functions at $t = T$ is used to evaluate the accuracy. Table A.1 shows the L1 error and convergence order for the CAC, THINC/WLIC, and PLIC methods. Overall, the PLIC method exhibits higher accuracy and convergence order compared to the other methods.



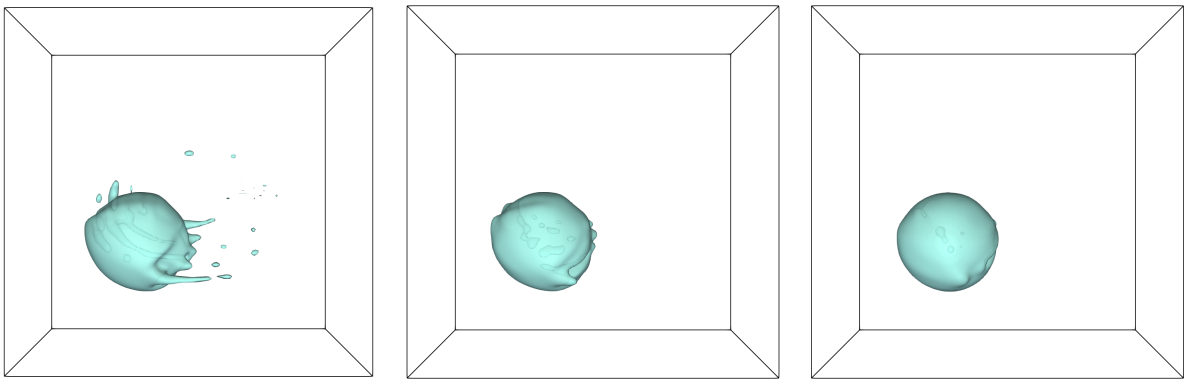
(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.5: Interface in 3D deformation field on $128 \times 128 \times 128$ grid at $t = 0.5T$.



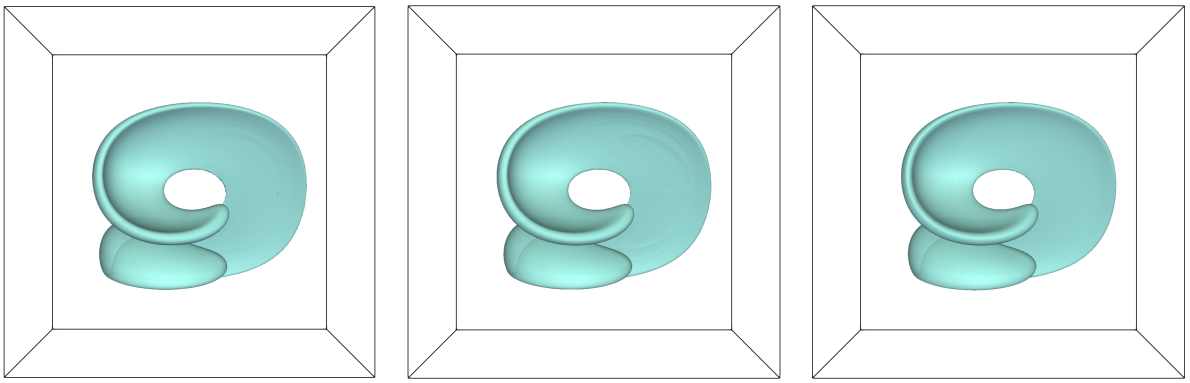
(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.6: Interface in 3D deformation field on $128 \times 128 \times 128$ grid at $t = 0.75T$.



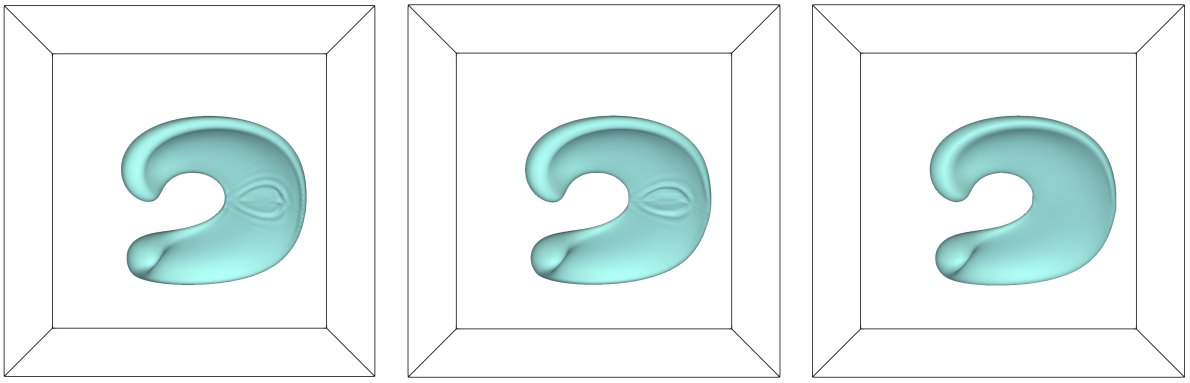
(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.7: Interface in 3D deformation field on $128 \times 128 \times 128$ grid at $t = T$.



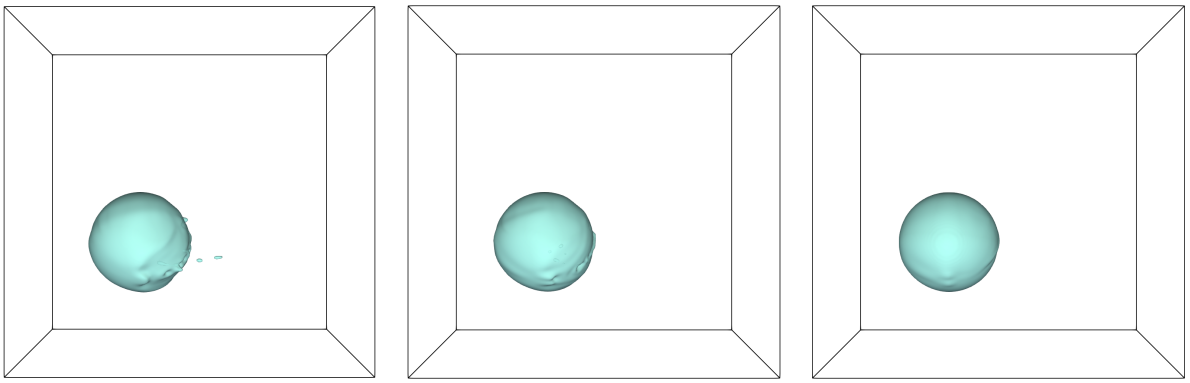
(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.8: Interface in 3D deformation field on $256 \times 256 \times 256$ grid at $t = 0.5T$.



(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.9: Interface in 3D deformation field on $256 \times 256 \times 256$ grid at $t = 0.75T$.



(a) CAC (b) THINC/WLIC (c) PLIC

Figure A.10: Interface in 3D deformation field on $256 \times 256 \times 256$ grid at $t = T$.

A.2 3D Deformation Field

The second test is a three-dimensional deformation problem. A sphere of radius 0.15 is centered at (0.35, 0.35, 0.35) in a unit domain. The 3D deformation field is given by

$$u = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right), \tag{A.3a}$$

$$v = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right), \tag{A.3b}$$

$$w = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos\left(\frac{\pi t}{T}\right), \tag{A.3c}$$

where the period $T = 3$. The results of the interface evolution on the 128×128 grid calculated by three methods are depicted in Fig. A.5, Fig. A.6 and Fig. A.7. The results on the 256×256 grid are depicted in Fig. A.8, Fig. A.9 and Fig. A.10. Both the 2D and 3D results demonstrate a better accuracy of the PLIC method for interface capturing, while the conservative Allen-Cahn model is less accurate with the same mesh resolution.

References

[1] John Kim and Parviz Moin. "Application of a fractional-step method to incompressible Navier-Stokes equations". In: *Journal of computational physics* 59.2 (1985), pp. 308–323.

[2] Sanghyun Ha, Junshin Park, and Donghyun You. "A GPU-accelerated semi-implicit fractional-step method for numerical solutions of incompressible Navier–Stokes equations". In: *Journal of Computational Physics* 352 (2018), pp. 246–264.

[3] Sanghyun Ha, Junshin Park, and Donghyun You. "A scalable multi-GPU method for semi-implicit fractional-step integration of incompressible Navier-Stokes equations". In: *arXiv preprint arXiv:1812.01178* (2018).

[4] Xiaolei Shi et al. "A parallel nonlinear multigrid solver for unsteady incompressible flow simulation on multi-GPU cluster". In: *Journal of Computational Physics* 414 (2020), p. 109447.

[5] Alexandre Joel Chorin. "Numerical solution of the Navier-Stokes equations". In: *Mathematics of computation* 22.104 (1968), pp. 745–762.

[6] Jonathan R Clausen. "Entropically damped form of artificial compressibility for explicit simulation of incompressible flow". In: *Physical Review E* 87.1 (2013), p. 013309.

[7] Adrien Toutant. "General and exact pressure evolution equation". In: *Physics Letters A* 381.44 (2017), pp. 3739–3742.

[8] Adrien Toutant. "Numerical simulations of unsteady viscous incompressible flows using general pressure equation". In: *Journal of Computational Physics* 374 (2018), pp. 822–842.

[9] Stuart E Rogers and Dochan Kwak. "Upwind differencing scheme for the time-accurate incompressible Navier-Stokes equations". In: *AIAA journal* 28.2 (1990), pp. 253–262.

[10] Xiaolei Shi and Chao-An Lin. "Simulations of Wall Bounded Turbulent Flows Using General Pressure Equation". In: *Flow, Turbulence and Combustion* (2020), pp. 1–16.

[11] Yann T Delorme et al. "A simple and efficient incompressible Navier-Stokes solver for unsteady complex geometry flows on truncated domains". In: *Computers & Fluids* 150 (2017), pp. 84–94.

[12] Alejandro Figueroa and Rainald Löhner. "Postprocessing-based interpolation schemes for nested Cartesian finite difference grids of different size". In: *International Journal for Numerical Methods in Fluids* 89.6 (2019), pp. 196–215.

[13] Adam Kajzer and Jacek Pozorski. "Application of the Entropically Damped Artificial Compressibility model to direct numerical simulation of turbulent channel flow". In: *Computers & Mathematics with Applications* 76.5 (2018), pp. 997–1013.

[14] Shintaro Matsushita and Takayuki Aoki. "A weakly compressible scheme with a diffuse-interface method for low Mach number two-phase flows". In: *Journal of Computational Physics* 376 (2019), pp. 838–862.

[15] Stanley Osher and James A Sethian. "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations". In: *Journal of computational physics* 79.1 (1988), pp. 12–49.

[16] Cyril W Hirt and Billy D Nichols. "Volume of fluid (VOF) method for the dynamics of free boundaries". In: *Journal of computational physics* 39.1 (1981), pp. 201–225.

[17] Daniel M Anderson, Geoffrey B McFadden, and Adam A Wheeler. "Diffuse-interface methods in fluid mechanics". In: *Annual review of fluid mechanics* 30.1 (1998), pp. 139–165.

[18] Mark Sussman and Elbridge Gerry Puckett. "A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows". In: *Journal of computational physics* 162.2 (2000), pp. 301–337.

[19] Shahab Mirjalili, Suhas S Jain, and Micheal Dodd. "Interface-capturing methods for two-phase flows: An overview and recent developments". In: *Center for Turbulence Research Annual Research Briefs* (2017), pp. 117–135.

[20] Stéphane Zaleski and Feng Xiao. "Special issue: Numerical methods and modeling of multiphase flows". In: *Journal of Computational Physics* 402 (2020), p. 108902.

[21] Peter Zaspel and Michael Griebel. "Solving incompressible two-phase flows on multi-GPU clusters". In: *Computers & Fluids* 80 (2013), pp. 356–364.

[22] Yu Wang, Chang Shu, and LM Yang. "An improved multiphase lattice Boltzmann flux solver for three-dimensional flows with large density ratio and high Reynolds number". In: *Journal of computational physics* 302 (2015), pp. 41–58.

[23] Takaji Inamuro et al. "An improved lattice Boltzmann method for incompressible two-phase flows with large density differences". In: *Computers & Fluids* 137 (2016), pp. 55–69.

[24] Abbas Fakhari and Diogo Bolster. "Diffuse interface modeling of three-phase contact line dynamics on curved boundaries: A lattice Boltzmann model for large density and viscosity ratios". In: *Journal of Computational Physics* 334 (2017), pp. 620–638.

[25] Chunhua Zhang, Zhaoli Guo, and Yibao Li. "A fractional step lattice Boltzmann model for two-phase flow with large density differences". In: *International Journal of Heat and Mass Transfer* 138 (2019), pp. 1128–1141.

[26] Yos Panagaman Sitompul and Takayuki Aoki. "A filtered cumulant lattice Boltzmann method for violent two-phase flows". In: *Journal of Computational Physics* 390 (2019), pp. 93–120.

[27] Taku Ohwada, Pietro Asinari, and Daisuke Yabusaki. "Artificial compressibility method and lattice Boltzmann method: Similarities and differences". In: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3461–3474.

[28] Adam Kajzer and Jacek Pozorski. "A weakly Compressible, Diffuse-Interface Model for Two-Phase Flows". In: *Flow, Turbulence and Combustion* (2020), pp. 299–333.

[29] Murray Rudman. "A volume-tracking method for incompressible multifluid flows with large density variations". In: *International Journal for numerical methods in fluids* 28.2 (1998), pp. 357–378.

[30] Davide Zuzio et al. "A new efficient momentum preserving Level-Set/VOF method for high density and momentum ratio incompressible two-phase flows". In: *Journal of Computational Physics* 410 (2020), p. 109342.

[31] T Arrufat et al. "A mass-momentum consistent, Volume-of-Fluid method for incompressible flow on staggered grids". In: *Computers & Fluids* 215 (2021), p. 104785.

[32] Nishant Nangia et al. "A robust incompressible Navier-Stokes solver for high density ratio multiphase flows". In: *Journal of Computational Physics* 390 (2019), pp. 548–594.

[33] Shahab Mirjalili and Ali Mani. "Consistent, energy-conserving momentum transport for simulations of two-phase flows using the phase field equations". In: *Journal of Computational Physics* 426 (2021), p. 109918.

[34] Jitendra Kumar Patel and Ganesh Natarajan. "A novel consistent and well-balanced algorithm for simulations of multiphase flows on unstructured grids". In: *Journal of computational physics* 350 (2017), pp. 207–236.

[35] Markus Bussmann, Douglas B Kothe, and James M Sicilian. "Modeling high density ratio incompressible interfacial flows". In: *Fluids Engineering Division Summer Meeting*. Vol. 36150. 2002, pp. 707–713.

[36] Stéphane Popinet. "An accurate adaptive solver for surface-tension-driven interfacial flows". In: *Journal of Computational Physics* 228.16 (2009), pp. 5838–5866.

[37] Néstor Balcázar et al. "A finite-volume/level-set method for simulating two-phase flows on unstructured grids". In: *International journal of multiphase flow* 64 (2014), pp. 55–72.

[38] Daniel Fuster and Stéphane Popinet. "An all-Mach method for the simulation of bubble dynamics problems in the presence of surface tension". In: *Journal of Computational Physics* 374 (2018), pp. 752–768.

[39] Adam O'Brien and Markus Bussmann. "A moving immersed boundary method for simulating particle interactions at fluid-fluid interfaces". In: *Journal of Computational Physics* 402 (2020), p. 109089.

[40] William J Rider. "Filtering non-solenoidal modes in numerical solutions of incompressible flows". In: *International journal for numerical methods in fluids* 28.5 (1998), pp. 789–814.

[41] Andrea Aprovitola and Filippo M Denaro. "A non-diffusive, divergence-free, finite volume-based double projection method on non-staggered grids". In: *International journal for numerical methods in fluids* 53.7 (2007), pp. 1127–1172.

[42] R Abbasi, A Ashrafizadeh, and A Shadaram. "A comparative study of finite volume pressure-correction projection methods on co-located grid arrangements". In: *Computers & Fluids* 81 (2013), pp. 68–84.

[43] Mohamed El Ouafa, Stephane Vincent, and Vincent Le Chenadec. "Monolithic solvers for incompressible two-phase flows at large density and viscosity ratios". In: *Fluids* 6.1 (2021), p. 23.

[44] Kai Yang and Takayuki Aoki. "Multi-GPU Scaling of a Conservative Weakly Compressible Solver for Large-scale Two-phase Flow Simulation". In: *PDCAT 2022, Lecture Notes in Computer Science*. Vol. 13798. Springer Nature Switzerland, 2023, pp. 16–27.

[45] Shintaro Matsushita and Takayuki Aoki. "Gas-liquid two-phase flows simulation based on weakly compressible scheme with interface-adapted AMR method". In: *Journal of Computational Physics* 445 (2021), p. 110605.

[46] Tongda Lian, Shintaro Matsushita, and Takayuki Aoki. "An AMR-Based Liquid Film Simulation with Surfactant Transport Using PLIC-HF Method". In: *Applied Sciences* 13.3 (2023), p. 1955.

[47] Xian Wang and Takayuki Aoki. "Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster". In: *Parallel Computing* 37.9 (2011), pp. 521–535.

[48] T Hashimoto et al. "Multi-GPU parallel computation of unsteady incompressible flows using kinetically reduced local Navier–Stokes equations". In: *Computers & Fluids* 167 (2018), pp. 215–220.

[49] CW Hirt and BD Nichols. "Adding limited compressibility to incompressible hydrocodes". In: *Journal of Computational Physics* 34.3 (1980), pp. 390–400.

[50] Alexandre Joel Chorin. "A numerical method for solving incompressible viscous flow problems". In: *Journal of computational physics* 2.1 (1967), pp. 12–26.

[51] Grétar Tryggvason et al. "A front-tracking method for the computations of multiphase flow". In: *Journal of computational physics* 169.2 (2001), pp. 708–759.

[52] Pao-Hsiung Chiu and Yan-Ting Lin. "A conservative phase field method for solving incompressible two-phase flows". In: *Journal of Computational Physics* 230.1 (2011), pp. 185–204.

[53] John W Cahn and John E Hilliard. "Free energy of a nonuniform system. I. Interfacial free energy". In: *The Journal of chemical physics* 28.2 (1958), pp. 258–267.

[54] Martin Geier, Abbas Fakhari, and Taehun Lee. "Conservative phase-field lattice Boltzmann model for interface tracking equation". In: *Physical Review E* 91.6 (2015), p. 063309.

[55] Elin Olsson and Gunilla Kreiss. "A conservative level set method for two phase flow". In: *Journal of computational physics* 210.1 (2005), pp. 225–246.

[56] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer Verlag, 2003.

[57] Kensuke 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". In: *Journal of Computational Physics* 232.1 (2013), pp. 252–271.

[58] Kensuke Yokoi. "A density-scaled continuum surface force model within a balanced force formulation". In: *Journal of Computational Physics* 278 (2014), pp. 221–228.

[59] Guang-Shan Jiang and Danping Peng. "Weighted ENO schemes for Hamilton–Jacobi equations". In: *SIAM Journal on Scientific computing* 21.6 (2000), pp. 2126–2143.

[60] Stéphane Popinet. "Numerical models of surface tension". In: *Annual Review of Fluid Mechanics* 50 (2018), pp. 49–75.

[61] Bin Xie, Peng Jin, and Feng Xiao. "An unstructured-grid numerical model for interfacial multiphase fluids based on multi-moment finite volume formulation and THINC method". In: *International Journal of Multiphase Flow* 89 (2017), pp. 375–398.

[62] Elias Konstantinidis and Yiannis Cotronis. "Graphics processing unit acceleration of the red/black SOR method". In: *Concurrency and Computation: Practice and Experience* 25.8 (2013), pp. 1107–1120.

[63] Bram Van Leer. "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method". In: *Journal of computational Physics* 32.1 (1979), pp. 101–136.

[64] Danping Peng et al. "A PDE-based fast local level set method". In: *Journal of computational physics* 155.2 (1999), pp. 410–438.

[65] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. "Strong stability-preserving high-order time discretization methods". In: *SIAM review* 43.1 (2001), pp. 89–112.

[66] Dalton JE Harvie, MR Davidson, and Murray Rudman. "An analysis of parasitic current generation in volume of fluid simulations". In: *Applied mathematical modelling* 30.10 (2006), pp. 1056–1066.

[67] S Aland and A Voigt. "Benchmark computations of diffuse interface models for two-dimensional bubble dynamics". In: *International Journal for Numerical Methods in Fluids* 69.3 (2012), pp. 747–761.

[68] Shu-Ren Hysing et al. "Quantitative benchmark computations of two-dimensional bubble dynamics". In: *International Journal for Numerical Methods in Fluids* 60.11 (2009), pp. 1259–1288.

[69] Changhong Hu and Makoto Sueyoshi. "Numerical simulation and experiment on dam break problem". In: *Journal of Marine Science and Application* 9.2 (2010), pp. 109–114.

[70] John Christopher Martin et al. "Part IV. An experimental study of the collapse of liquid columns on a rigid horizontal plane". In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 244.882 (1952), pp. 312–324.

[71] G Brereton and D Korotney. "Coaxial and oblique coalescence of two rising bubbles". In: *Dynamics of bubbles and vortices near a free surface* 119 (1991), pp. 50–73.

[72] M van Sint Annaland, NG Deen, and JAM Kuipers. "Numerical simulation of gas bubbles behaviour using a three-dimensional volume of fluid method". In: *Chemical Engineering Science* 60.11 (2005), pp. 2999–3011.

[73] PK Stansby, A Chegini, and TCD Barnes. "The initial stages of dam-break flow". In: *Journal of Fluid Mechanics* 374 (1998), pp. 407–424.

[74] Imre M Jánosi et al. "Turbulent drag reduction in dam-break flows". In: *Experiments in Fluids* 37.2 (2004), pp. 219–229.

[75] ZH Gu et al. "Interface-preserving level set method for simulating dam-break flows". In: *Journal of computational physics* 374 (2018), pp. 249–280.

[76] F Xiao, Y Honma, and T Kono. "A simple algebraic interface capturing scheme using hyperbolic tangent function". In: *International Journal for Numerical Methods in Fluids* 48.9 (2005), pp. 1023–1040.

[77] Kensuke Yokoi. "Efficient implementation of THINC scheme: a simple and practical smoothed VOF algorithm". In: *Journal of Computational Physics* 226.2 (2007), pp. 1985–2002.

[78] Satoshi Ii, Bin Xie, and Feng Xiao. "An interface capturing method with a continuous function: The THINC method on unstructured triangular and tetrahedral meshes". In: *Journal of Computational Physics* 259 (2014), pp. 260–269.

[79] Stéphane Popinet. "Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries". In: *Journal of Computational Physics* 190.2 (2003), pp. 572–600.

[80] Libor Lobovský et al. "Experimental investigation of dynamic pressure loads during dam break". In: *Journal of Fluids and Structures* 48 (2014), pp. 407–434.

[81] Subrahmanyan Chandrasekhar. *Hydrodynamic and hydromagnetic stability*. Oxford, 1961.

[82] Massoud Rezavand, Chi Zhang, and Xiangyu Hu. "A weakly compressible SPH method for violent multi-phase flows with high density ratio". In: *Journal of Computational Physics* 402 (2020), p. 109092.

[83] Dahya Bhaga and ME Weber. "Bubbles in viscous liquids: shapes, wakes and velocities". In: *Journal of fluid Mechanics* 105 (1981), pp. 61–85.

[84] Jinsong Hua and Jing Lou. "Numerical simulation of bubble rising in viscous liquid". In: *Journal of Computational Physics* 222.2 (2007), pp. 769–795.

[85] KA Sallam, Christian Aalburg, and GM Faeth. "Breakup of round nonturbulent liquid jets in gaseous crossflow". In: *AIAA journal* 42.12 (2004), pp. 2529–2540.

[86] Pei-Kuan Wu et al. "Breakup processes of liquid jets in subsonic crossflows". In: *Journal of Propulsion and power* 13.1 (1997), pp. 64–73.

[87] Xiaoyi Li and Marios C Soteriou. "High fidelity simulation and analysis of liquid jet atomization in a gaseous crossflow at intermediate Weber numbers". In: *Physics of Fluids* 28.8 (2016), p. 082101.

[88] Eugenio Aulisa et al. "Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry". In: *Journal of Computational Physics* 225.2 (2007), pp. 2301–2319.

[89] Ruben Scardovelli and Stephane Zaleski. "Analytical relations connecting linear interfaces and volume fractions in rectangular grids". In: *Journal of Computational Physics* 164.1 (2000), pp. 228–237.

[90] Gabriel D Weymouth and Dick K-P Yue. "Conservative volume-of-fluid method for free-surface simulations on cartesian-grids". In: *Journal of Computational Physics* 229.8 (2010), pp. 2853–2865.

[91] John B Bell, Phillip Colella, and Harland M Glaz. "A second-order projection method for the incompressible Navier-Stokes equations". In: *Journal of computational physics* 85.2 (1989), pp. 257–283.

[92] Sharen J Cummins, Marianne M Francois, and Douglas B Kothe. "Estimating curvature from volume fractions". In: *Computers & structures* 83.6-7 (2005), pp. 425–434.

[93] Joaquin Lopez et al. "An improved height function technique for computing interface curvature from volume fractions". In: *Computer methods in applied mechanics and engineering* 198.33-36 (2009), pp. 2555–2564.

[94] Nvidia Corporation. *CUDA C best practices guide*. URL: https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html.

[95] Shahab Mirjalili, Christopher B Ivey, and Ali Mani. "Comparison between the diffuse interface and volume of fluid methods for simulating two-phase flows". In: *International Journal of Multiphase Flow* 116 (2019), pp. 221–238.

[96] DJ Torres and JU Brackbill. "The point-set method: front-tracking without connectivity". In: *Journal of Computational Physics* 165.2 (2000), pp. 620–644.

[97] Bin Xie et al. "A conservative solver for surface-tension-driven multiphase flows on collocated unstructured grids". In: *Journal of Computational Physics* 401 (2020), p. 109025.

[98] Kai Yang and Takayuki Aoki. "Weakly compressible Navier-Stokes solver based on evolving pressure projection method for two-phase flow simulations". In: *Journal of Computational Physics* 431 (2021), p. 110113.

[99] Kai Yang and Takayuki Aoki. "A Momentum-Conserving Weakly Compressible Navier–Stokes Solver for Simulation of Violent Two-Phase Flows with High Density Ratio". In: *International Journal of Computational Fluid Dynamics* 36.9 (2022), pp. 776–796.

[100] Arthur R Butz. "Space filling curves and mathematical programming". In: *Information and Control* 12.4 (1968), pp. 314–330.

[101] Mohamed Wahib, Naoya Maruyama, and Takayuki Aoki. "Daino: a high-level framework for parallel and efficient AMR on GPUs". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press. 2016, p. 53.

[102] Florian Schornbaum and Ulrich Rüde. "Massively parallel algorithms for the lattice Boltzmann method on nonuniform grids". In: *SIAM Journal on Scientific Computing* 38.2 (2016), pp. C96–C126.

[103] Ken Museth. "VDB: High-Resolution Sparse Volumes with Dynamic Topology". In: *ACM Trans. Graph.* 32.3 (July 2013).

[104] Rajsekhar Setaluri et al. "SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation". In: *ACM Trans. Graph.* 33.6 (Nov. 2014).

[105] Han Shao, Libo Huang, and Dominik L. Michels. "A Fast Unsmoothed Aggregation Algebraic Multigrid Framework for the Large-Scale Simulation of Incompressible Flow". In: *ACM Trans. Graph.* 41.4 (July 2022).

[106] Kui Wu et al. "Fast fluid simulations with sparse volumes on the GPU". In: *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 157–167.

[107] Muhammad A Awad et al. "Better GPU Hash Tables". In: *arXiv preprint arXiv:2108.07232* (2021).

[108] Saman Ashkiani, Martin Farach-Colton, and John D Owens. "A dynamic hash table for the GPU". In: *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2018, pp. 419–429.

[109] Robert Bridson. *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015.

[110] Carsten Burstedde, Lucas C Wilcox, and Omar Ghattas. "p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees". In: *SIAM Journal on Scientific Computing* 33.3 (2011), pp. 1103–1133.

[111] William J Rider and Douglas B Kothe. "Reconstructing volume tracking". In: *Journal of computational physics* 141.2 (1998), pp. 112–152.

[112] Douglas Enright et al. "A hybrid particle level set method for improved interface capturing". In: *Journal of Computational physics* 183.1 (2002), pp. 83–116.