USC-SIPI REPORT #404

Advanced Liquid Simulation Techniques for Computer Graphics Applications

by

Youngmin Kwak

August 2010

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Viterbi School of Engineering
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Suite 400
Los Angeles, CA 90089-2564 U.S.A.

ADVANCED LIQUID SIMULATION TECHNIQUES FOR COMPUTER GRAPHICS APPLICATIONS

by

Youngmin Kwak

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2010

Dedication

This dissertation is dedicated to my wife, Ally, and daughter, Leah. I appreciate their continuous devotion and prayer. I love you with all my heart.

Table of Contents

List Of Tables

List Of Figures

Abstract

The particle level set method (PLSM) and the lattice Boltzmann method (LBM) have been two major physics-based liquid simulation techniques used in computer graphics to generate splendid and dynamic visual effects. The PLSM suffers from a high computational cost which arises from the global pressure correction step whereas the LBM requires a large memory size to store distribution functions.

In this research, we propose a hybrid lattice Boltzmann method (HLBM), which integrates the PLSM and the LBM, to visualize realistic liquid motion with emphasis on the behavior of the liquid-gas interface. The HLBM first runs the LBM solver, computes macroscopic velocities, and extrapolates the velocity field to the gas region. Subsequently, the level set function and particles are advected by the extrapolated velocity field, and advected particles are used to correct errors in the level set function based on the PLSM. Finally, the density difference between the LBM and the PLSM solvers is added to distribution functions to correct errors of the LBM. Simulation results show that the HLBM improves the quality of the fluid simulation without increasing the number of grids. As compared with the simulation using the LBM with 50^3 grids, the mean of the geometrical distance from the ground truth has been decreased by 21.70% and 13.02% for the water drop and the broken dam simulations, respectively, using the HLBM with the same

number of grids. The simulation results also show that the HLBM offers more splashy and dynamic visual effects than the LBM without increasing the grid size.

Also we propose a multicomponent-multiphase hybrid lattice Boltzmann method (MCMP-HLBM) which integrates the PLSM and the MCMP-LBM, to visualize realistic bubble motion with emphasis on the behavior of the liquid-bubble interface. The MCMP-HLBM first runs the MCMP-LBM solver and computes composite macroscopic velocities. Then, the level set function and particles are advected by the the composite velocity field, and advected particles are used to correct errors in the level set function based on the PLSM. Finally, the density difference between the MCMP-LBM and the PLSM solvers is added to the distribution functions to correct the errors of the MCMP-LBM. We test the method for the bubble coalescence and rising simulations. The results show that the MCMP-HLBM improves the quality of the fluid simulation without increasing the number of grids. Compared with the simulation using the MCMP-LBM, the normalized absolute difference from the ground truth is 61.50% and 36.50% less using the MCMP-HLBM for two dimensional two- and three-bubble coalescence simulations, respectively, using the MCMP-HLBM with the same number of grids. Also the normalized absolute difference from the ground truth using the MCMP-LBM has been decreased by 44.93%, 56.02%, and 40.62% for two dimensional single-, two-, and three-bubble rising simulations, respectively, using the MCMP-HLBM with the same number of grids. For the case of three dimensional single-, two-, and three-bubble rising simulations using the MCMP-HLBM, the mean of the geometrical distance from the ground truth has been decreased by 11.75%, 38.95%, and 26.57% as compared with the simulation using the

MCMP-LBM, respectively. The simulation results also show that the MCMP-HLBM offers more detailed visual results than the MCMP-LBM without increasing the grid size.

Chapter 1

Introduction

1.1 Significance of the Research

Computer games and compute-aided special effects in movies have become the mainstream of the entertainment industry. Realistic fluid simulation plays an important role in many of these scenes. Examples of fluid simulation include: smoke, cloud, liquid, bubble, fire and so on. In this research, we propose several graphic techniques to enhance the visual performance of liquid simulation.

Accurate liquid simulation will result in a splendid and dynamic visual effects. It moves very fast and has a surface that is different from other fluids such as smoke and fire. The physics-based approach has been commonly used to simulate liquids since it is more accurate compared that the non-physics-based approach based on texture or noise synthesis [57, 62]. It approximates the law of physics by numerical algorithms, and creates realistic and plausible motion of animated liquids automatically. Besides, it is easy to incorporate some control mechanism such as user interaction in the physics-based approach.

There are two main fluid simulation methods in the physics-based approach: 1) the particle level set method (PLSM) and 2) the lattice Boltzmann method (LBM). The PLSM is efficient in smooth surface representation and numerically stable. However, it demands a global pressure correction step that involves the solution of the Poisson equation [18]. Thus, the computational complexity is higher. Besides, it is difficult to simulate the splashing effect with this method. The LBM is an efficient mass-conserving algorithm. However, it demands a large amount of memory to store distribution functions at each lattice. It has a very tight restriction on the time step to guarantee numerical stability of the computation. As a result, it is difficult to simulate a smooth liquid surface.

Each of the two methods discussed above has its own strength and weakness. It is desirable to improve and integrate them for more realistic liquid simulation. In this research, we first propose a hybrid LBM for free surface fluid simulation that integrates the PLSM and the LBM. Then, we present a hybird LBM for multicomponent-multiphase fluids.

1.2 Review of Previous Work

The history of fluid simulation goes back to 1700 when Newton formulated a set of basic equations to describe fluid flow mathematically. After that, quite a few scientists such as Euler, Navier and Stokes worked on this problem. The Navier-Stokes and the Boltzmann equations were derived separately to describe the law of fluid flows. The Navier-Stokes equations provide a macroscopic fluid movement description while the Boltzmann equation offers a microscopic fluid motion characterization.

The level set method [47] was used to solve the Navier-Stokes equations numerically for fluid simulation. Later on, it was improved by adding particles to correct errors, and the resultant method is called the PLSM. More recently, further improvements of the level set method have been studied by many researchers. They include: the boundary handling [19], integration with the multi-grid method using the octree data structure [41], and simulation of multi-phase fluids [42]. The PLSM is one of the popular fluid simulation methods because of its realistic representation of liquid. However, it suffers from a high computational complexity of solving the Poisson equation which is needed in the global pressure correction step.

LBM was originated from the lattice gas cellular automata [51]. It provides a first-order explicit discretization of the Boltzmann equation in a discrete phase-space. The simulation region of the LBM is divided into a Cartesian grid of cells, each of which only interacts with cells of its direct neighborhood while the PLSM demands interaction of all cells in the global pressure correction step. Thus, the LBM is simpler and faster than the PLSM. Furthermore, adaptive time steps and multi-grid methods were proposed on top of the basic LBM in recent years [71, 83]. However, the LBM demands a larger memory space to deal with distribution functions. Besides, it does not have a smooth surface representation.

Multicomponent-multiphase LBM (MCMP-LBM) was introduced by Shan and Chen [56] by introducing non-local interactions between particles. Swift et al. [66,67] developed the MCMP-LBM using the free energy approach. For the simulation of bubble with high density ratios, Inamuro et al. [32] used the projection method together with free-energy model to deal with immiscible fluids with large density ratios. Inamuro et al. [31]

conducted simulations for bubbly flows with large density ratios using the projection method. More recently, Gupta and Kumar [20] presented multiple bubble dynamics with respect to Eotvos number, Reynolds number, and Morton number.

1.3 Contributions of the Research

For liquid animation, we first run a fluid solver, *e.g.,* the particle level set solver or the lattice Boltzmann solver, to find the surface information. Then, we triangulate the surface information using a marching cube algorithm. Finally, we render the the triangulated mesh with a ray tracer. The main contributions of this research lie in the improvement of the solver by integrating the PLSM and the LBM solvers. It will be addressed in detail in Chapters 4 and 5.

The LBM discretizes and solves the Boltzmann equation with very fast and efficient algorithm with streaming an collision steps. But it requires large amount of memory to store float point distribution functions. So the LBM is not suitable for high resolution fluid simulation. To get a fast and realistic free surface fluid simulation, we improve the LBM by incorporating the PLSM.

- A new hybrid LBM is proposed to improve the liquid-gas surface simulation.

  The LBM can be improved by adding the PLSM. This process consists of the following steps. First, the distribution functions at each lattice are used to evaluate the macroscopic velocity field. Then, the macroscopic velocity field is extrapolated using the fast marching method to calculate velocities in gas lattices. Third, the

extrapolated velocity field advects the level set function and particles. Fourth, the advected particles correct errors in the level set function as done in the PLSM. Finally, the density difference between the LBM and the PLSM solvers is added to distribution functions to correct errors of the LBM. The proposed algorithm, called the hybrid lattice Boltzmann method (HLBM), improves the quality of the simulation and offers more splashing effect than the LBM without increasing the number of grids.

Multicomponent-multiphase LBM (MCMP-LBM) is useful for multiple chemical components and multiple phases. The interfaces between different components and phases originate from the specific interactions among fluid molecules. Thus, the macroscopic Navier-Stokes equations is not suitable for solving such microscopic interactions. To get a fast and realistic bubble simulation, we improve the MCMP-LBM by incorporating the PLSM.

- A new MCMP hybrid LBM is proposed to improve the liquid-bubble surface simulation.

  The MCMP-LBM can be improved by adding the PLSM. This process consists of the following steps. First, the distribution functions at each lattice are used to evaluate the composite macroscopic velocity field. Then, the level set function and particles are advected by the the composite velocity field, and advected particles are used to correct errors in the level set function based on the PLSM. Finally, the density difference between the LBM and the PLSM solvers is added to the distribution functions to correct the errors of the MCMP-LBM. The proposed algorithm, called

the MCMP hybrid lattice Boltzmann method (MCMP-HLBM), improves the quality of the simulation and offers more detailed visual results than the MCMP-LBM without increasing the number of grids.

1.4 Organization of the Thesis

The thesis is organized as follows. In Chapter 2, the Navier-Stokes equations for incompressible viscous fluids and some basic numerical techniques to solve them are reviewed. Discretization methods in the computational domain are described, and related previous work is reviewed. The basic level set algorithm and the PLSM are explained in Chapter 3. Next, basic LBM is reviewed and the hybrid LBM is proposed in Chapter 4. The MCMP-LBM is studied and the MCMP-HLBM is proposed in Chapter 5. Finally, concluding remarks are given in Chapter 6.

Chapter 2

Background Review

Some background knowledge needed for this research will be reviewed in this chapter. Navier-Stokes equations (NSEs) for incompressible viscous fluid will be described in Sec. 2.1, the Eulerian and Lagrangian methods to solve NSEs in two different computational domains will be studied in Sec. 2.2, and numerical techniques to solve NSEs will be discussed in Sec. 2.3. Finally, some related previous work will be examined in Sec. 2.4.

2.1 Navier-Stokes Equations (NSEs) for Incompressible Viscous Fluid

Navier-Stokes equations (NSEs) provide a good mathematical model for fluid flows. Its origin can be traced back to Newton [77] who formulated a set of basic equations for theoretical description of fluids about 300 years ago. About half a century later, basic equations for momentum conservation and pressure were developed by Euler [2]. Navier [74] worked on the fluid mechanics equations at the end of the 18th century, as did Stokes [76] several years later. Navier solved the problem for viscous fluid flow analytically. NSEs

in a general setting could not be practically solved until the middle of the 20^{th} century, when numerical techniques needed to solve the resulting equations were developed. For more information on fluid mechanics in general and NSEs in particular can be found in [35].

Conservation of mass is one of the important properties in fluid mechanics, as the mass of the fluid has to be constant for a given fluid system. This is ensured by the following continuity equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0, \tag{2.1}$$

where the partial derivative of the second term is written in Einstein notation. That is, subscript i appearing twice denotes a sum over all possible coordinates [75]. In this case, it is a sum over the three spatial dimensions. For a fluid of constant density, Eq. (2.1) can be simplified as

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.2}$$

which means that the velocity field is divergence free to conserve mass. In other words, the in-flux and the out-flux of a fixed volume are the same.

NSEs with the mass-conserving constraint can be written as

$$\underbrace{\rho\left(\frac{\partial u_j}{\partial t} + u_i \frac{\partial u_j}{\partial x_i}\right)}_{advection} + \underbrace{\frac{\partial P}{\partial x_j}}_{pressure} + \underbrace{\frac{\partial \tau_{ij}}{\partial x_i}}_{momentum} = \rho g_j, \quad j = 1, 2, 3. \tag{2.3}$$

Equation (2.3) consists of three parts. The first part is responsible for the mass force such as advection. The second part is the partial derivative of pressure P, which represents the surface force acting on the fluid. The third part, which is also the most complicated part,

contains tensor τ_{ij} and introduces the momentum effect due to molecular movement. Being similar to the friction between fluid layers, it is attributed to the momentum exchange during the Brownian motion process of molecules.

For Newtonian fluids, *i.e.,* fluids with a viscosity that is independent of the shear rate, tensor τ_{ij} can be written as

$$\tau_{ij} = -\nu \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right) + \frac{2}{3} \delta_{ij} \mu \frac{\partial u_k}{\partial x_k}, \tag{2.4}$$

where μ denotes the dynamic shear viscosity, a value depending on the physical properties of the fluid, ν denotes the kinematic viscosity, which is related to the dynamic viscosity by

$$\nu = \mu / \rho,$$

and the Kronecker symbol denotes a tensor with $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$, otherwise. Since τ_{ij} can be computed by Eq. (2.4), this leaves six unknown variables in Eq. (2.2) and Eq. (2.3). They are the pressure, the three velocity components, the density and the viscosity. For the incompressible (ρ = const) and Newtonian (ν = const) fluid, the four remaining unknowns (*i.e.,* pressure and three velocities) can be solved by the simplified equations via

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.5}$$

which is the continuity equation, and

$$\rho \left(\frac{\partial u_j}{\partial t} + u_i \frac{\partial u_j}{\partial x_i} \right) + \frac{\partial P}{\partial x_j} = \mu \frac{\partial^2 u_j}{\partial x_i^2} + \rho g_i, \tag{2.6}$$

which is the momentum conservation equations of the velocity field. They can also be written in conventional notation as

$$\nabla \cdot \mathbf{u} = 0, \tag{2.7}$$

and

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} - \frac{1}{\rho}\nabla P + \mathbf{f}, \tag{2.8}$$

where $\mathbf{f}$ is the external force such as the gravitational force.

The continuity equation of the velocity field means that the velocity field should be divergence free. In other words, the influx and outflux of a region should be the same. Thus, Eq. (2.7) enables the velocity field to be incompressible. The momentum conservation equation can be factored into four parts: the advection, the diffusion, the pressure, and the force terms.

- The advection term, $-(\mathbf{u} \cdot \nabla)\mathbf{u}$, transports velocity by the velocity field. For example, if there is a smoke particle in front of a fan which is rotating with the same angular speed, the velocity field built by the fan will convey the particle to another position by advection. The particle has the same velocity before and after the conveyance if there is no external force such as the gravitational force.

- The diffusion term, $\nu \nabla^2 \mathbf{u}$, explains the spreading effect of fluids. If there is smoke of higher density as compared to its neighbors, it will spread out to its neighbors at a rate of the diffusion coefficient, ν.

- The pressure term accounts for the propagation of the external force in a fluid. If an external force is applied to a fluid, it is not instantaneously propagated to the entire volume, but pushes molecules closer to the force. Thus, the pressure plays a similar role of the force for the area which is farther away.

The momentum conservation equation is non-linear due to the advection term, $-(\mathbf{u} \cdot \nabla)\mathbf{u}$. We have to deal with the advection term carefully, which will be studied in Sec. 2.3.2. Other attributes such as the density and the temperature also have the momentum conservation equations in form of

$$\frac{\partial a}{\partial t} = -(\mathbf{u} \cdot \nabla)a + \nu_a \nabla^2 a - \alpha_a + S_a, \tag{2.9}$$

where ν_a is a diffusion constant, α_a is the dissipation rate, and S_a is the source term.

In fluid mechanics, these equations are usually expressed in dimensionless form. This is valid as fluids behave similarly at different size and time scales when they have the same Reynolds number (Re), which is a dimensionless value and can be calculated as

$$Re = \frac{UL}{\nu}, \tag{2.10}$$

where U is the macroscopic flow speed and L is the characteristic length (or the distance) of the problem. Thus, a fluid with a given velocity and viscosity behaves similarly to one of a lower velocity and correspondingly smaller viscosity. For the same reason, two problems are comparable when the flow speed is increased while the characteristic length is decreased by the same factor. As an application, to measure the flow around an

aerodynamic body, wind tunnels of a smaller model and an increased flow speed are often used [69]. More details on NSEs, their derivation and applications can be found in many textbooks on fluid dynamics, such as [35] and [13]. Numerical techniques to solve NSEs will be studied in Sec. 2.3.

2.2 Discretization Methods

2.2.1 Eulerian and Lagrangian Methods

There are two typical methods to formulate the computational problem: the Eulerian method and the Lagrangian method. They are compared below.

- The Lagrangian method handles the computational domain from the fluid viewpoint and solves NSEs using the finite element method (FEM) [38, 85]. The grid is attached to the fluid so that it moves with the fluid. It is computationally efficient, good for irregular geometry, and good for small deformation.

- The Eulerian (grid) method handles the computational domain from the spatial description and solves NSEs using the finite difference method (FDM) [28, 59]. The grid is fixed in space and time. It is computationally inefficient, difficult to treat irregular geometries, and good for large deformation.

If the cyan region inside a triangle as shown in Fig. 2.1 (a) is the material (such as a fluid) which deforms by a vector field, the Lagrangian description represents the triangle with lots of particles (red dots) and trace particle positions and attributes as time proceeds as shown in Fig. 2.1 (b). In contrast, the Eulerian description partitions

the entire domain with cells uniformly as shown in Fig. 2.1 (c), where each cell contains the velocity and attributes as time goes on. Some cells will have both white (air) and cyan (liquid) materials at the same time.

We need a way to represent the material of interest. The material field was considered in [30] while the level set (LS) function was proposed in [47]. The LS function will be described in Sec. 3.2. Also, the Lattice-Boltzmann method was considered in [70], which will be reviewed in Sec. 4.2.

2.2.2 Cell-centered and Staggered Grids

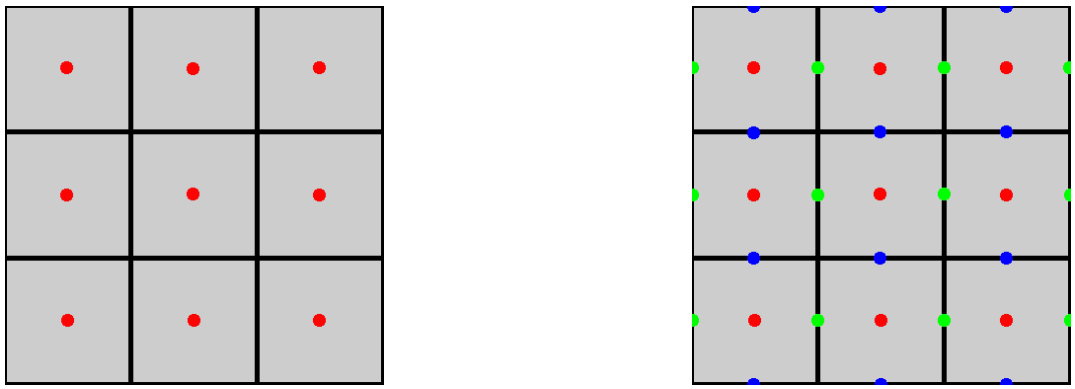
Variables such as velocity, pressure, and temperature are stored in each cell using the Euler method. There exist two grid structures: the cell-centered grid and the staggered grid. They will be studied in this subsection.

Fig. 2.2 shows the cell-centered and the staggered grid methods to form the grid structure. The cell-centered grid assumes that there is only one particle at the center of each cell as shown in Fig. 2.2(a), and the particle has attributes such as the velocity, pressure, and temperature. In contrast, the staggered grid assumes that there are one

(a) The computational domain (b) The Lagrangian method (c) The Eulerian method

Figure 2.1: Comparison of the Eulerian and the Lagrangian descriptions.

particle at the center and 4 particles (for 2D case) on faces of each cell. The center particle only has scalar attributes such as pressure, temperature, and the LS function while face particles only have vector attributes such as the velocity. Red dots as shown in Fig. 2.2(b) represent particles that contain scalar quantities such as pressure and temperature. Green and blue dots contains the x- and y- component of vector quantities such as the horizontal and the vertical velocities, respectively.



(a) The cell-centered grid (b) The staggered grid

Figure 2.2: Illustration of the cell-centered and the staggered grids.

The staggered grid has some advantages as compared to the cell-centered grid since vector quantities on the face can be calculated more naturally by the finite difference discretization of scalar quantities at the center of cells.

2.3 Numerical Techniques to Solve NSEs

NSEs, *i.e.,* Eq. (2.8), can be rewritten as three equations in three coordinates as

$$\begin{aligned}
&\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}+\frac{\partial uw}{\partial z}=-\frac{1}{\rho}\frac{\partial P}{\partial x}+f_x+\nu(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}),\\
&\frac{\partial v}{\partial t}+\frac{\partial vu}{\partial x}+\frac{\partial v^2}{\partial y}+\frac{\partial vw}{\partial z}=-\frac{1}{\rho}\frac{\partial P}{\partial y}+f_y+\nu(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}),\\
&\frac{\partial w}{\partial t}+\frac{\partial wu}{\partial x}+\frac{\partial wv}{\partial y}+\frac{\partial w^2}{\partial z}=-\frac{1}{\rho}\frac{\partial P}{\partial z}+f_z+\nu(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}),
\end{aligned} \tag{2.11}$$

where $\mathbf{u}=(u,v,w)$ and $\mathbf{f}=(f_x,f_y,f_z)$. To solve Eq. (2.11) numerically, we must discretize them first. The simplest method to discretize the time derivative is to use the first-order accurate forward Euler method. The midpoint method is commonly used because it is of second-order accuracy. For the spatial discretization using the stagger grid system, Eq. (2.11) can be discretized using explicit finite difference approximation as [15]

$$\begin{aligned}
\widetilde{u}_{i+1/2,j,k} = u_{i+1/2,j,k} &+ \delta t\{(1/\delta x)[(u_{i,j,k})^2-(u_{i+1,j,k})^2]\\
&+(1/\delta y)[(uv)_{i+1/2,j-1/2,k}-(uv)_{i+1/2,j+1/2,k}]\\
&+(1/\delta z)[(uv)_{i+1/2,j,k-1/2}-(uv)_{i+1/2,j,k+1/2}]+f_x\\
&+(1/\rho\delta x)[P_{i,j,k}-P_{i+1,j,k}]\\
&+(\nu/\delta x^2)(u_{i+3/2,j,k}-2u_{i+1/2,j,k}+u_{i-1/2,j,k})\\
&+(\nu/\delta y^2)(u_{i+1/2,j+1,k}-2u_{i+1/2,j,k}+u_{i+1/2,j-1,k})\\
&+(\nu/\delta z^2)(u_{i+1/2,j,k+1}-2u_{i+1/2,j,k}+u_{i+1/2,j,k-1})\},
\end{aligned} \tag{2.12}$$

for each velocity component u, v, and w of cell (i,j,k). In Eq. (2.12), δx, δy, and δz represent the x-, y-, and z-directional grid spacings, respectively, δt means the time

difference for time discretization, and $\widetilde{u}$ is the velocity component of the x-direction at the next time step.

The implementation of Eq. (2.12) is straightforward. However, there is a stability constraint on the spatial spacing and the time step in numerical integration:

$$\max\left[u\frac{\delta t}{\delta x}, v\frac{\delta t}{\delta y}, w\frac{\delta t}{\delta z}\right] < 1, \tag{2.13}$$

which is called the CFL condition [13].

Some velocities do not lie on cell faces and should be approximated by averaging over the nearest available values, *e.g.,*

$$u_{i,j,k} = \frac{1}{2}(u_{i+1/2,j,k} + u_{i-1/2,j,k}).$$

However, updated velocities, $(\widetilde{u}, \widetilde{v}, \widetilde{w})$, do not satisfy Eq. (2.7) since they are not divergence free. Thus, as explained in [15], one can calculate the divergence of each cell and

set it proportional to the pressure difference, δp. After that, the pressure difference was used to update velocities as

$$\begin{aligned}
u_{i+1/2,j,k} &= u_{i+1/2,j,k} + (\delta t/\delta x)\delta P \\
u_{i-1/2,j,k} &= u_{i-1/2,j,k} - (\delta t/\delta x)\delta P \\
v_{i,j+1/2,k} &= u_{i,j+1/2,k} + (\delta t/\delta y)\delta P \\
v_{i,j-1/2,k} &= u_{i,j-1/2,k} - (\delta t/\delta y)\delta P \\
w_{i,j,k+1/2} &= u_{i,j,k+1/2} + (\delta t/\delta z)\delta P \\
w_{i,j,k-1/2} &= u_{i,j,k-1/2} - (\delta t/\delta z)\delta P
\end{aligned} \tag{2.14}$$

and the cell pressure is updated according to

$$\widetilde{P}_{i,j,k} = P_{i,j,k} + \delta P. \tag{2.15}$$

After updating one cell by the other, the velocity field is still not divergence free. To address this issue, one perform the iteration until the divergence of all cells is less than a certain threshold. Stam [60] introduced the Helmholtz-Hodge decomposition to the computer graph (CG) community to maintain the divergence free condition in the momentum conservation equation. By the Helmholtz-Hodge decomposition, any vector field can be decomposed into the sum of a divergence-free vector field and the gradient of a scalar field as

$$\mathbf{w} = \mathbf{u} + \nabla P, \tag{2.16}$$

where $\nabla \cdot \mathbf{u} = 0$. By applying the divergence to Eq. 2.16, we get the well known Poisson equation as

$$\nabla \cdot \mathbf{w} = \nabla \cdot \mathbf{u} + \nabla^2 P = \nabla^2 P. \tag{2.17}$$

This decomposition is called the projection step, $\mathbf{P}$, and Eq. (2.16) can be written as

$$\mathbf{u} = \mathbf{P}(\mathbf{w}) = \mathbf{w} - \nabla P. \tag{2.18}$$

Finally, Eq. (2.7) and Eq. (2.8) can be merged into one equation as

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{P}(-(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}), \tag{2.19}$$

which is called the modified Navier-Stokes equation. The Poisson equation can be solved using multigrid methods [21].

A general method to solve Eq. (2.19) is to split it into 4 parts: force application, diffusion, advection, and projection [60]. Fig. 2.3 shows the pipeline to solve Eq. (2.19). For the advection term, $-(\mathbf{u} \cdot \nabla)\mathbf{u}$, Stam [60] used the semi-Lagrangian advection which is unconditionally stable. This method traces backward the point according to the velocity field. The implementation of the diffusion, advection, and projection steps will be discussed in the coming subsections.

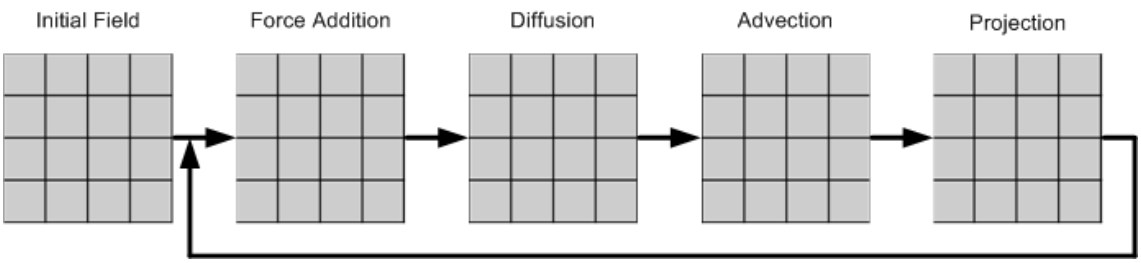


Figure 2.3: The basic structure to solve the modified NSE.

2.3.1 Diffusion

The diffusion equation can be written as

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \nabla^2 \mathbf{u}, \tag{2.20}$$

where ν is the diffusion coefficient. For a smaller diffusion coefficient, we can discretize the diffusion term using the finite-difference method (FDM) as

$$\begin{aligned} \widetilde{u}_{i+1/2,j,k} = u_{i+1/2,j,k} + \delta t\{&(\nu/\delta x^2)(u_{i+3/2,j,k} - 2u_{i+1/2,j,k} + u_{i-1/2,j,k}) \\ &+(\nu/\delta y^2)(u_{i+1/2,j+1,k} - 2u_{i+1/2,j,k} + u_{i+1/2,j-1,k}) \\ &+(\nu/\delta z^2)(u_{i+1/2,j,k+1} - 2u_{i+1/2,j,k} + u_{i+1/2,j,k-1})\}. \end{aligned} \tag{2.21}$$

However, the above discretization scheme cannot be applied to the case with a large diffusion coefficient since the resultant scheme is not stable and the velocity oscillates for a larger diffusion coefficient. To overcome this problem, Stam [61] adopted an implicit method and solved the system by an iterative method known as the Gauss-Seidel

relaxation method [78]. The final discretization scheme using an implicit method can be written as

$$\begin{aligned}\widetilde{u}_{i+1/2,j,k} = u_{i+1/2,j,k} + \delta t\{&(\nu/\delta x^2)(\widetilde{u}_{i+3/2,j,k} - 2\widetilde{u}_{i+1/2,j,k} + \widetilde{u}_{i-1/2,j,k}) \\ &+(\nu/\delta y^2)(\widetilde{u}_{i+1/2,j+1,k} - 2\widetilde{u}_{i+1/2,j,k} + \widetilde{u}_{i+1/2,j-1,k}) \\ &+(\nu/\delta z^2)(\widetilde{u}_{i+1/2,j,k+1} - 2\widetilde{u}_{i+1/2,j,k} + \widetilde{u}_{i+1/2,j,k-1})\}.\end{aligned} \tag{2.22}$$

Stam [61] used twenty iterations of the Gauss-Seidel relaxation to solve Eq. (2.22). The implicit method is stable regardless of δt and the grid size.

2.3.2 Advection

The advection equation can be written as

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u}. \tag{2.23}$$

As mentioned in Sec. 2.1, the advection equation is non-linear. It can be discretized using FDM as

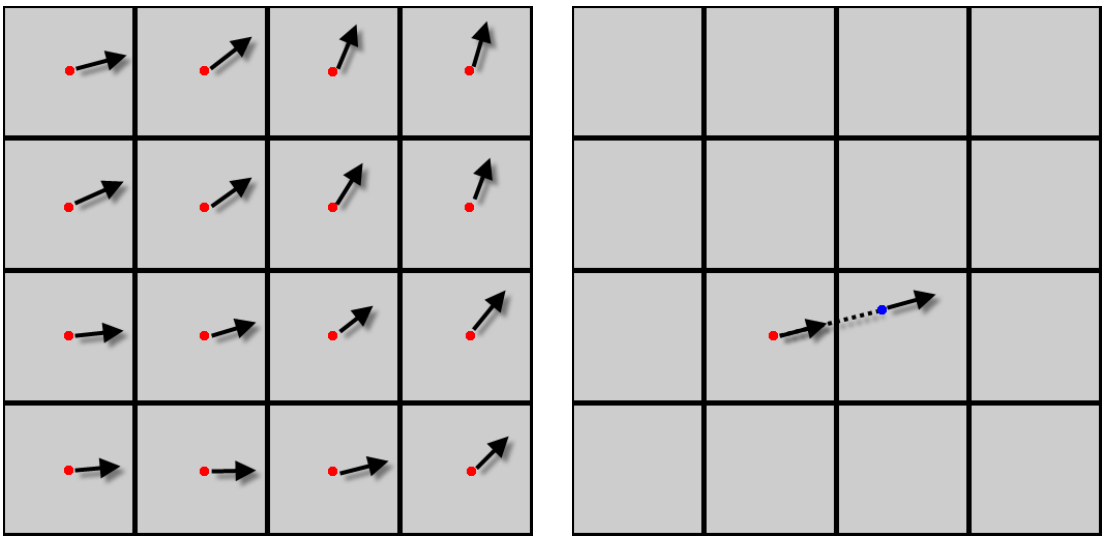
$$\begin{aligned}\widetilde{u}_{i+1/2,j,k} = u_{i+1/2,j,k} + \delta t\{&(1/\delta x)[(u_{i,j,k})^2 - (u_{i+1,j,k})^2] \\ &+(1/\delta y)[(uv)_{i+1/2,j-1/2,k} - (uv)_{i+1/2,j+1/2,k}] \\ &+(1/\delta z)[(uv)_{i+1/2,j,k-1/2} - (uv)_{i+1/2,j,k+1/2}]\}.\end{aligned} \tag{2.24}$$

Eq. (2.24) has a tight stability condition, known as the CFL condition [13]. Therefore, both δt and the grid size have to be very small for the solution to be stable.

Stam [60] proposed a semi-Lagrangian method to find an unconditionally stable solution of the advection equation. As mentioned in Sec. 2.1, the advection term transports

the material by the velocity field. This process is called the forward advection. If there is a particle at the center of each cell, the particle will move to another location by following the velocity to the corresponding location.

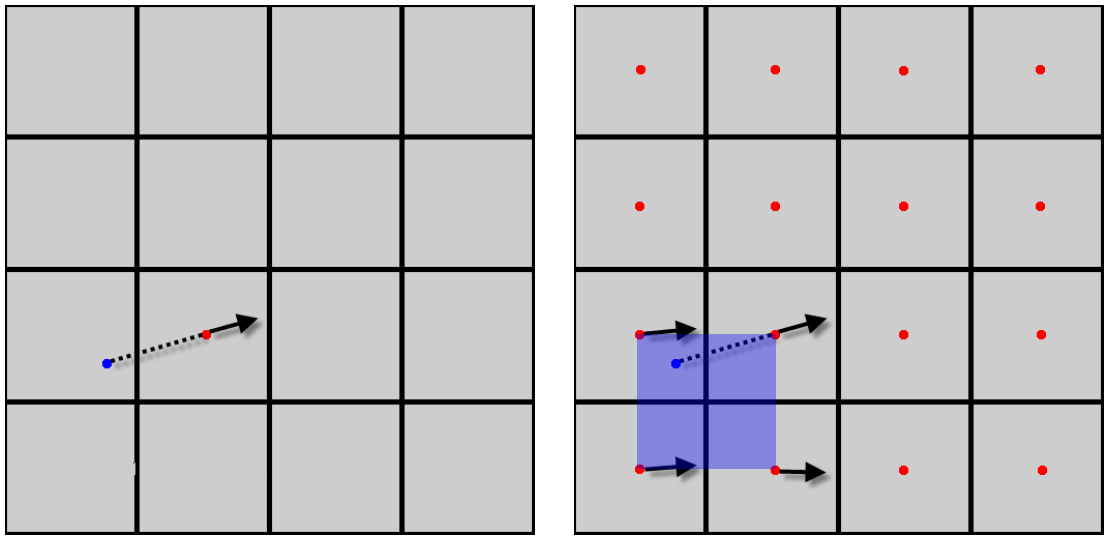
Consider the velocity field at time t_0 as shown in Fig. 2.4(a). In one time step, the particle moves along the direction of the velocity and the distance between the original and new location is equal to $\mathbf{u}\delta t$ by the forward advection method. The particle at the red dot position will move to the blue dot position as shown in Fig. 2.4(b). The particle at red and blue dots have the same attributes (*e.g.,* velocity, density, and pressure) since all attributes are transported from the red dot to the blue dot. Again, the forward advection method is stable only when the time step is sufficiently small such that $\triangle t < \Delta\tau/|\mathbf{u}|$, where $\Delta\tau$ is the grid spacing [60]. It is also difficult to estimate the velocity field from particles that are located at irregular positions.



(a) The original velocity field (b) The forward advection

Figure 2.4: Illustration of the forward advection method.

To overcome the difficulty of the forward advection method, Stam [60] proposed a backward advection method, which is also called the semi-Largrangian advection method. The backward advection method traces particles backward in time. The same red particle can be traced backward in time as shown in Fig. 2.5(a). The direction of tracing is the opposite of the velocity and the distance of tracing is the same as the forward advection case, *i.e.,* $\mathbf{u}\delta t$. As a result, the blue dot denotes the new location of the particle, which was in the red dot position at the previous time step. In other words, a particle moves from the blue dot position to the red one in one time step. The particle at the red dot position has the same attributes as that at the blue dot position. To find attributes of the blue dot, we can use a simple bi-linear interpolation scheme (for the two dimensional case) of the nearest 4 particles as shown in Fig. 2.5(b).



(a) Red particles goes backward in time (b) Bi-linear interpolation

Figure 2.5: Illustration of the backward advection method.

The backward advection method is unconditionally stable and easy to implement. It is also much faster than the direct discretization method using FDM.

2.3.3 Projection

The projection step is to conserve the divergence free property of the velocity field. To do that, we have to solve the Poisson equation with the Neumann boundary condition. The Poisson equation become a sparse linear system when spatially discretized and can be solved using the preconditioned conjugate gradient method. After finding the pressure value at each cell center, the velocity update can be done by Eq. (2.14).

2.3.4 Spatial Discretization

Numerical solutions of NSEs include many time and spatial discretizations. Also, the level set (LS) equation that comes from the evolution of the LS function needs time and spatial discretizations. For spatial discretization, we use upwinding schemes because they approximate derivatives by biasing the finite difference stencil in the direction where the characteristic information comes from [47]. The LS equation describes the evolution of the LS function in time and space, which can be written as

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0. \tag{2.25}$$

Its one-dimensional version, discretized by the forward Euler mehtod, is given by

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \mathbf{u}^n \phi_x^n = 0. \tag{2.26}$$

The upwind discretization checks whether the current grid has positive or negative u_i. If $u_i > 0$, we define ϕ_x as ϕ_x^-. If $u_i < 0$, we define ϕ_x as ϕ_x^+ where

$$\begin{aligned}(\phi_x^+)_i &= D^+\phi_i = \frac{\phi_{i+1} - \phi_i}{\delta x}, \\ (\phi_x^-)_i &= D^-\phi_i = \frac{\phi_i - \phi_{i-1}}{\delta x}.\end{aligned} \tag{2.27}$$

When $u_i = 0$, $u_i(\phi_x)_i$ vanishes so that we do not have to approximate ϕ_x at that grid position.

Eq. (2.27) provides the first-order approximation to Eq. (2.26) since $D^+\phi$ and $D^-\phi$ are of the first-order accuracy. The Hamilton-Jacobi ENO (HJ ENO) [25] is a higher order upwinding scheme, which is more accurate than the simple first-order upwinding scheme. It uses the idea of essentially nonoscillatory (ENO) polynomial interpolation for the numerical solution of the conservation laws. The HJ weighted ENO (WENO) scheme [34] takes a convex combination of ENO approximations to remove the overkill in smooth regions where the data are well behaved. In this work, we use the 5th order HJ WENO scheme for the spatial discretization of the LS equation.

2.3.5 Time Discretization

Although we use the 5th order accurate HJ WENO scheme for the spatial discretization of Eq. (2.25), the forward Euler time discretization is only of first-order accuracy in time. Thus, we examine higher order time discretization schemes in this subsection.

The total variation diminishing (TVD) Runge-Kutta (RK) method is a higher order time discretization scheme which enables us to obtain accurate numerical solutions. The first-order TVD RK scheme is the same as the forward Euler method. The second-order

TVD RK scheme is known as the midpoint rule or the modified Euler method. For Eq. (2.25), we first take an Euler step to advance the solution to time $t^n + \delta t$ as

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \mathbf{u}^n \phi_x^n = 0, \tag{2.28}$$

which is followed by the second Euler step to advance the solution to time $t^n + 2\delta t$ as

$$\frac{\phi^{n+2} - \phi^{n+1}}{\Delta t} + \mathbf{u}^{n+1} \phi_x^{n+1} = 0. \tag{2.29}$$

By taking the average of ϕ^n and ϕ^{n+2}, which is calculated in the second Euler step, ϕ^{n+1} can be found by

$$\phi^{n+1} = \frac{1}{2}\phi^n + \frac{1}{2}\phi^{n+2}. \tag{2.30}$$

The third-order accurate TVD RK scheme takes the first and the second Euler steps as given in Eq. (2.28) and Eq. (2.29) with an averaging step:

$$\phi^{n+\frac{1}{2}} = \frac{3}{4}\phi^n + \frac{1}{4}\phi^{n+2}. \tag{2.31}$$

Then, another Euler step is taken to advance the solution to time $t^n + \frac{3}{2}\delta t$ as

$$\frac{\phi^{n+\frac{3}{2}} - \phi^{n+\frac{1}{2}}}{\Delta t} + \mathbf{u}^{n+\frac{1}{2}} \phi_x^{n+\frac{1}{2}} = 0, \tag{2.32}$$

which is followed by the second averaging step

$$\phi^{n+1} = \frac{1}{3}\phi^n + \frac{2}{3}\phi^{n+\frac{3}{2}}. \tag{2.33}$$

In our work, we use the midpoint or the third-order TVD RK methods for time discretization of NSEs and the LS equation by their applications.

2.4 Numerical Fluid Simulation: A Brief Literature Survey

2.4.1 Finite Differencing and the Particle Level Set Method

Incompressible NSEs were solved using the finite difference discretization by Harlow and Welch in [23], where they proposed a marker-and-cell (MAC) method to track the surface position. They also presented boundary conditions at the rigid wall and the free surface.

Upon the basis of [23], Foster and Metaxas [15] performed realistic liquid animation in the three dimensional (3D) space. They discretized 3D NSEs using the finite difference discretization on the staggered grid structure. They also introduced a height field to characterize the change in local surface elevation at each time step using the local fluid velocity (*i.e.*, by examining the vertical component of the fluid motion and the horizontal convection of the surface elevation from adjacent cell columns). However, their method demands a small time step imposed by the CFL condition [13].

Stam [60] divided the solution to NSEs into 4 parts (*i.e.,* force application, advection, diffusion, and projection) and intoduced the simi-Lagrangian advection scheme to make the advection step stable even with a large time step. He also used the relaxation method to solve the Poisson equation in the projection step. More details of the implementation in [60] was described in [61].

Fedkiw *et al.* [12] proposed the ghost fluid method (GFM) that removed nonphysical oscillations near material interfaces, especially in a multi-material problem associated

with deformable solids. An Eulerian scheme which treats the interface in the Lagrangian way was introduced in [12].

Forster and Fedkiw [14] and Enright *et al.* Enright02b introduced a hybrid particle level set method (PLSM) for liquid animation. A thickened front tracking technique and a velocity extrapolation method were proposed in [11] to improve the particle level set method (PLSM). More recently, Enright *et al.* [9] proposed a fast and accurate semi-Lagrangian PLSM, which enabled fast calculation of NSEs by a semi-Lagrangian advection scheme, which was coupled with a first order accurate fast marching method to evolve the LS function.

Losasso *et al.* [41] used the octree data structure by discretizing the Poisson equation on that grid. The rigid fluid method, a technique for animating the interplay between rigid bodies and a viscous incompressible fluid with a free surface was presented by Carlson *et al.* [6]. They treated rigid bodies as fluids with different rigidity. Losasso *et al.* [42] proposed a multiple LS method for multiple interacting liquids. Overlaps and vacuums of these LS functions were removed by a projection algorithm.

The Lagrangian scheme based on smoothed particle hydrodynamics (SPH) was also developed in [8, 45].

2.4.2 Lattice Boltzman Method (LBM)

The Boltzmann equation is part of the classical statistical physics that describes the behavior of a gas on the microscopic scale. The lattice Boltzmann method (LBM) follows the cellular automata to model a complex system with a set of simple and local rules at each cell [80]. It divides the simulation region into a Cartesian grid of cells, each

of which only interacts with cells in its direct neighborhood. While the conventional physics-based fluid solvers directly discretize NSEs, the LBM is essentially a first-order explicit discretization of the Boltzmann equation in a discrete phase-space. It was shown in [64, 79] that the LBM approximates NSEs with good accuracy. The derivation of the LBM will be given in Sec. 4.2.4. For a more detailed review of the LBM, we refer to [64, 79].

Historically, the LBM evolved from methods in the gas simulation that computes the motion of each molecule in the gas purely with integer operations. Hardy *et al.* [22] made the first attempt to perform fluid simulation with this approach. It took about ten years to discover that the isotropy of lattice vectors is crucial to a correct approximation of NSEs, which was reported in [16]. Motivated by this observation, McNamara and Zanetti [44] developed the first algorithm that was actually called LBM by performing simulations with averaged floating point values instead of single fluid molecules. The basic LBM was further improved by a simplified collision operator with a single time relaxation parameter in [7]. This collision operator, known as the Bhatnagar-Gross-Krook (BGK) approximation [3], was derived independently by Chen *et al.* in [7]. Since then, the LBM has been applied to the solution of many fluid mechanics problems such as the direct numerical simulation of turbulence [82] and the Eulerian-Lagrangian simulation [43], among many others. Moreover, the LBM is available in commercial fluid solvers [39], which are in production use in aerospace and car companies.

Since the LBM can handle problems with a wide range of Knudsen numbers, it can be applied to problems where NSEs are no longer applicable. For example, it can be applied to hypersonic or rarefied gas flows [63].

Chapter 3

Particle Level Set Method for Fluid Simulation

3.1 Introduction

Fluids are represented by smoke and liquid. While both of them follow the rules of NSEs, they are different in the existence of the fluid surface. For a glass of water, the interfaces exist among water, air and glass. The movement of water can be visualized by the movement of surfaces between water and air. To characterize these boundaries numerically, Tryggvason *et al.* [72] discretized the front with particles. The marker particles were used in [23, 50]. The volume of fluid (VOF) method and the level set (LS) method were also used to track interfaces [29, 33, 46]. Among these different numerical methods, the LS method is the most popular one to track interfaces in the Eulerian liquid simulation for several reasons. First, it is easy to extract the geometric information. Second, it is easy to handle merging and breaking interfaces. Third, it is easy to implement in three dimensions. More details of the LS method will be studied in Sec. 3.2.

However, the LS method has some limitations. The LS method cannot preserve the mass and the volume as the simulation goes on. For example, if we pour water into a cup,

the volume of water decreases with time using the LS method. To correct the volume loss, marker particles were used along with the LS method, which results in the particle level set (PLS) method. The PLS method improves the volume loss dramatically. More details of the PLS method will be presented in Sec. 3.3.

3.2 Level Set Method (LSM)

The level set (LS) is an implicit function to construct the surface between fluids. It was introduced to the computer graphics community by Osher and Fedkiw [47] in 2002, and has played a key role in liquid simulation since then.

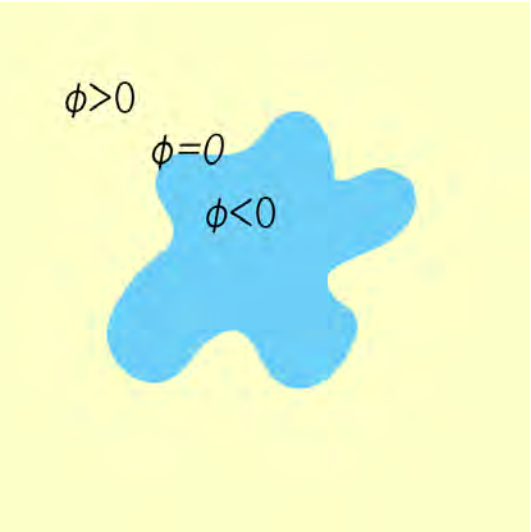


Figure 3.1: The basic structure to the solution of the modified NSE.

Fig. 3.1 shows an example of a liquid (inside of the ellipse) surrounded by the air (outside of the ellipse). The LS function, ϕ, is a scalar function in R^3 defined by

$$\begin{aligned} \phi(\mathbf{x},t) &> 0 \qquad \text{for } \mathbf{x} \in \Omega, \\ \phi(\mathbf{x},t) &\leq 0 \qquad \text{for } \mathbf{x} \in \Omega^c, \end{aligned} \tag{3.1}$$

where $\Omega \subset R^3$. In fluid simulation, the LS function is defined at the center of each cell since it is a scalar function. Although the interface exists between $\mathbf{x} \in \Omega$ and $\mathbf{x} = \Omega$, we use $\mathbf{x} = \Omega$ to denote the surface. The LS function evolves by an externally given velocity field, $\mathbf{u}$, which is obtained by the numerical solution to NSEs. The evolution equation of the LS function is called the level set equation [48], which can be written as

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0. \tag{3.2}$$

Eq. (3.2) can be spatially discretized using the 5th order accurate HJ WENO scheme and temporally discretized using the 3rd order TVD RK scheme as done in [10]. More recently, Enright, Losasso and Fedkiw [9] proposed a fast first-order accurate semi-Lagrangian advection scheme which was coupled with a first-order accurate fast marching method to characterize the evolution of the LS function since higher order schemes do not visually improve the quality of the simulation.

By nature, the LS function is a signed distance function since it means the distance to the interface. Its gradient should have an unit absolute value at all grid points, *i.e.,* $|\nabla \phi = 1|$. Due to this property, the LS function is a smoothly varying function well suited for higher order accurate numerical methods. However, the LS function loses its signed distance property under extreme topological changes. Several reinitialization algorithms had been proposed to maintain the LS function as a signed distance function for all cases. Sussman, Smereka and Osher [65] proposed a method that finds the steady state solution of the following equation:

$$\phi_\tau + \mathrm{sgn}(\phi_0)(|\nabla \phi| - 1) = 0, \tag{3.3}$$

where τ is a fictitious time that can go to infinity, and $\text{sgn}(\phi_0)$ is a one-dimensional smeared out signum function approximated as [65]

$$\text{sgn}(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + (\triangle x)^2}}, \tag{3.4}$$

where $\triangle x$ is the grid spacing for the equidistant Eulerian grid system.

Although Eq. (3.3) can be solved by an iterative numerical solver, it is more efficient to use a fast marching method as proposed in [53, 55]. Also, Eq. (3.3) only has to be solved locally near the interface for computational efficiency. The resulting scheme is called the narrow-band LS method [1]. Geometric quantities can also be easily calculated from the LS function via

$$\begin{aligned} \mathbf{n} &= \frac{\nabla\phi}{|\nabla\phi|}, \\ \kappa &= \nabla \cdot (\frac{\nabla\phi}{|\nabla\phi|}), \end{aligned} \tag{3.5}$$

where $\mathbf{n}$ is the unit normal and κ is the curvature.

Although the reinitialization process keeps the LS function a signed distance function at the end of each time step, it cannot prevent the volume loss caused by smoothly varying property of the LS function.

3.3 Particle Level Set Method (PLSM)

The particle level set method (PLSM) was proposed by Enright *et al.* [10] to overcome the volume loss of LSM. PLSM is a thickened front tracking algorithm that uses massless marker particles to assist LSM in tracking flow characteristics in under-resolved regions around the interface. Particles are labeled by the corresponding LS values with the

positive or the negative sign. Positive particles are located in the band near the interface which has positive LS function values. Negative particles are located in the band near the interface which has negative LS function values. The band, in general, has a width of $3\triangle x$. The number of particles in each cell is usually chosen to be 16 for the 2D grid and 64 for the 3D grid [15]. Each particle has its own LS value and radius. The minimum and maximum radii are $0.1\triangle x$ and $0.5\triangle x$, respectively. The radius of a particle is set according to

$$r_p = \begin{cases} r_{max} & \text{if } s_p\phi(\mathbf{x}_p) > r_{max}, \\ s_p\phi(\mathbf{x}_p) & \text{if } r_{min} \leqslant s_p\phi(\mathbf{x}_p) \leqslant r_{max}, \\ r_{min} & \text{if } s_p\phi(\mathbf{x}_p) < r_{min}, \end{cases} \tag{3.6}$$

where s_p is the sign of the particle.

3.3.1 Advection of Level Set Equation

Eq. (3.2) is the same as the traditional advection equation except that the LS function is advected instead of the velocity. Thus, Eq. (3.2) can be solved using the semi-Lagrangian advection scheme as described in Sec. 2.3.2.

For a given grid, $\mathbf{x}_{i,j} = (i\Delta x, j\Delta y)$, and temporal discretization, $t^n = n\Delta t$, Eq. (3.2) can be discretized by the semi-Lagrangian method as

$$\phi_{i,j}^{n+1} = \alpha\beta\phi_{r+1,s+1}^n + (1-\alpha)\beta\phi_{r,s+1}^n + \alpha(1-\beta)\phi_{r+1,s}^n + (1-\alpha)(1-\beta)\phi_{r,s}^n, \tag{3.7}$$

where

$$
\begin{aligned}
r &= i - \lceil u_{i,j}\frac{\Delta t}{\Delta x}\rceil, \qquad & \alpha &= \left(\frac{(i-r)\Delta x - u_{i,j}\Delta t}{\Delta x}\right), \\
s &= j - \lceil v_{i,j}\frac{\Delta t}{\Delta y}\rceil, \qquad & \beta &= \left(\frac{(j-s)\Delta y - v_{i,j}\Delta t}{\Delta y}\right),
\end{aligned}
\tag{3.8}
$$

and $\mathbf{u}(\mathbf{x}_{i,j}) = (u_{i,j}, v_{i,j})$. As mentioned in Sec. 2.3.2, this method is unconditionally stable. For the time integration of particles, we need the second order accurate TVD RK midpoint rule as proposed in [9].

3.3.2 Error Correction

After the LS function, ϕ, and particles are advected in time, errors in the LS function should be corrected by particles to improve the surface precision. This process consists of three steps as described below.

1. Error Identification

   Particles on the wrong side of the interface by more than their radii are called escaped particles, which indicate the existence of errors in the LS representation of the interface. There are positive and negative escaped particles as shown Fig. 3.2, where red dots denote escaped positive particles, blue dots denote escaped negative particles, cyan and yellow regions represent liquid and air, repectively.

2. Error Quantification

   A spherical LS function, ϕ_p, with each particle p whose size is determined by the particle radius can be written as

$$
\phi_p(\mathbf{x}) = s_p(r_p - |\mathbf{x} - \mathbf{x_p}|). \tag{3.9}
$$

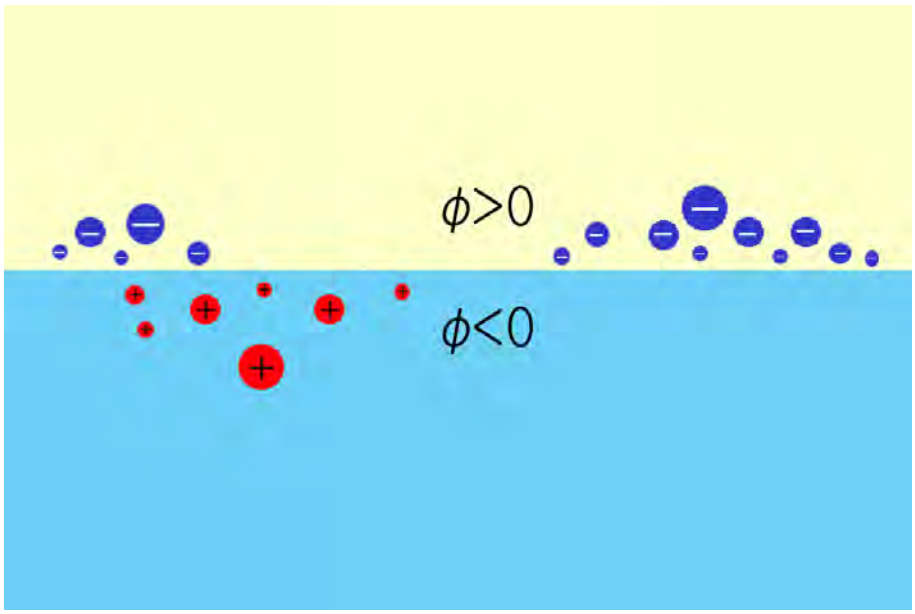


Figure 3.2: Illustration of escaped positive (blue) and negative (red) particles.

The particle-defined LS function is computed locally on the eight corners of the cell containing the particle. The local values of ϕ_p are the particle preconditions of the values of the overall LS function, ϕ, on the corners of the cell. Any variation of ϕ from ϕ_p indicates the magnitude of potential errors in the LS solution.

3. Error Correction

Escaped positive particles are used to rebuild the region of $\phi > 0$ while escaped negative particles are used to rebuild the region of $\phi \leq 0$. The ϕ_p values of eight grid points on the boundary of the cell containing escaped particles are calculated using Eq. (3.9). Next, each ϕ_p is compared with the local value of ϕ, and the maximum of these two defines ϕ^+. This is done for all escaped positive particles.

Given LS ϕ and a set of escaped positive particles E^+, we initialize ϕ^+ with ϕ and then calculate

$$\phi^+ = \min_{\forall p \in E^+} (\phi_p, \phi^+). \tag{3.10}$$

Similarly, to calculate a reduced error representation of the $\phi \leq 0$ region, ϕ^- is initialized with ϕ and then calculated as

$$\phi^- = \min_{\forall p \in E^-} (\phi_p, \phi^-). \tag{3.11}$$

Note that ϕ^+ and ϕ^- may not agree due to errors in particles, the LS method, and interpolation errors, etc. ϕ^+ and ϕ^- are merged back into a single level set by setting ϕ equal to the value of ϕ^+ or ϕ^- which is smaller in the magnitude at each grid point, *i.e.,*

$$\phi = \begin{cases} \phi^+ & \text{if } |\phi^+| \leq |\phi^-|, \\ \phi^- & \text{if } |\phi^+| > |\phi^-|. \end{cases} \tag{3.12}$$

The minimum magnitude is used to reconstruct the interface since the error correction process gives the priority to values that are closer to the interface.

In Fig. 3.3, blue and red circle represents negative and positive escaped particles, respectively. And the radii of the positive and negative escaped particles are 0.32 and 0.11, respectively. (We assure the grid spacing is 1 for this example.) Green circles represent

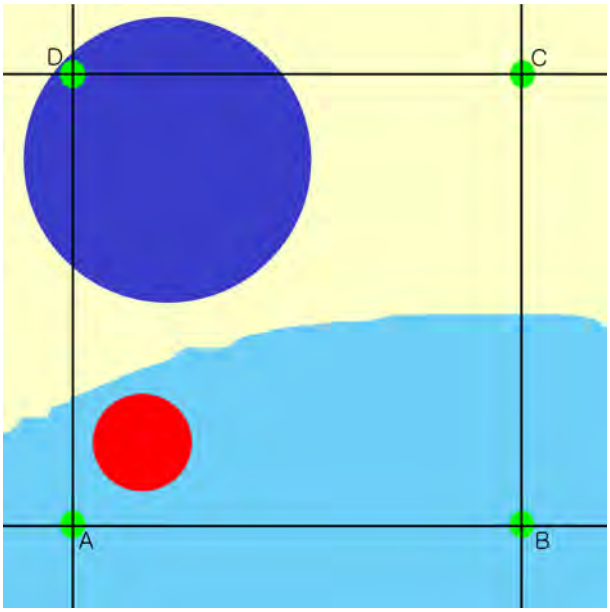


Figure 3.3: Illustration of two escaped particle and their radii to explain error correction process.

the 4 neighboring cells' center. Yellow and cyan mean liquid and gas regions, respectively.

Let's assume

$$\phi(\mathbf{x}_A) = -0.30$$

$$\phi(\mathbf{x}_B) = -0.42$$

$$\phi(\mathbf{x}_C) = +0.55$$

$$\phi(\mathbf{x}_D) = +0.70\ .$$

Then we can calculate spherical LS functions of the positive escaped particle for A, B, C, and D using Eq. (3.9) as

$$\phi_p(\mathbf{x}_A) = +(0.11 - 0.24) = -0.13$$

$$\phi_p(\mathbf{x}_B) = +(0.11 - 0.87) = -0.76$$

$$\phi_p(\mathbf{x}_C) = +(0.11 - 1.18) = -1.07$$

$$\phi_p(\mathbf{x}_D) = +(0.11 - 0.85) = -0.74\ .$$

Sphrical LS functions of the negative escaped particle for A, B, C, and D using Eq. (3.9) as

$$\phi_p(\mathbf{x}_A) = -(0.32 - 0.83) = +0.51$$

$$\phi_p(\mathbf{x}_B) = -(0.32 - 1.10) = +0.78$$

$$\phi_p(\mathbf{x}_C) = -(0.32 - 0.80) = +0.48$$

$$\phi_p(\mathbf{x}_D) = -(0.32 - 0.30) = -0.02 \ .$$

From Eq. (3.10), ϕ^+ can be calculated as

$$\phi^+(\mathbf{x}_A) = \max(-0.13, -0.30) = -0.13$$

$$\phi^+(\mathbf{x}_B) = \max(-0.76, -0.42) = -0.42$$

$$\phi^+(\mathbf{x}_C) = \max(-1.07, +0.55) = +0.55$$

$$\phi^+(\mathbf{x}_D) = \max(-0.74, +0.70) = +0.70 \ .$$

From Eq. (3.11), ϕ^- can be calculated as

$$\phi^-(\mathbf{x}_A) = \min(+0.51, -0.30) = -0.30$$

$$\phi^-(\mathbf{x}_B) = \min(+0.78, -0.42) = -0.42$$

$$\phi^-(\mathbf{x}_C) = \min(+0.48, +0.55) = +0.48$$

$$\phi^-(\mathbf{x}_D) = \min(-0.02, +0.70) = -0.02 \ .$$

Finally from Eq. (3.12), ϕ can be updated as

$$\phi(\mathbf{x}_A) = -0.13$$

$$\phi(\mathbf{x}_B) = -0.42$$

$$\phi(\mathbf{x}_C) = +0.48$$

$$\phi(\mathbf{x}_D) = -0.02\ .$$

As a result, $\phi(\mathbf{x}_A)$ has been changed from -0.30 to -0.13 because the positive escaped particle increased it. $\phi(\mathbf{x}_B)$ has not been changed because B is far from both escaped particles. $\phi(\mathbf{x}_C)$ has been slightly changed from 0.55 to 0.48 because the negative escaped particle decreased it. $\phi(\mathbf{x}_D)$ has been changed from 0.70 to -0.02 because the negative escaped particle decreased and changed the sign of it.

3.3.3 Reinitialization and Radii Adjustment

The LS function, ϕ, is maintained as a signed distance function by solving Eq. (3.3) with a fast marching technique. For the sake of efficiency, ϕ is reinitialized within a band of the interface. The integration of the narrow band optimization scheme and the fast marching method provides a fast reinitilazation procedure. To ensure proper ϕ values for the semi-Lagrangian update, ϕ is reinitialized within a band of $\pm 5 \max(\Delta x, \Delta y)$ of the interface. This procedure is performed after each combined process of the semi-Lagrangian update and error correction. Reinitialization may cause the zero level set to shift, which is not desirable. To overcome this problem, particles are also used to correct these errors. Finally, particles resample their position relative to the zero LS, $\phi = 0$, and

adjust their radii accordingly. Any particles which remain escaped have their radii set to the minimum particle radius value.

3.4 Fast Marching Method (FMM)

The fast marching method (FMM) computes the solution of the Eikonal equation [55] of the following form

$$|\nabla u| = F(x, y). \tag{3.13}$$

The main task is to build an approximation to the gradient term which correctly deals with the development of corners and cusps in the evolving solution. It is well known that the Eikonal equation becomes non-differentiable so that an appropriate weak solution must be developed. This is related to the entropy condition for interface propagation as introduced in [55]. Eq. (3.13) can be discretized as

$$[\max(D_{ij}^{-x}u, -D_{ij}^{+x}u)^2 + \max(D_{ij}^{-y}u, -D_{ij}^{+y}u)^2]^{1/2} = F_{ij} \tag{3.14}$$

FMM is a method that systematically advances the front in an upwind fashion to produce the solution, u. The upwind difference structure in Eq. (3.14) means that information propagates along one way; namely, from smaller values of u to larger values. In other words, FMM solves Eq. (3.14) by building the solution outward from the smallest u value. The algorithm is made fast by confining the building zone to a narrow band around the front.

To achieve this, one can sweep the front ahead in an upwind fashion by considering a set of points in a narrow band around the existing front and march this narrow band forward, freezing the values of existing points and bringing new ones into the narrow band structure. The key is in the selection of the proper grid point in the narrow band to update. The algorithm first classifies points into three sets: *Far*, *Close* and *Accepted*. We tag points as *Accepted* initially, then all points one grid point away as *Close*, and, finally, all other grid points as *Far*. Then, one sweep of the upwind computation can be stated as follows.

1. Initializaton: Let *Trial* be the point in *Close* with the smallest value for u.
2. Move all neighbors of *Trial* that are in *Far* into *Close*.
3. Recompute the values of u at all neighbors of *Trial* that are in *Close* according to Eq. (3.14) by solving the quadratic equation, treating all points in *Close* and *Far* as if they had the value of ∞.
4. Move point *Trial* to *Accepted*.

This algorithm works well since the process of recomputing the values of u at downwind neighboring points cannot yield a value smaller than any of accepted points. Consequently, we can march the solution outward, always selecting the narrow band grid point with the minimum trial value for u, and readjusting neighbors.

Another way to look at this is that each minimum trial value begins with an application of Huyghen's principle, and the expanding wavefront touches and updates all points. The speed of the algorithm depends on a heapsort technique to locate the smallest element in the set *Trial* efficiently. We can perform an operation count of FMM below.

Given N computational points in a domain, we would like to solve the Eikonal equation away from an initial curve (or surface) Γ lying in this domain. By using the heapsort, the smallest such point can be located in $O(logN)$. Furthermore, since each point in the domain is touched only once during the update, the total operation count to solve the Eikonal equation is $O(NlogN)$.

If one would like to produce a solution this is close to the front (one or two points away), one might attempt to iterate the solution as done in [84]. However, except for a small range around the boundary, this approach is less efficient than FMM. Since we will use the algorithm to extend velocities at any distance from the interface, FMM is preferred. For more details, we refer to [54].

3.5 Signed Distance and Velocity Extrapolation

Given a LS function ϕ^n, our goal is to build an extension velocity F_{ext} such that, if $|\nabla\phi| = 1$, the extension velocity maintains the unit gradient. To achieve this, we need to solve the following equation:

$$\nabla\phi^{temp} \cdot \nabla F_{ext} = 0, \tag{3.15}$$

where ϕ^{temp} is the signed distance function which has the same zero level set as the LS function, ϕ^n. It is worthwhile to emphasize that we do not use this computed signed distance to re-initialize the LS function, but use it only in the construction of F_{ext}.

3.5.1 Signed Distance

We use ϕ^n to denote a LS function where superscript n indicates the time step. It does not correspond to the signed distance function. Instead, FMM can be used to compute signed distance ϕ^{temp} by solving the Eikonal equation:

$$|\nabla T| = 1, \tag{3.16}$$

at either side of the interface with boundary condition $T = 0$ in the zero level set of ϕ. Then, T will be the temporary signed distance function ϕ_{temp}.

FMM is run separately for grid points outside and inside the front. The only difficulty lies in the initialization stage of FMM. That is, the computation of approximate distances of points in the set of *Close* to initiate FMM. Values of grid points in the initial set of *Close* lying outside of a 2D front can be determined as follows. First, we label those grid points where one of the neighbors lies inside the front as *Close*. Values at these points must be assigned to approximate the distances to the front. While this can be computed exactly for a smooth front, a faster method that uses the intersection of the front with grid lines only can be designed. This is particularly useful when the front is given as the zero level set of a function defined at grid points and a smooth representation is not available. There are five possible cases to be considered as shown in Fig. 3.4.

- In Fig. 3.4(a), only one of the neighboring points is at the other side of the front. The distance to the intersection point from the line connecting the two grid points is denoted by s. This value is larger than the real distance to the front, but most

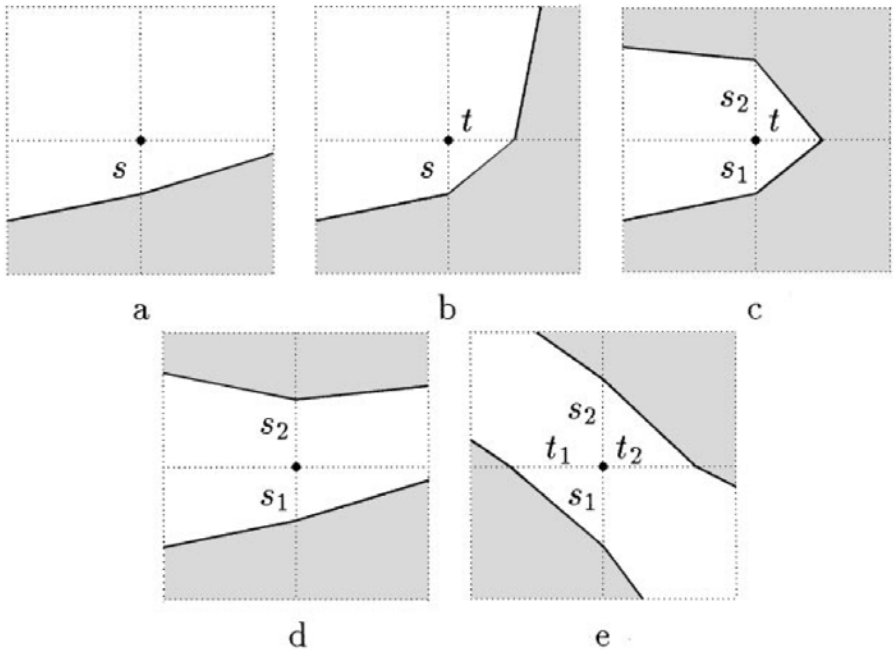


Figure 3.4: Illustration of five cases of a point's neighborhood from Adalsteinsson *et. al.* [1].

likely the value at the grid point at the other side is the distance to the same point, so that the zero level set will not have moved after reinitialization.

- In Fig. 3.4(b), two of the neighbors are at the other side of the front. In this case the value is defined as the exact distance to the line segment between the two intersection points. If s and t are distances to intersection points, the exact distance d satisfies

$$\left(\frac{d}{s}\right)^2 + \left(\frac{d}{t}\right)^2 = 1.$$

The left-hand side is an upwind approximation to the gradient of the distance function, since the distance is zero at the intersection points. This suggests what the solution should be for the remaining three cases and how it should be computed in 3D.

- In Fig. 3.4(c), the distance is the positive solution to

$$\left(\frac{d}{\min(s_1, s_2)}\right)^2 + \left(\frac{d}{t}\right)^2 = 1.$$

- In Fig. 3.4(d), the distance is

$$d = \min(s_1, s_2).$$

- In Fig. 3.4(e), the distance is the positive solution to

$$\left(\frac{d}{\min(s_1, s_2)}\right)^2 + \left(\frac{d}{\min(t_1, t_2)}\right)^2 = 1.$$

3.5.2 Velocity Extrapolation

For advection, the velocity field should be extended along an interface to grid points around the front. This should extend the velocity in a continuous manner and avoid the introduction of any discontinuity in the speed close to the front as much as possible. Mathematically, we want to construct a speed function, F_{ext}, which satisfies the following equation:

$$\nabla F_{ext} \cdot \nabla \phi^{temp} = 0. \tag{3.17}$$

This problem can be solved by marching outwards via FMM systematically and simultaneously by attaching two values to each grid point, *i.e.,* the distance from the front and the extended speed value. The signed distance, ϕ^{temp}, to the front is computed using FMM as described in the previous section. As FMM constructs the signed distance at

each grid point, the value of F_{ext} is updated simultaneously by Eq. (3.17). In the gradient stencil, neighboring points closer to the front are used to maintain the upwind ordering of the point construction. To be more specific, being similar to the construction of signed distances, we have to find the speed values for points in the initial set of *Close* to initiate the process. Then, the extension value is updated whenever the distance value is updated according to Eq. (3.17).

One technique to build extension velocities near the front would be to copy the value of the closest grid point. Here, we take a weighted average of values at points used in computing the distance instead, where the weight is proportional to one over the square of the distance. This is equivalent to solving Eq. (3.17).

As an example, consider the five cases in Fig. 3.4. For simplicity, we compute the extension value at point (i, j) in the center only.

- For Fig. 3.4(a), the extension speed is $f = f(i, j - s)$.

- For Fig. 3.4(b), the gradient is given by

$$\left(-\frac{d}{t}, \frac{d}{s}\right).$$

The discretized equation $\nabla F_{ext} \cdot \nabla \phi^{temp} = 0$ becomes

$$\begin{aligned} 0 &= \left(-\tfrac{f-f(i+t,j)}{t}, \tfrac{f-f(i,j-s)}{s}\right) \cdot \left(-\tfrac{d}{t}, \tfrac{d}{s}\right) \\ &= d\left[\tfrac{f-f(i+t,j)}{t^2} + \tfrac{f-f(i,j-s)}{s^2}\right], \end{aligned}$$

where

$$f = \frac{(1/t^2)f(i+t,j) + (1/s^2)f(i,j-s)}{1/t^2 + 1/s^2}.$$

This gives the solution for the remaining cases and in the 3D space. The above expression assumes that the speed of the interface is given at intersection points of the interface with the grid lines. If it is given at other points, one can either use the interpolation to get the desired values or modify the above algorithm.

- For Fig. 3.4(c), we have

$$f = \frac{(1/t^2)f(i+t,j) + (1/s^2)f(i,j+s)}{1/t^2 + 1/s^2},$$

where $s = s_1$, if $|s_1| < |s_2|$, and $s = s_2$, otherwise.

- For Fig. 3.4(d), we have

$$f = f(i,j+s),$$

where s is chosen as given before.

- For Fig. 3.4(e), we have

$$f = \frac{(1/t^2)f(i+t,j) + (1/s^2)f(i,j+s)}{1/t^2 + 1/s^2},$$

where s and t are chosen, respectively, from entries in (s_1, s_2) and (t_1, t_2) with a smaller absolute value.

Once values for both the signed distance and the extension function are established at *Close* points, we need to update extension values. As the distance value is updated

using FMM, a new extension value is chosen such that $\nabla F_{ext} \cdot \nabla \phi^{temp} = 0$, where the gradient of F_{ext} and ϕ^{temp} are calculated using points that contributed in the update of ϕ. If no points from a grid direction are used, the corresponding component of the gradient is zero.

As an example, consider the case in Fig. 3.4(b), where the new distance value at (i, j) is found by solving Eq. (3.14). Let $(i+1, j)$ and $(i, j-1)$ be points used in updating the distance. If v is the new extension value, it has to satisfy the upwind version of Eq. (3.17), namely

$$\left(\frac{\phi^{temp}_{i+1,j} - \phi^{temp}_{i,j}}{h}, \frac{\phi^{temp}_{i,j} - \phi^{temp}_{i,j-1}}{h} \right) \cdot \left(\frac{F^{temp}_{i+1,j} - v}{h}, \frac{v - F_{i,j-1}}{h} \right) = 0.$$

Since $(i+1, j)$ and $(i, j-1)$ have been accepted, F is defined at those points, and this equation can be solved for v as

$$v = \frac{F_{i+1,j}(\phi^{temp}_{i,j} - \phi^{temp}_{i+1,j}) + F_{i,j-1}(\phi^{temp}_{i,j} - \phi^{temp}_{i,j-1})}{(\phi^{temp}_{i,j} - \phi^{temp}_{i+1,j}) + (\phi^{temp}_{i,j} - \phi^{temp}_{i,j-1})}.$$

This means that $\nabla F_{ext} \cdot \nabla \phi^{temp} = 0$ is satisfied for all points on the grid, except the points along the front itself, at the end. At those points, the previous construction will make the equation satisfied when the gradient approximation is computed using points on the front as mentioned before. Finally, FMM allows one to extend either the normal speed function F or an advective component velocity field (u, v) by extending each component separately. In other words, Eq. (3.17) should be solved for x-, y-, and

z-directional velocity component separately for the velocity extrapolation of the three dimensional space.

3.6 Conclusion

The PLSM provides one important family of numerical methods for fluid simulation in computational fluid dynamics (CFD). The PLSM is good for the smooth surface representation but bad for the global pressure correction step to solve Poisson equation. The PLSM will be combined to the lattice Boltzmann method in Chapter 4 for free surface fluids and Chapter 5 for multicomponent-multiphase fluids.

Chapter 4

Hybrid Lattice Boltzmann Method for Free Surface Fluid Simulation

4.1 Introduction

The lattice Boltzmann (LB) solvers and the conventional NS solvers are two primary tools in simulating fluid flow. These state-of-the-art solvers were compared in [17]. It was concluded that there is no clear winner. For a special class of problems, each type of solver has its advantages and disadvantages. However, the computational complexity of the LB solver is comparable to that of discretizing the corresponding NS problem. We will describe the basic algorithm of the lattice Boltzmann method (LBM) and then present a novel hybrid lattice Boltzmann method (HLBM) for efficient liquid simulation in this chapter.

Generally speaking, a simple LB implementation performs well for complex geometries, since each cell contains information about fluid velocity and pressure as well as spatial derivatives. It even allows an accurate representation of obstacles in coarse grids. The free surface of a fluid usually results in complex and time-dependent topologies. This

motivates the use of the LBM in simulating free surface flows, and the resultant model is simple since the LBM can model complex boundary conditions. The main contribution of this chapter is the proposal of a hybrid lattice Boltzmann method (HLBM), which combines the advantages of the particle level set method (PLSM) and the lattice Boltzmann method (LBM).

The rest of this chapter is organized as follows. The LBM will be introduced in Sec. 4.2. The LBM with a free surface will be examined in Sec. 4.3. Then, the hybrid LBM algorithm that integrates the PLSM with the LBM will be presented in Sec. 4.4. Computer simulation results will be shown in Sec. 4.5. Finally, concluding remarks are given in Sec. 4.6.

4.2 Lattice Boltzmann Method (LBM)

4.2.1 Basic Algorithm

The basic algorithm of the LBM consists of two steps: the streaming step and the collision step. These are usually applied in association with no-slip boundary conditions in domain boundaries or obstacles. The free surface condition is adopted in the simulation of the free surface flow. The simplicity of the basic algorithm is evident. The LBM restricts the particle movement to a limited number of directions. A three dimensional model with 19 velocities (commonly denoted as D3Q19) will be used in this chapter. Alternatives are models with 15 or 27 velocities. However, the latter one has no apparent advantages over the model with 19 velocities while the model with 15 velocities has decreased stability. The D3Q19 model is thus preferable since it demands less memory than the model with

27 velocities. For the 2D case, the model with nine velocities, denoted by D2Q9, is the most common model. The D2Q9 model and the D3Q19 model with its lattice velocity vectors are shown in Fig. 4.1.

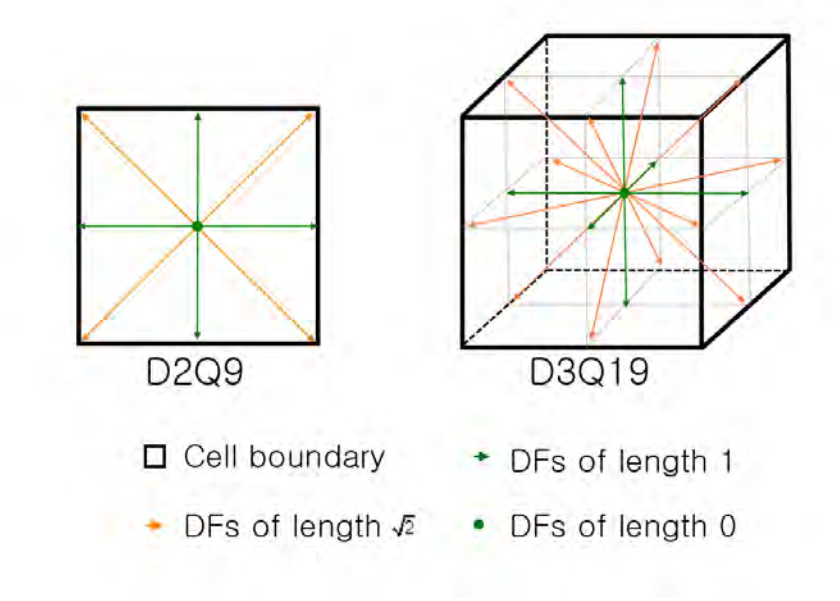


Figure 4.1: The commonly used LB models in two and three dimensions.

The D3Q19 model with its lattice vector $\mathbf{e}_{1..19}$ is examined in detail below. The velocity vectors take the following values: $\mathbf{e}_1 = (0,0,0)^T$, $\mathbf{e}_{2,3} = (\pm 1,0,0)^T$, $\mathbf{e}_{4,5} = (0,\pm 1,0)^T$, $\mathbf{e}_{6,7} = (0,0,\pm 1)^T$, $\mathbf{e}_{8..11} = (\pm 1,\pm 1,0)^T$, $\mathbf{e}_{12..15} = (0,\pm 1,\pm 1)^T$, and $\mathbf{e}_{16..19} = (\pm 1,0,\pm 1)^T$. For each velocity, a floating point number $f_{1..19}$, representing the fraction of particles moving with this velocity, needs to be stored. Thus, in the D3Q19 model, there are particles not moving at all (f_1), moving with speed 1 ($f_{2..7}$), and moving with speed $\sqrt{2}$ ($f_{8..19}$).

In the following, all formulas of the LBM will be expressed in terms of the particle distribution functions (DFs). The subscript of $\tilde{i}$ denotes the value for the inverse direction of a value with subscript i. In other words, f_i and $f_{\tilde{i}}$ are opposite DFs with inverse velocity

vectors, $\mathbf{e}_{\bar{i}} = -\mathbf{e}_i$. During the streaming step, all DFs are advected with their respective velocities so that it is similar to the advection step in the PLSM except that velocities in the LBM are discrete samples in the space. This results in a movement of the floating point value to the neighboring cells as shown in Fig. 4.2.

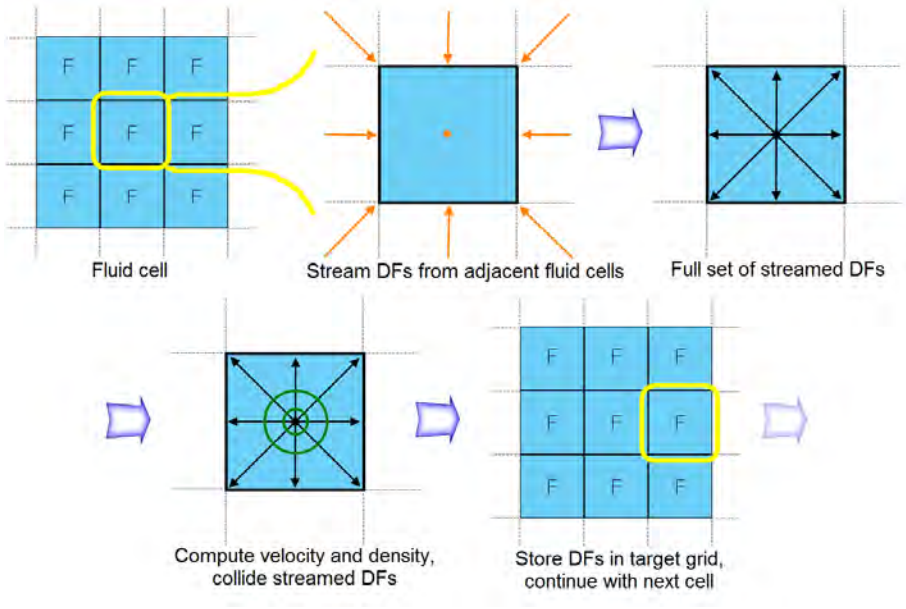


Figure 4.2: Illustration of the streaming step and the collision step for a fluid cell.

Being formulated in terms of DFs, the streaming step can be written as

$$f_i{}^*(\mathbf{x}, t + \triangle t) = f_i(\mathbf{x} + \triangle x \mathbf{e}_{\bar{i}}, t), \tag{4.1}$$

where $\triangle x$ is the size of a cell and $\triangle t$ is the time step-size. They are normalized by $\triangle t / \triangle x = 1$, which makes it possible to handle the advection by a simple copying operation as described above. The streaming step alone is not enough to simulate the behavior of an incompressible fluid, which is governed by the on-going collision of particles with

each other. The collision step accounts for this by weighting DFs of a cell with the so-called equilibrium distribution functions, denoted by ${f_i}^{eq}$, which depend on the density and the velocity of the fluid.

In this work, the incompressible model from [26] is used, which alleviates the compressibility effect of the standard model using a modified equilibrium DF and velocity calculation. The density and the velocity can be computed by the summation of all DFs in one cell; namely,

$$\rho = \sum f_i, \quad \text{and} \quad \mathbf{u} = \sum \mathbf{e}_i f_i. \tag{4.2}$$

For direction i, the equilibrium DF f_i^{eq} can be computed by

$$f_i^{eq} = w_i \left[\rho + 3\mathbf{e}_i \cdot \mathbf{u} - \frac{3}{2}\mathbf{u}^2 + \frac{9}{2}(\mathbf{e}_i \cdot \mathbf{u})^2 \right], \tag{4.3}$$

where

$$w_i = 1/3 \ \text{ for } \ i = 1,$$

$$w_i = 1/18 \ \text{ for } \ i = 2, \cdots, 7,$$

$$w_i = 1/36 \ \text{ for } \ i = 8, \cdots, 19.$$

The equilibrium DFs represent a stationary state of the fluid, which however does not mean that the fluid is still. The values of DFs would not change if the whole fluid was in an equilibrium state. For viscous flows, the equilibrium state (which is equivalent to a Stokes flow) can be globally reached. In this case, DFs will converge towards constant values. The collision of molecules in a real fluid is approximated by linearly relaxing

the DFs of a cell towards their equilibrium state. Thus, each f_i is weighted with the corresponding f_i^{eq} as

$$f_i(\mathbf{x}, t + \triangle t) = (1 - w) f^*(\mathbf{x}, t + \triangle t) + w f_i^{eq}, \tag{4.4}$$

where w is a parameter that controls the viscosity of the fluid. Sometimes, $\tau = 1/w$ is also used to denote the lattice viscosity. Parameter w is in the range of $(0, 2]$, where values close to 0 result in very viscous fluids while values near 2 result in more turbulent flows. Usually, they are more visually interesting. However, when w is close to 2, the method may become unstable. A method to stabilize the computation with a turbulence model will be explained in Sec. 4.2.2. Parameter w is given by the kinematic viscosity of a fluid. Details of the parametrization will be explained in Sec. 4.2.3.

Values computed by Eq. (4.4) are stored as DFs for time $t + \triangle t$. Since each cell needs the DFs of its adjacent cells from the previous time step, two arrays for DFs (*i.e.*, the current and the last time steps) are usually used. The easiest way to implement the no-slip boundary conditions is to apply the link bounce back rule that results in a placement of the boundary halfway between fluid and obstacle cells. If the neighboring cell at $(\mathbf{x} + \triangle t \mathbf{e}_i)$ is an obstacle cell during streaming, the DF from the inverse direction of the current cell is used. That is, we change Eq. (4.1) to

$$f_i^*(\mathbf{x}, t + \triangle t) = f_{\tilde{i}}(\mathbf{x}, t). \tag{4.5}$$

The basic steps for a cell next to an obstacle cell are illustrated in Fig. 4.3.

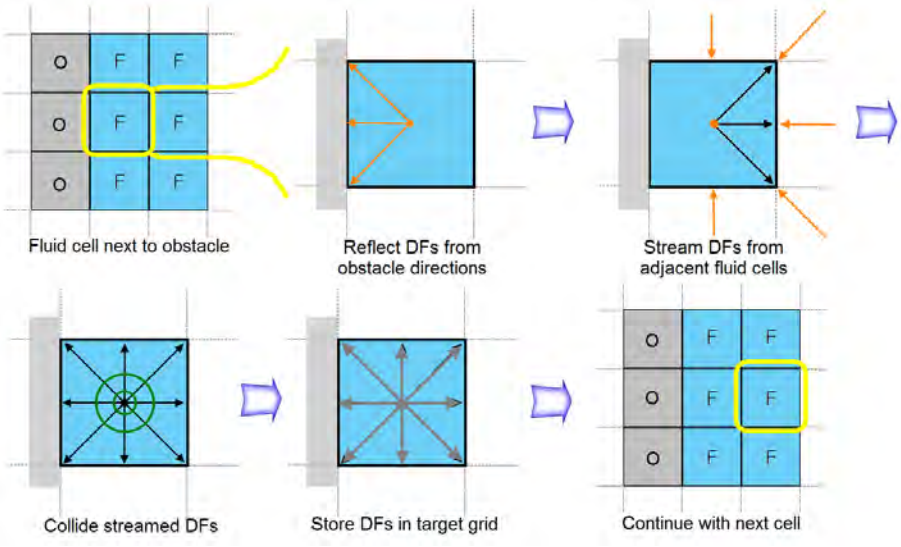


Figure 4.3: The streaming and collision steps for a fluid cell next to an obstacle.

The implementation of the algorithm includes a flag field to distinguish fluid and obstacle cells, and two arrays of single-precision floating point variables, with 19 values for each cell in the grid. During a loop over all cells in the current grid, each cell collects the neighboring DFs according to Eq. (4.1) and Eq. (4.5) for adjacent fluid cells and obstacle cells, respectively. The density and the velocity are computed and used to calculate the equilibrium DFs. These are weighted with the streamed DFs and written into the other grid, continuing with the next cell in the grid. Subsequent time steps alternate in streaming and colliding DFs from one grid array to the other.

In contrast with the standard finite-difference NS solver, the implementation of Eq. (4.5) for DFs of cells is much simpler but it demands more memory. A typical NS solver usually requires 7 floating point values at each grid point (pressure, three velocity components, and three temporary variables). It might need higher resolutions to resolve obstacles with the same accuracy in some cases. The implementation of a more sophisticated LB

implementation with grid compression [49], the memory requirement can be reduced to almost one half of usual requirements. Furthermore, the use of an adaptive time step size is common for a NS solver while the time step size of LBM is, by default, fixed to 1. Since the maximum lattice velocity may not exceed 1/3 for the LBM to remain stable, it might need several time steps to advance to the same degree as a NS solver would reach in one single step. However, each of these time steps usually requires a significantly smaller amount of work, as the LBM can be computed very efficiently on modern CPUs. Moreover, it does not require additional global computation such as the pressure correction step as done in the PLSM.

4.2.2 Stability

To simulate turbulent flows with the LBM, the basic algorithm has to be extended since its stability is limited once the relaxation parameter, τ, approaches 1/2 (which is equivalent to the case where w is close to 2). Here, the Smagorinsky sub-grid model as used in [37,73] will be applied. Its primary use is to stabilize the simulation, instead of relying on its ability to accurately model subgrid scale vortices in the simulation. As compared with the small slowdown due to the increased complexity of the collision step, this usually results in a large improvement on efficiency, as simulations would otherwise require considerably finer grid resolutions. The sub-grid turbulence model calculates the local stress tensor in the LBM [58]. This tensor computation is relatively easy for the LBM, since each cell already contains the information about derivatives of hydrodynamic variables in each DF.

The magnitude of the stress tensor is then used in each cell to modify the relaxation time according to the eddy viscosity. To calculate the modified relaxation time, the

Smagorinsky constant C is used. In our simulations, C is set to 0.03. Values in this range are commonly used in LB simulation, and were shown to yield good modeling of sub-grid vortices [81]. The turbulence model is integrated into the basic algorithm as described in Sec. 4.2.1 by adding the calculation of the modified relaxation time after the streaming step, and using this value in the normal collision step.

The modified relaxation time τ_s can be calculated by the following steps. First, the non-equilibrium stress tensor at each cell is computed as

$$\prod_{\alpha,\beta} = \sum_{i=1}^{19} e_{i_\alpha} e_{i_\beta} (f_i - f_i^{eq}),$$

where α and β run over the three spatial dimensions, while i is the index of the respective velocity vector for the D3Q19 model. The intensity of the local stress tensor S is then computed as

$$S = \frac{1}{6C^2} \left(\sqrt{\nu^2 + 18C^2 \sqrt{\prod_{\alpha,\beta} \prod_{\alpha,\beta}}} - \nu \right), \tag{4.6}$$

and the modified relaxation time is given by

$$\tau_s = 3(\nu + C^2 S) + \frac{1}{2}. \tag{4.7}$$

We see from Eq. (4.6) that S always has a positive value so that local viscosity increases depending on the size of the stress tensor calculated from the non-equilibrium part of the DFs of the cell to be relaxed. This effectively removes instability due to a small value of τ.

4.2.3 Parametrization

The conversion of dimensional quantities, denoted by primed symbols, into dimensionless quantities used in the LBM will be described in this subsection. Let S' be the length of one side of the domain that is quantized into r cells. The cell size used in the LBM can then be computed as $\triangle x' = S'/r$. Then, the actual value of kinematic viscosity is $\nu'[\frac{m^2}{s}]$, domain size $S'[m]$, a desired grid resolution r, and a gravitational force $g'[\frac{m}{s^2}]$, the corresponding lattice quantities will be computed as described below.

The dimensional timestep $\triangle t'$ is computed by limiting the compressibility due to the gravitational force. Here, we use $g_c = 0.005$ to keep the compressibility below half a percent. Thus,

$$\triangle t' = \sqrt{\frac{g_c \cdot \triangle x'}{|g'|}} \tag{4.8}$$

yields a time step ensuring that the force exerted on each cell due to gravitational acceleration has an effect less than the factor of compression, g_c. Given $\triangle x'$ and $\triangle t'$, the dimensionless lattice viscosity ν and relaxation time w are computed as

$$\nu = \nu' \frac{\triangle t'}{\triangle x'^2}, \tag{4.9}$$

$$\tau = 3\nu + 1/2, \quad w = \frac{1}{\tau}. \tag{4.10}$$

Likewise, the lattice acceleration due to gravity, g, is calculated as

$$g = g' \frac{\triangle t'^2}{\triangle x'}.$$

In conclusion, a valid parametrization for the LBM-based fluid simulation is given by the physical scale, the desired kinematic viscosity and the compressibility factor. For a given grid resolution, the values of τ and $\triangle t$ can be calculated accordingly. Note that, when the grid resolution is small in combination with a low viscosity, the resulting value of τ will be close to 1/2. With the turbulence model in Sec. 4.2.2, the simulation will remain stable, but effectively increase the viscosity to a value that can be handled by the chosen grid resolution.

4.2.4 Derivation

The Boltzmann equation will be studied, and the relationship between NSEs, the Boltzmann equation, and LBM will be investigated.

4.2.4.1 The Boltzmann Equation

With external force $\mathbf{G}$, the Boltzmann equation can be written as

$$\frac{\partial f}{\partial t} + \xi \cdot \frac{\partial f}{\partial \mathbf{x}} + \mathbf{G} \cdot \frac{\partial f}{\partial \xi} = \Omega(f), \tag{4.11}$$

where function f gives the amount of particles traveling with a given speed, volume, time and position. The left-hand-side of Eq. 4.11 describes the overall motion of molecules with microscopic velocity ξ through the force field that is given by $\mathbf{G}$ at $\mathbf{x}$, while the right-hand-side models the interaction of molecules with the collision operator Ω. It is an integral equation that includes the differential collision cross section for two particles

which can be calculated geometrically by approximating molecules with rigid spheres for the collision.

Due to the complicated nature of collision operator Ω, it is often replaced by a simpler expression that preserves the collision invariant. The standard model is the BGK approximation [3], which can be represented by

$$\Omega_{BGK}(f) = \frac{f^e - f}{\tau}, \tag{4.12}$$

where f^e is of the Maxwellian distribution representing the local equilibrium that is parameterized by the conserved quantities such as density ρ, speed ξ and temperature T. Each collision changes the distribution function, f, proportional to the departure from the local equilibrium f^e, where the amount of correction is modified by relaxation time τ. In general, the collision time is dependent on properties of the gas and its current state. However, for the BGK approximation, it is simplified and represented as a single value.

The local equilibrium is reached when $\Omega(f^e, f^e)$ vanishes. With this property, it can be shown that f is collision invariant, and it does not change under the effect of a collision. The density ρ, momentum ξ_a, and energy E are the Lagrangian parameters. For a normalized particle of unit mass, these quantities can be computed as

$$\int f d\xi = \rho, \quad \int f u_a d\xi = \rho \xi_a, \quad \text{and} \quad \int \mathrm{f} \frac{\mathrm{u}^2}{2} \mathrm{d}\xi = \rho \mathrm{E}. \tag{4.13}$$

The macroscopic flow speed ξ_a, density ρ, and fluid temperature T parameterize the Maxwell distribution. In the 3D space, it can be written as

$$f^M = \rho \left(\frac{m^2}{2\pi RT} \right)^{3/2} e^{\frac{-(\varepsilon - u)^2 m^2}{2RT}}, \tag{4.14}$$

where R is the Boltzmann constant and m is the mass of a particle.

4.2.4.2 Chapman-Enskog Expansion

NSEs can be derived from the Boltzmann equation by a multi-scale analysis called the Chapman-Enskog expansion. It relies on the Knudsen number, $Kn = \lambda / L_C$, which is the ratio between the mean free path length, λ, and the characteristic shortest scale, L_C, of the macroscopic system of consideration. The Knudsen number should be much smaller than one for the fluid to be treated as a continuous system. To derive NSEs from the Boltzmann equation, the latter is split according to a hierarchy of different scales of space and time variables. It is based on the expansion parameter, ε, for which the Knudsen number Kn will be used. Usually, the expansion is truncated after terms of the second order. In the following, the derivation of Euler equations will be shown, which also illustrates subsequent steps necessary to derive full NSEs.

The representation $t = \varepsilon t_1 + \varepsilon^2 t_2$ is chosen for the time variable. Variable t represents the scale of the fast local relaxation in a fluid by collision. Sound waves are of scale t_1, which is considerably slower than local relaxations. But they are faster than diffusion processes which take place in time scale t_2. On the other hand, only one spatial expansion is considered, giving the following expansion of the first order, $\mathbf{x} = \varepsilon \mathbf{x}_1$. This is due to

the fact that advection and diffusion are both considered in similar spatial scales $\mathbf{x}_1$. The differential operators are represented in the same way as

$$\frac{\partial}{\partial x_a} = \varepsilon \frac{\partial}{\partial x_a}, \text{ and } \frac{\partial}{\partial t} = \varepsilon \frac{\partial}{\partial t} + \varepsilon^2 \frac{\partial}{\partial t}. \tag{4.15}$$

For consistent expansion, the second order terms in space are needed. The moment equations of f are directly expanded to the following form:

$$f = \sum_{n=0}^{\infty} \varepsilon^n f^n. \tag{4.16}$$

It is assumed that the time dependence of f is only caused by variables ρ, $\mathbf{u}$ and T. The expansion of Eq. (4.11) in both space and time up to the second order yields

$$\varepsilon \frac{\partial f}{\partial t_1} + \varepsilon^2 \frac{\partial f}{\partial t_2} + \varepsilon u_a \cdot \frac{\partial f}{\partial x_a} + \frac{1}{2} \varepsilon^2 u_a u_b \frac{\partial^2 f}{\partial x_a \partial x_b} = \Omega(f^0) + \varepsilon \frac{\partial \Omega(f^1)}{\partial f}. \tag{4.17}$$

Note that f^0 is of the Maxwell distribution and $\Omega f(f^0)$ is zero due to the definition of the BGK collision approximation in Eq. (4.12). The three scales from $O(\varepsilon^0)$ to $O(\varepsilon^2)$ can be distinguished in Eq. (4.17) and handled separately.

In the following, subsequent expansions of conservation equations will be performed. Using the second-order Knudsen number expansion of the mass m, the first order terms of Eq. (4.17) yields

$$\frac{\partial \rho}{\partial t_1} + \frac{\rho \partial u_a}{\partial x_{1a}} = 0, \tag{4.18}$$

and

$$\frac{\rho\partial u_a}{\partial t_1} + \frac{\partial \int u_a u_b f^0 du}{\partial x_{1b}} = 0. \tag{4.19}$$

When the integral of the second equation is evaluated analytically, it can be replaced by $\rho u_a u_b + \rho T \delta_{ab}$ which leads to

$$\frac{\rho\partial u_a}{\partial t_1} + \frac{\partial \rho u_a u_b}{\partial x_{1b}} + \frac{\partial \rho T \delta_{ab}}{\partial x_{1b}} = 0, \tag{4.20}$$

which is the Euler equation for inviscid flows without dissipation.

Finally, to derive NSEs, the second-order equations have to be considered. For these, both equilibrium and non-equilibrium levels have to be handled in the expansion. Still, by setting the first order conservation terms to zero and restoring the continuous form of these equations, NSEs as shown in Eq. (2.3) can be derived. This is possible since the higher-order term of order (*e.g.*, $O(u^3)$) can be neglected under the assumption of small Mach numbers in the expansion. The full derivation with these additional steps that are similar to the expansion step as shown above, is given in, e.g., [24, 79].

4.2.4.3 Derivation of Lattice Boltzmann Equation

In this subsection, we will derive the lattice Boltzmann equation from the continuous Boltzmann equation [27]. Although the lattice Boltzmann equation was historically derived from the lattice gas cellular automata, the method described here allows the derivation of the lattice Boltzmann equation from an arbitrary kinetic equation. The following abbreviations will be used:

$$f(\mathbf{x}, \xi, t) = f(t),$$

$$f(\mathbf{x}+\xi a, \xi, t+a) = f(t+a).$$

The same abbreviations hold for g, which denotes an equilibrium distribution function as explained in Sec. 4.2.1. As a starting point, the Boltzmann equation with BGK collision approximation will be used as

$$\frac{\partial f(t)}{\partial t} + \xi \frac{\partial f(t)}{\partial \mathbf{x}} = -\frac{1}{\lambda}\left[f(t) - g(t)\right], \tag{4.21}$$

where f is the particle distribution function at time t and position $\mathbf{x}$ with microscopic velocity ξ. Parameter $1/\lambda = An$ is the relaxation time for collision, which is calculated from the number of particles n and the proportional coefficient A. Here, the collision term is linearized according to Eq. (4.12) for simplicity yet without losing generality. Furthermore, g is of the Maxwell distribution f^M from Eq. (4.14).

The hydrodynamic properties of the fluid, such as density ρ, velocity $\mathbf{u}$, and the temperature T, can be calculated with the moments of function f. Besides, energy γ from the energy density $\rho\gamma$ can be used to determine the temperature of the fluid. They are given below:

$$\rho = \int f(\mathbf{x}, \xi, t) d\xi, \tag{4.22}$$

$$\rho\mathbf{u} = \int \xi f(\mathbf{x}, \xi, t) d\xi, \tag{4.23}$$

$$\rho\gamma = \int \frac{1}{2}(\xi - \mathbf{u}) f(\mathbf{x}, \xi, t) d\xi. \tag{4.24}$$

Although the equilibrium distribution function, g, is written as a function of time and velocity, it is calculated by these hydrodynamic moments. Hence, these values have to be correctly approximated after discretization.

Time discretization

Eq. (4.21) can be formulated as an ordinary differential equation (ODE) as

$$\frac{Df}{Dt} + \frac{1}{\lambda} f = \frac{1}{\lambda} g, \tag{4.25}$$

where

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \xi \frac{\partial}{\partial \mathbf{x}}$$

is the time derivative along the microscopic velocity. If δt is small and g is a smooth function, then Eq. (4.25) can be simplified as

$$f(t + \delta_t) - f(t) = -\frac{\delta_t}{\lambda}(f(t) - g(t)), \tag{4.26}$$

where relaxation time $\frac{\delta_t}{\lambda}$ is usually written as $\frac{1}{\tau}$. This formula is already similar to Eq. (4.4) above.

Approximation of equilibrium distribution

The Maxwell distribution used as the equilibrium distribution function g is given by Eq. (4.14). For a particle with mass 1 and D dimensions, it is represented as

$$g(\mathbf{u}) = \frac{\rho}{(2\pi RT)^{D/2}} e^{-\frac{(\varepsilon-\mathbf{u})^2}{2RT}}. \tag{4.27}$$

Function $g(\mathbf{u})$ will be expanded in $\mathbf{u}$ up to the second order, which is a valid approximation for small velocities and low Mach numbers. The local equilibrium distribution found is

$$f^{eq} = \frac{\rho}{(2\pi RT)^{D/2}} e^{-\frac{\varepsilon^2}{2RT}} \left(1 + \frac{\xi \cdot \mathbf{u}}{RT} + \frac{(\xi \cdot \mathbf{u})^2}{2(RT)^2} - \frac{\mathbf{u}^2}{2RT}\right). \tag{4.28}$$

It is derived by expanding the quadratic form in the exponent of e from Eq. (4.27) and expanding the resulting equation using Taylor's series.

Discretization of velocities

For simplicity, the D2Q9 model will be considered here. As given in Eq. (4.22), the moment integrals over the whole velocity space should be evaluated. As the velocity is not yet discretized, these run from $-\infty$ to $+\infty$ in both x and y directions for a 2D model. The moments of particle distribution functions are important for consistency with NSEs. Another important property to be retained by discretization is isotropy which is probably the most important properties of Navier-Stokes symmetries. Thus, the lattice should be invariant to rotations, which can be checked by examining isotropy tensors in [79]. For the LBM derivation, the moments are directly used as a constraint for the numerical integration method. For models that include temperature, the integration of moments

up to the second order should also be included. An isothermal model will be used here where only the first moment of the velocity is required. The moments of Eq. (4.28) in 2D can be written as

$$I = \int \psi(\xi) f^{(0)} d\xi \quad \text{with} \quad \psi(\xi) = \xi_x^m \xi_y^n, \tag{4.29}$$

where ψ is the moment function that contains powers of the velocity components. After restructuring the equation, moments of up to the third order will occur in the equation - one from the velocity moment, and two from the $(\xi \cdot u)^2$ term. For numerical treatment, Eq. (4.29) can be written as

$$I = \frac{\rho}{(2\pi RT)^{D/2}} \int \psi(\xi) e^{-\frac{\xi^2}{2RT}} \left(1 + \frac{\xi \cdot \mathbf{u}}{RT} + \frac{(\xi \cdot \mathbf{u})^2}{2(RT)^2} - \frac{\mathbf{u}^2}{2RT}\right) d\xi. \tag{4.30}$$

The next step for the derivation of the LBM is to numerically integrate these moments with

$$\int f(x) W(x) dx = \sum_{j=1}^{N} w_j f(x_j),$$

where $W(x)$ is the weighting function (e^{-x^2}, in our case), and $f(x)$ is a polynomial in x (*e.g.*, $f(\zeta_x) = \zeta_x^m$). To numerically integrate functions such as $e^{-\zeta^2}$, the commonly used Gauss-Hermite quadrature can be applied, which is correct for polynomials in W up to the order of $2N - 1$. The order of the quadrature has to be chosen according to the order of the moment polynomial ψ. Although the model is isothermal, the energy due to the temperature should be kept constant. Hence, there is no additional level of

freedom for the temperature. However, it has to be taken into account for the moment integration. The Gauss-Hermite quadrature of the third order (N = 3) is thus required

$$I_i^m = \sum_{j=1}^{3} w_j {\zeta_j}^m. \tag{4.31}$$

The values of ζ and w are given by the Gauss-Hermite quadrature as $\zeta_1 = -\sqrt{3/2}$, $\zeta_2 = 0$, $\zeta_3 = +\sqrt{3/2}$, $w_1 = \sqrt{\pi}/6$, $w_2 = 2\sqrt{\pi}/3$ and $w_3 = \sqrt{\pi}/6$. The moment function can be shortened as

$$I = \frac{\rho}{\pi} \sum_{i=1}^{3} \sum_{j=1}^{3} w_i w_j \psi(\zeta_{i,j}) \left(1 + \frac{\xi \cdot \mathbf{u}}{RT} + \frac{(\xi \cdot \mathbf{u})^2}{2(RT)^2} - \frac{\mathbf{u}^2}{2RT} \right), \tag{4.32}$$

where $\zeta_{i,j}$ is the vector given by the quadrature as $\zeta_{i,j} = \sqrt{2RT}(\zeta_i, \zeta_j)^T$. As the two sums run over three values for i and j each, there are a total of nine possible values for $\zeta_{i,j}$ and $w_i w_j$. For these, a new single index will be introduced. Furthermore, a number of substitutions can be made. Since an isothermal model is used, temperature T can be replaced by a constant $c = \sqrt{2RT}\sqrt{3/2} = \sqrt{3RT}$. The speed of sound $c_s = 1/\sqrt{3}$ in the model yields $c_s^2 = c^2/3 = RT$. The weights w, divided by π are

$$\begin{aligned} w_0 &= w_2 w_2 = 4/9 \\ w_{1..4} &= w_1 w_2, \;\; w_2 w_1, \;\; w_3 w_2, \;\; w_2 w_3 = 1/9 \\ w_{5..8} &= w_1 w_3, \;\; w_3 w_1, \;\; w_1 w_1, \;\; w_3 w_3 = 1/36 \end{aligned} \tag{4.33}$$

Each component of vectors $\zeta_{i,j}$ is either 0 or $\sqrt{2RT}\sqrt{3/2} = \sqrt{3RT} = c$:

$$\begin{aligned} e_0 &= \zeta_{1,1} = (0,0)^T \\ e_{1..4} &= \zeta_{1,2}, \zeta_{2,1}, \zeta_{3,2}, \zeta_{2,3} = (1,0)^T c, (0,1)^T c \\ e_{5..8} &= \zeta_{1,3}, \zeta_{3,1}, \zeta_{1,1}, \zeta_{3,3} = (1,1)^T c \end{aligned} \tag{4.34}$$

With these discrete velocities, Eq. (4.32) can be written as:

$$I = \sum_{\alpha=1}^{9} W_\alpha \psi(e_\alpha) f_\alpha^{eq}, \tag{4.35}$$

where $W_\alpha = 2\pi RT e^{\frac{\xi^2}{2RT}}$. This yields the equilibrium distribution function as given in Eq. (4.3) for each of the nine velocities:

$$f_\alpha^{eq} = w_\alpha \rho \left(1 + \frac{3\mathbf{e}\cdot\mathbf{u}}{c^2} + \frac{9(\mathbf{e}\cdot\mathbf{u})^2}{2c^4} - \frac{3\mathbf{u}^2}{2c^2}\right). \tag{4.36}$$

Note that the lattice velocity vectors were given by the chosen Gauss-Hermite quadrature. The configuration of the lattice is likewise obtained from these velocities. It is possible to discretize velocities and lattice configuration differently, as shown in [27], and [5]. Other LBM, such as the D3Q27 model, can be derived in the same way. For the more often used 3D model such as D3Q19, it is difficult to apply this method directly. Problems arise from the more irregular arrangement of velocity vectors that cannot be easily formulated as a quadrature term. For these models, the *ansatz method* can be used [79]. For a given kinetic equation, such as the one in Eq. (4.21) with an equilibrium distribution or the one in Eq. (4.28), velocity weights for a specific lattice can be calculated.

Multi-scale analysis yields constraints for moments of f that can be used to compute the desired coefficients.

4.3 Lattice Boltzmann Simulations with a Free Surface

Simulation of free surfaces demands a distinction between regions that contain fluid and regions that contain only gas. This is done by marking cells that contain no fluid as empty in the flag field. As with obstacle cells, the DFs of these cells are completely ignored in the simulation. However, in contrast to boundary cells, the fluid might move into this empty area at some point in the simulation. To track the fluid motion, another cell type is introduced, which is called the interface cell. These cells form a closed layer, as shown in Fig. 4.4 between fluid and empty cells.

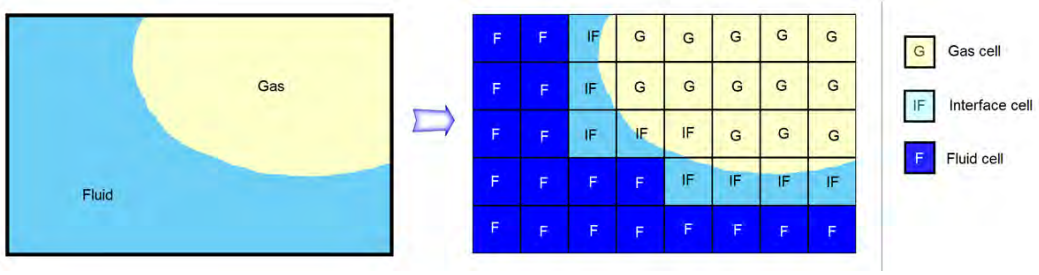


Figure 4.4: Different cell types required for visible free surface.

Here, the main simulation task is to track the free surface. It consists of three steps: 1) computation of the interface movement; 2) the boundary conditions at the fluid interface, and 3) re-initialization of cell types. In this section, these three steps are executed for an interface cell (instead of the standard streaming and collision step.) An overview of the procedure is given in Fig. 4.5.

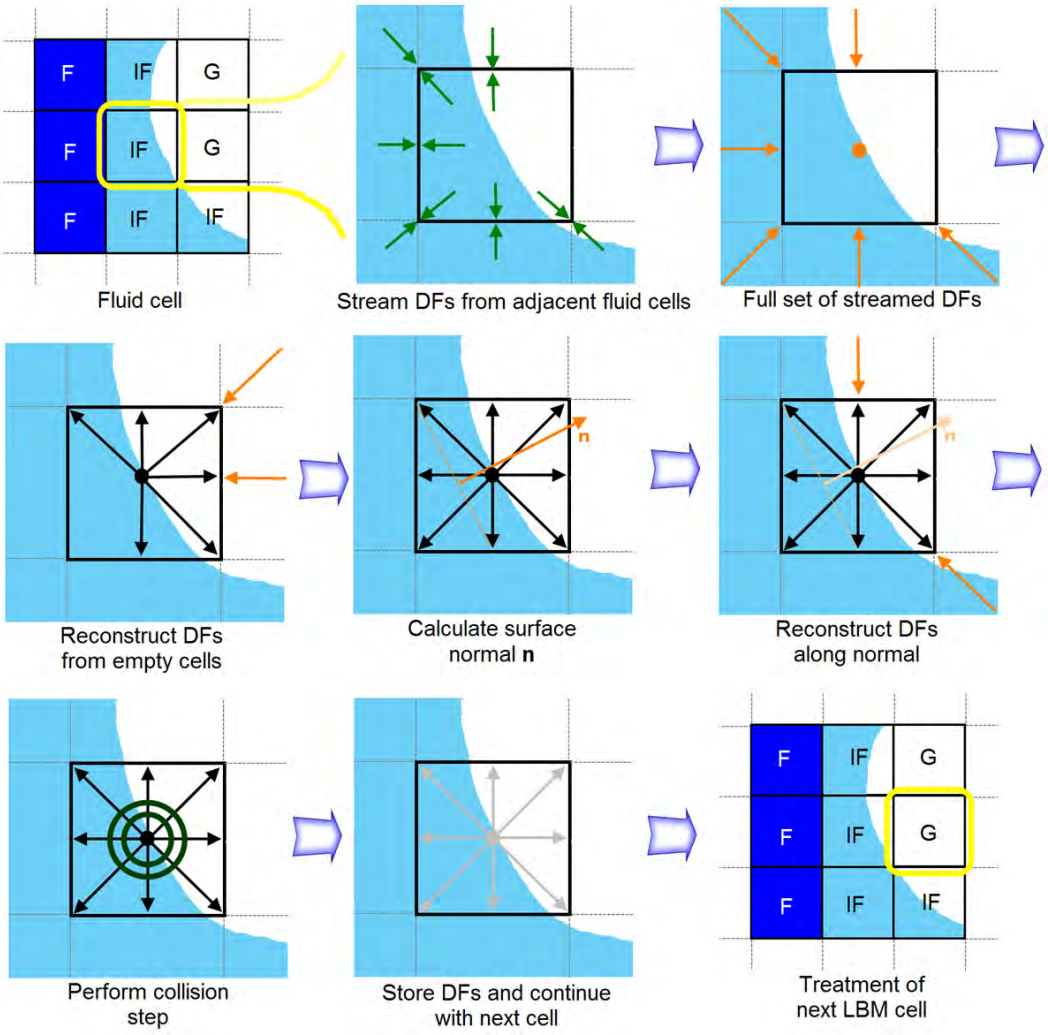


Figure 4.5: Illustration of steps to be executed for an interface cell.

4.3.1 Interface Movement

The movement of the fluid interface is tracked by the calculation of the mass contained in each cell. This requires two additional values to be stored at each cell, mass m and fluid fraction ϵ. The fluid fraction is computed by the cell mass and density as

$$\epsilon = m/\rho. \tag{4.37}$$

Being similar to the volume-of-fluid (VOF) method, interface motion is tracked by computing fluxes between cells. However, as DFs correspond to a certain number of particles, the change of mass is directly computed from values that are streamed between two adjacent cells for each of the directions in the model. For an interface cell and a fluid cell at $(\mathbf{x} + \triangle t\mathbf{e}_i)$ is given by

$$\triangle m_i(\mathbf{x}, t + \triangle t) = f_{\tilde{i}}(\mathbf{x} + \triangle t\mathbf{e}_i) - f_i(\mathbf{x}, t). \tag{4.38}$$

The first DF is the amount of fluid that enters a cell in the current time step, the second one is the amount that leaves the cell. The mass exchange for two interface cells should consider the area of the fluid interface between two cells. It is approximated by averaging the fluid fraction values of two cells. Thus, Eq. (4.38) becomes

$$\begin{aligned} \triangle m_i(\mathbf{x}, t + \triangle t) &= s_e \frac{\epsilon(\mathbf{x} + \triangle t\mathbf{e}_i, t) + \epsilon(\mathbf{x}, t)}{2}, \\ \text{with } \; s_e &= f_{\tilde{i}}(\mathbf{x} + \triangle t\mathbf{e}_i) - f_i(\mathbf{x}, t) \end{aligned} \tag{4.39}$$

Both equations are completely symmetric, as the amount of fluid leaving one cell has to enter the other one, and vice versa. Thus, we have

$$\triangle m_i(\mathbf{x}) = -\triangle m_{\tilde{i}}(\mathbf{x} + \triangle t\mathbf{e}_i).$$

For interface cells with neighboring fluid cells, the mass change has to conform to DF's exchange during streaming, as fluid cells do not require additional computations. Their fluid fraction is always equal to one and their mass equals their current density. The mass

change values for all directions are added to the current mass for interface cells, resulting in mass in the next time step as

$$m(\mathbf{x}, t + \triangle t) = m(\mathbf{x}, t) + \sum_{i=1}^{19} \triangle m_i(\mathbf{x}, t + \triangle t). \tag{4.40}$$

4.3.2 Free Surface Boundary Conditions

As described above, DFs of empty cells are never accessed. However, interface cells always have empty cell neighbors. Thus, during the streaming step, only DFs from fluid cells or other interface cells are streamed normally while DFs that would be read from empty cells need to be reconstructed with corresponding boundary conditions at the free surface. These boundary conditions do not require additional constructs such as ghost layers around the interface. Thus, they can be treated locally at each cell. An atmospheric pressure of $\rho_A = 1$ is used, which is also the reference density and pressure of the fluid. Moreover, it is assumed that the viscosity of the fluid is significantly lower than that of the gas phase while having a higher density. Hence, the gas follows the fluid motion at the interface. In terms of DFs this means that, if there is an empty cell at $(\mathbf{x} + \triangle t\mathbf{e}_i)$, then we have

$$f_i(\mathbf{x}, t + \triangle t) = f_i^{eq}(\rho_A, \mathbf{u}) + f_{\bar{i}}^{eq}(\rho_A, \mathbf{u}) - f_i(\mathbf{x}, t), \tag{4.41}$$

where $\mathbf{u}$ is the velocity of the cell at position $\mathbf{x}$ and time t according to Eq. (4.2).

The pressure of the atmosphere onto the fluid interface is introduced by ρ_A for the density of the equilibrium DFs. Applying Eq. (4.41) to all directions with empty neighbor cells would result in a full set of DFs for interface cells. However, to balance forces on

each side of the interface, DFs coming from the direction of the interface normal are also reconstructed. Thus, if DF f_i would be streamed from an empty cell, or if

$$\mathbf{n} \cdot \mathbf{e}_{\bar{i}} > 0 \;\text{ with }\; \mathbf{n} = \frac{1}{2} \begin{pmatrix} \epsilon(x_{j-1,k,l}) - \epsilon(x_{j+1,k,l}) \\ \epsilon(x_{j,k-1,l}) - \epsilon(x_{j,k+1,l}) \\ \epsilon(x_{j,k,l-1}) - \epsilon(x_{j,k,l+1}) \end{pmatrix} \tag{4.42}$$

holds, f_i can be reconstructed using Eq. (4.41). Here $\mathbf{x}_{j,k,l}$ simply denotes the position of the cell at plane l, row k and column j in the array. Hence, the normal is approximated with central differences of the fluid fraction in each spatial direction. Now, all DFs for the interface cell are valid so that the standard collision is performed using Eq. (4.36). The density that was calculated during collision is now used to check whether the interface cell filled or emptied during this time step as

$$\begin{aligned} &m(\mathbf{x}, t + \triangle t) > (1 + \kappa)\rho(\mathbf{x}, t + \triangle t) \rightarrow \text{cell filled}, \\ &m(\mathbf{x}, t + \triangle t) < (0 - \kappa)\rho(\mathbf{x}, t + \triangle t) \rightarrow \text{cell emptied}. \end{aligned} \tag{4.43}$$

An additional offset, $\kappa = 10^{-3}$, is used (instead of 0 or 1) for emptying and filling thresholds to prevent the new surrounding interface cells from being re-converted in the following LB step. Instead of immediately converting emptied or filled cells themselves, their positions are stored in a list (one for emptying, and the other for filling cells), and the conversion is done when the main loop over all cells has been completed.

4.3.3 Flag Re-initialization

This step takes place when all cells have been updated to ensure two properties. First, once the filled and emptied interface cells have been converted into their respective types, the layer of interface cells has to be closed again. Next, the conservation of mass has to be maintained during the conversion. While empty and fluid cells have a mass of exactly zero and one, respectively, interface cells that have filled or emptied according to Eq. (4.43) usually have an excess mass on conversion. This excess mass, which can be positive or negative, needs to be distributed to neighboring interface cells.

All neighboring empty cells are converted to interface cells. For each of them, the average density ρ^{avg} and velocity $\mathbf{v}^{avg}$ of the surrounding fluid and interface cells are computed. The DFs of empty cells are then initialized with the equilibrium $f_i^{eq}(\rho^{avg}, \mathbf{v}^{avg})$. Here, it is necessary to remove any interface cells that are needed as boundary for a filled cell from the list of emptied interface cells. During the same pass, the flag of the filled cells is changed to fluid. Likewise, for all emptied cells, the surrounding fluid cells are converted to interface cells, simply taking the former fluid cell's DFs at each corresponding new interface cell. Furthermore, emptied interface cells are now marked as being empty. In a second pass, the excess mass m^{ex} is distributed among the surrounding interface cells for each emptied and filled cell. m^{ex} is equal to the mass m of the emptied cell (according to Eq. (4.43) this value is negative), and calculated as $(m - \rho)$ for filled cells.

Like the mass values larger than the density in filled ones, negative mass values in emptied interface cells mean that the fluid interface moved beyond the current cell during the last time step. To compensate this, the mass is not distributed evenly among the

surrounding interface cells, but weighted according to the direction of the interface normal $\mathbf{n}$ (which is computed as in Eq. (4.42)):

$$m(\mathbf{x} + \triangle t\mathbf{e}_i) = m(\mathbf{x} + \triangle t\mathbf{e}_i) + m^{ex}(\eta/\eta_{total}), \tag{4.44}$$

where η_{total} is the sum of all weights η_i, and each of which is computed as

$$\eta_i = \begin{cases} \mathbf{n} \cdot \mathbf{e}_i & \text{if } n \cdot e_i > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for filled cells, and}$$

$$\eta_i = \begin{cases} -\mathbf{n} \cdot \mathbf{e}_i & \text{if } \mathbf{n} \cdot \mathbf{e}_i < 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for emptied cells.} \tag{4.45}$$

As the mass of adjacent interface cells changes, the fluid fraction also changes accordingly. For the computational steps described so far, it is important that they yield the same results independent of the order in which the filled and emptied cells are converted. Thus, interpolation for empty cells may only interpolate values from cells that are not new interface cells themselves. Once the cell conversion is complete, the current grid is valid, and is advanced by starting the main loop over all cells again.

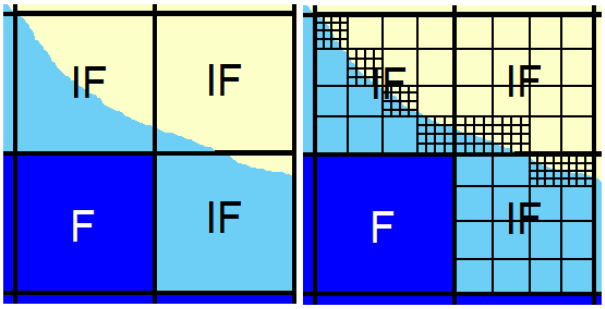


Figure 4.6: Multi-resolution density calculation up to 3rd level, where the left figure shows the original profile of cells and the right figure shows the multi-resolution density calculation. Symbols F, IF, and G denote fluid, interface, and gas cells, respectively.

4.4 Hybrid Lattice Boltzmann Method (HLBM) for Free Surface Fluid Simulation

As explained in Sec. 3.2, liquid simulation using the LSM enables smooth surface representation but it suffers from a huge computational cost because of the global pressure correction step to solve the Poisson equation for the entire computational domain. On

step 1	If current cell is F, $\rho = 1$ Else if current cell is G, $\rho = 0$ Else if current cell is IF, split the cell into 4^3 sub-cells and check whether the sub-cell is F, G, or, IF
step 2	If current sub-cell is F, $\rho = 1/4^3$ Else if current sub-cell is G, $\rho = 0$ Else if current sub-cell is IF, split the sub-cell into 4^3 sub-sub-cells and check whether the sub-sub-cell is F, G, or, IF
step 3	If current sub-sub-cell is F, $\rho = 1/4^6$ Else if current sub-sub-cell is G, $\rho = 0$ Else if current sub-sub-cell is IF, $\rho = 1/2 \times 1/4^6$
step 4	Find the sum of ρ for entire cell

Table 4.1: Multi-resolution density calculation up to 3rd level for the PLSM part. Here, we use F, IF, and G to denote fluid, interface, and gas cells, respectively, and ρ is density of the current (sub)cell.

the other hand, liquid simulation using the LBM has an efficient basic algorithm and preserves mass as discussed in Sec. 4.2. However, it suffers from a small time step restriction and a high memory requirement [69]. In this section, we propose a hybrid algorithm that integrates the LBM with the PLSM for more realistic and faster liquid simulation.

To combine the LBM with the PLSM, we first need to find the macroscopic velocity field to advect the level set function and particles. The macroscopic velocity of each cell can be calculated using Eq. (4.2) and the distance from the center of each cell to the fluid interface can be calculated using the marching cube algorithm [40]. Thus, the level set function can be advected using the macroscopic velocity field. And the semi-Lagrangian advection scheme [60] is used for the advection method. However, the macroscopic velocities of a lattice cell can be calculated only for fluid and interface cells. In other words, the velocities of a gas cell are always zero using Eq. (4.2) because the distribution functions at $\mathbf{x}$, $f_i(\mathbf{x})$, are all zero. As the level set functions have to be

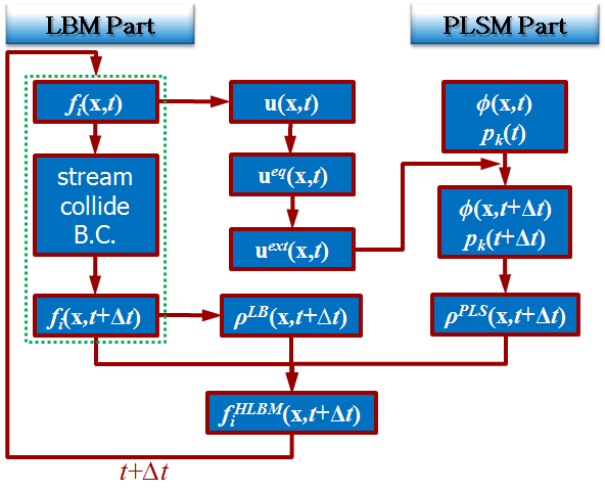


Figure 4.7: Overview of the hybrid lattice Boltzmann method, where the dotted box (green) represents the overview of the LBM only.

defined in both the gas and fluid regions, velocities from the fluid have to be extrapolated into the gas region with the fast marching method as described in Sec. 3.4.

The hybrid lattice Boltzmann method (HLBM) is described below.

- Step 1: We run the LBM solver, where the streaming and the collision steps are performed. The obstacle and free surface boundary conditions are also applied, and the distribution functions of the next time step, $f_i(\mathbf{x}, t + \Delta t)$, for each lattice are calculated.

- Step 2: Macroscopic velocities for the current time step, $\mathbf{u}(\mathbf{x}, t)$, are calculated using Eq. (4.2).

- Step 3: The velocity field, $\mathbf{u}(\mathbf{x}, t)$, is extrapolated to the gas region because the LBM does not have velocities for the gas region. This extrapolated velocity field, $\mathbf{u}_{ext}(\mathbf{x}, t)$, is required for the advection of the PLSM because the semi-Lagrangian advection scheme needs velocities of the gas region along with velocities of the liquid region.

- Step 4: The level set function, $\phi(\mathbf{x}, t)$, and particles, $\mathbf{p}_k(t)$, are advected by the extrapolated velocity field, which is calculated in the previous step. The level set function are advected using the semi-Lagrangian advection scheme and particles are advected using the 3^{rd} order TVD-RK method.

- Step 5: Advected particles correct errors in the level set function in the PLSM.

- Step 6: Now, we have two different density fields obtained from the LBM and the PLSM solvers. For the LBM part, we can calculate the density of each cell, ρ_{LB}, by

the sum of distribution functions using Eq. (4.2). For the PLSM part, the density of each cell, ρ_{PLS}, is calculated using multi-resolution density calculation scheme up to 3^{rd} level as described in Table 4.1. Fig. 4.6 also shows 2D example of the density calculation method.

- Step 7: The density difference is between the LBM and the the PLSM solvers is added to distribution functions to correct errors of the LBM as

$$f_i^{HLBM}(\mathbf{x}, t+\Delta t) = \ f_i(\mathbf{x}, t+\Delta t) + \frac{\rho_{PLS}(\mathbf{x}, t+\Delta t) - \rho_{LB}(\mathbf{x}, t+\Delta t)}{M}, \quad (4.46)$$

where f_i and f_i^{HLBM} represent distribution functions from the LBM and HLBM, respectively. In Eq. (5.10), ρ_{LB} and ρ_{PLS} are densities calculated from the LBM and the PLSM, respectively. Note that M is equal to 19 for the $D3Q19$ model.

Fig. 4.7 shows a schematic overview of the HBLM algorithm, where $\mathbf{p}_k(t)$ represents the particle with id k at time t.

4.5 Experimental Results

For fluid animation, we first need to solve the Boltzmann equation numerically and this system is called the fluid solver. We use El'Beem, a free surface fluid solver using the LBM, for this purpose [68]. For the hybrid LBM (HLBM) solver, we added the PLSM modules such as level set functions, particles, velocity extrapolation, error correction, and so on. After running the HLBM, we get bobj (binary obj) files which can be imported to Blender [4], a free open source 3D content creation suite. Then, Blender can modify the

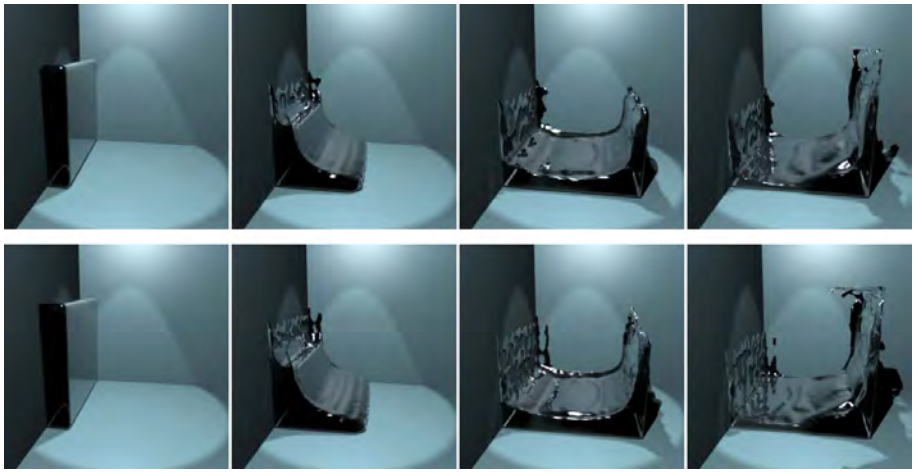


Figure 4.8: The broken dam simulation using the LBM (the top row) and HLBM (the bottom row) with 50^3 grids. The columns from the left to the right represent the 1^{st}, the 6^{th}, the 11^{th}, and the 16^{th} frames, respectively.

material property such as color and render the scene using its internal rendering engine. Simulation results in this chapter were obtained using a PC with 2.2GHz CPU and 4GB RAM.

Fig. 4.8 shows every 5 frames from the broken dam simulation using the LBM (the top row) and the HLBM (the bottom row). Both methods were run with 50^3 grids of size 0.1 m, 50 frames/sec, and the no-slip boundary condition. For each frame using the LBM, it took about 20 seconds for the fluid solver and surface generation and 80 seconds of rendering time with 600×600 image. For the simulation of the HLBM, we used 64 particles for each cell with $|\phi| < 6\triangle x$ as an initial condition. It took about 24 seconds for the fluid solver and surface generation and the rendering time took about 85 seconds for each frame using the internal raytracing renderer in Blender. Thus, the simulation time using the HLBM was about 20 % more than the LBM.

Fig. 4.9 shows every 5 frames from the water drop simulation using the LBM (the top row) and the HLBM (the bottom row). Both cases were run with 50^3 grids of size

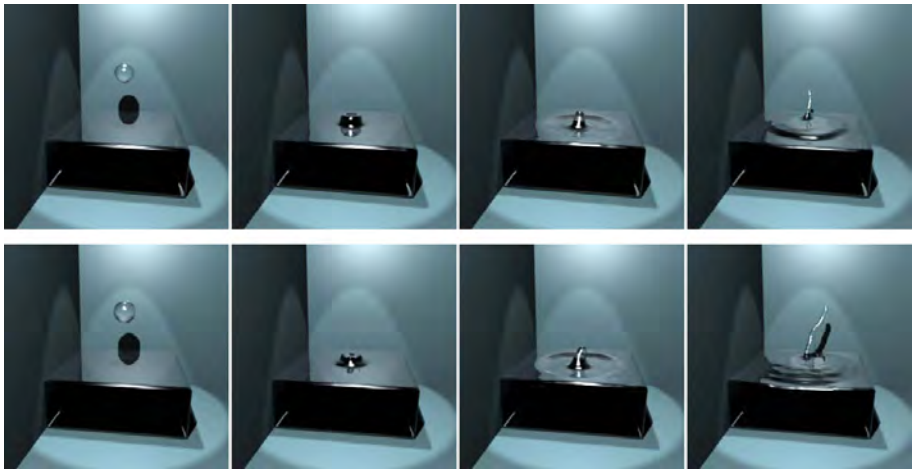


Figure 4.9: The water drop simulation using the LBM (the top row) and the HLBM (the bottom row) with 50^3 grids. The columns from the right to the left represent the 1^{st}, the 6^{th}, the 11^{th}, and the 16^{th} frames, respectively.

0.1 m, 50 frames/sec, and under the no-slip boundary condition. The LBM took about 22 seconds for the fluid solver and surface generation and 91 seconds for the rendering time for each frame. The HBLM took about 25 seconds for the fluid solver and surface generation and the rendering took about 95 seconds for each frame. The simulation time of the HLBM is 13.6 % higher than the LBM. As we see from Figs. 4.8 and 4.9, the visual quality of simulation results is improved using the HLBM. Especially, the HLBM enables a more splashy effect because the resolution of the fluid simulation is increased by adding particles.

Fig. 4.10 shows the 11^{th} frame of the broken dam simulation using the LBM (the upper left) and the HLBM (the upper right), and the 17^{th} frame of the water drop simulation using the LBM (the lower left) and HLBM (the lower right). For the broken dam simulation, we see a more splashy effect of the HLBM at the right wall of the domain. This splashy effect is also present at the water drop simulation, especially at the center

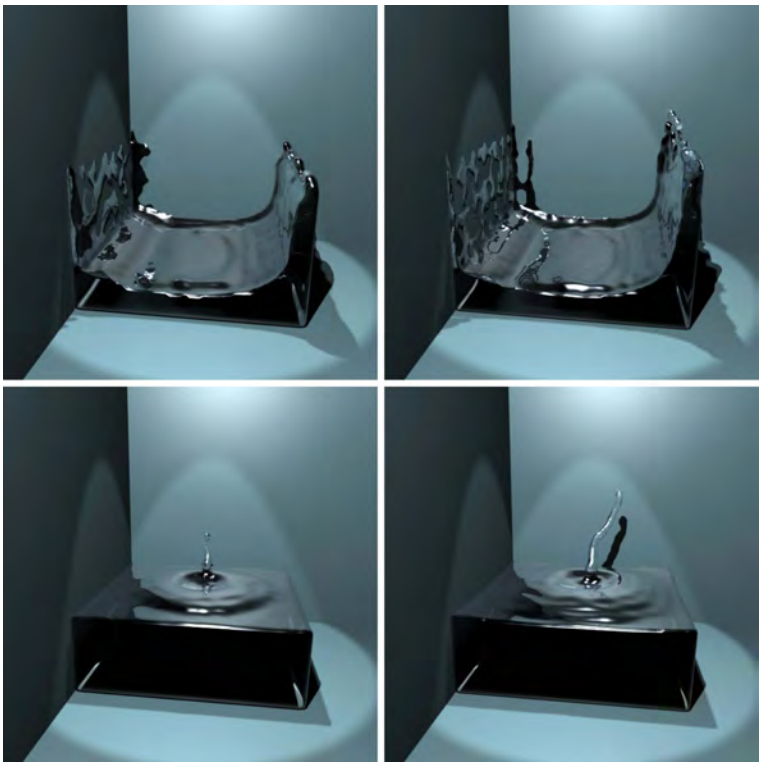


Figure 4.10: The 11^{th} frame of the broken dam simulation using the LBM (the upper left) and the HLBM (the upper right) and the 17^{th} frame of the water drop simulation using the LBM (the lower left) and the HLBM (the lower right).

of the image. Also, the liquid surface has finer detail with the HLBM than that with the LBM.

To quantify the visual improvement of the HLBM over the LBM, we first obtained the ground truth for the broken dam and the water drop examples using the LBM with the a very high resolution grid (*i.e.*, 150^3) and the same initial and boundary conditions. Then, we got simulation results using the LBM and the HLBM with grids of a lower resolution. They include: the LBM with the 50^3 grid, LBM with the 60^3 grid, the LBM with the 64^3 grid, and the HLBM with the 50^3 grid. Finally, we use MeshDev [52] to compare the geometric distances between the computed results and their ground truth values.

frame number			1	6	11	16	21	26	31	36	41
broken dam	LBM	50^3	1.3905	1.3983	1.3958	1.3853	1.8460	1.8010	1.8129	1.5937	1.3257
broken dam	LBM	60^3	0.8951	0.9615	1.3301	1.2225	1.5318	1.5781	1.8747	1.4580	1.3989
broken dam	LBM	64^3	0.8075	0.8977	1.1926	1.0811	1.4502	1.5414	1.6588	1.5370	1.2923
broken dam	HLBM	50^3	1.0595	1.0957	1.1883	1.1699	1.5155	1.5584	1.7826	1.5315	1.2318
water drop	LBM	50^3	1.9095	1.9687	1.8346	2.0708	1.8280	1.9787	1.9620	1.9264	1.9271
water drop	LBM	60^3	1.2617	1.4168	1.3258	1.7971	1.4298	1.4253	1.3824	1.3462	1.3581
water drop	LBM	64^3	1.1299	1.1837	1.0890	1.3264	1.3137	1.2559	1.2067	1.1423	1.1450
water drop	HLBM	50^3	1.4388	1.5026	1.3983	1.9262	1.4577	1.4862	1.4888	1.4507	1.4789

Table 4.2: The mean of the geometrical distance to the ground truth, where results were obtained using the LBM with the 50^3 grid, the LBM with the 60^3 grid, the LBM with the 64^3 grid, and the HLBM with the 50^3 grid and geometrical distances were calculated for every fifth frame.

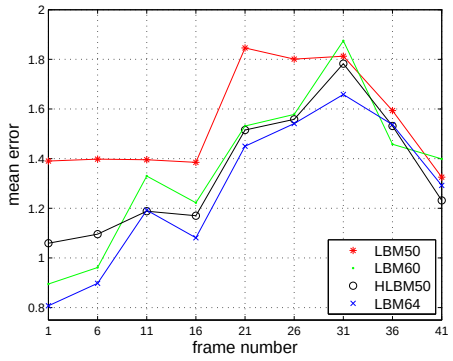
frame number			1	6	11	16	21	26	31	36	41
broken dam	LBM	50^3	0.8071	1.0939	0.9880	1.0442	1.8604	1.9892	2.3704	1.9391	1.0136
broken dam	LBM	60^3	0.3840	0.5090	1.0040	0.8678	1.3458	1.8347	3.6313	1.6073	1.2040
broken dam	LBM	64^3	0.2716	0.4542	1.2410	0.6734	1.4046	1.8883	2.9675	2.2410	1.1349
broken dam	HLBM	50^3	0.4892	0.6729	0.9241	0.8014	1.4740	1.6775	3.2785	1.9424	0.9815
water drop	LBM	50^3	1.5986	1.7722	1.6713	2.7791	1.4224	1.8309	1.8254	1.8262	1.8622
water drop	LBM	60^3	0.8145	0.8276	0.9576	8.4270	1.3640	1.1492	0.9811	1.0047	1.0170
water drop	LBM	64^3	0.5123	0.5578	0.5955	1.9227	2.8787	1.4572	0.9623	0.7333	0.7101
water drop	HLBM	50^3	0.9064	1.0187	0.9350	9.0859	1.9982	1.0485	1.0499	1.0188	1.0830

Table 4.3: The variance of the geometrical distance to the ground truth, where results were obtained using the LBM with the 50^3 grid, the LBM with the 60^3 grid, the LBM with the 64^3 grid, and the HLBM with the 50^3 grid, and geometrical distances are calculated for every fifth frame.

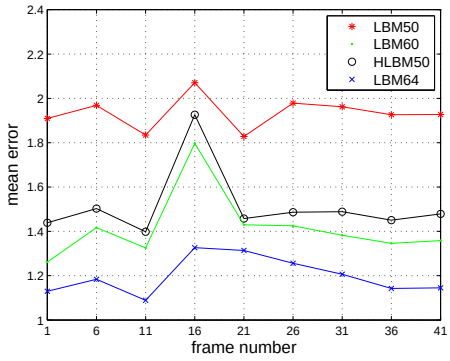
The mean and the variance of the geometrical distances are given in Table 4.2 and Table 4.3, respectively. Based on the data in Table 4.2, we plot the mean of the geometrical error as a function of the frame number in Figs. 4.11 for the broken dam and water drop cases, respectively. We have the following observations.

- For the water drop case, the geometrical distance is significantly larger at the 16^{th} frame since this is a very splashy frame. Similarly, for the broken dam case, the 31^{st} frame is a splashy frame and it also has a larger geometrical distance. Thus, we conclude that the error becomes larger for splashier frames.

- For the broken dam case, the mean error of the HLBM with the 50^3 grid is 13.02% lower than that of the LBM with the 50^3 grid, and 0.96% lower than that of the



(a) Broken Dam



(b) Water Drop

Figure 4.11: The mean of the geometrical error as a function of the frame number for the LBM with the 50^3 grid (the red line), the LBM with the 60^3 grid (the green line), the LBM with the 64^3 grid (the blue line), and the HLBM with the 50^3 grid (the black line).

LBM with the 60^3 grid. The HLBM with the 50^3 grid performs almost the same as the LBM with the 60^3 grid.

- For the water drop case, the mean error of the HLBM with the 50^3 grid is 21.70% lower than that of the LBM with the 50^3 grid, but 6.95% higher than that of the LBM with the 60^3 grid.

One reason for the performance difference between the broken dam and the water drop cases could be that the water drop case has a splashier effect, which lowers the accuracy of the HLBM when the grid resolution is low.

As to the computational complexity, the simulation time of the LBM with the 60^3 grid demands 1.7 times more simulation time than the LBM with the 50^3 grid. On the other hand, the HLBM with the 50^3 grid takes about 1.2 times more simulation time than the LBM with the 50^3 grid. Thus, the proposed HLBM improves the quality of the simulation without increasing the computational cost much.

4.6 Conclusion

The PLSM requires a high computational cost to solve the Poisson equation which comes from the global pressure correction step. Although the LBM is simpler and faster than the PLSM, it demands a larger amount of memory. In this work, we integrated the LBM and the PLSM and derived a new method, called the HLBM, to overcome these difficulties. It was shown by experimental results that the HLBM can offer a splashy and dynamic visual effect with the aid of the PLSM. Furthermore, it can improve the quality of the fluid simulation of the LBM without increasing the grid size.

Chapter 5

Hybrid Lattice Boltzmann Method for MCMP Fluid Simulation

5.1 Introduction

The multicomponent-multiphase LBM (MCMP-LBM) method is useful for multiple chemical components such as oil and water and multiple phases such as liquid and vapor. The interfaces between different components and phases originate from specific interactions among fluid molecules. Thus, the macroscopic Navier-Stokes equations are not suitable for solving such microscopic interaction. Instead, the LBM is suitable for the description of the microscopic interaction by modifying the collision operator. The MCMP-LBM method was introduced by Shan and Chen [56] using non-local interactions between particles. Swift *et al.* [66, 67] using the free energy approach to develop the MCMP-LBM method. The LBM method provides a first-order explicit discretization of the Boltzmann equation in a discrete phase-space. The simulation region of the LBM is divided into a Cartesian grid of cells, each of which only interacts with cells of its direct neighborhood. In contrast, the PLSM demands interaction of all cells in the global pressure correction

step. Generally speaking, the LBM is simpler and faster than the PLSM at the cost of two shortcomings. First, it demands more memory to store distribution functions. Second, it has tight time step restrictions.

In this Chapter, we propose an MCMP hybrid lattice Boltzmann method (MCMP-HLBM) that integrates the multicomponent-multiphase LBM and the PLSM to simulate bubble dynamics. The rest of this chapter is organized as follows. The MCMP-LBM method will be introduced in Sec. 5.2. Then, the MCMP hybrid LBM algorithm that integrates the PLSM with the LBM will be presented in Sec. 5.3. Computer simulation results will be shown in Sec. 5.4. Finally, concluding remarks are given in Sec. 5.5.

5.2 Multicomponent-Multiphase Lattice Boltzmann Method

In Shan-Chen's (S-C) model, multiple phases were simulated by introducing non-local interactions between particles at each lattice site [56]. The lattice Boltzmann equation of Shen-Chen's MCMP model can be written as

$$f_{\sigma,i}(\mathbf{x}+\mathbf{e}_i\triangle\mathbf{x}, t+\triangle t) = f_{\sigma,i}(\mathbf{x},t) - \frac{f_{\sigma,i}(\mathbf{x},t) - f^{eq}_{\sigma,i}(\mathbf{x},t)}{\tau_\sigma}, \tag{5.1}$$

where $\triangle t$ and $\triangle\mathbf{x}$ represent the time and the spatial step sizes, respectively, $f^{eq}_{\sigma,i}(\mathbf{x},t)$ is the equilibrium distribution function of component σ used to represent a stationary state of the fluid, and where i is the directional phase space of a lattice. The rate of change toward the equilibrium of component σ is $1/\tau_\sigma$, the inverse of the relaxation time, and it is chosen to produce the desired value of fluid viscosity.

A composite macroscopic velocity represents the flow of the bulk fluid as

$$\mathbf{u}' = \frac{\sum_{\sigma} \frac{1}{\tau_\sigma} \sum_{i} f_{\sigma,i} \mathbf{e}_a}{\sum_{\sigma} \frac{1}{\tau_\sigma} \rho_\sigma}, \tag{5.2}$$

where τ_σ and ρ_σ represent the relaxation time and the density of component σ, respectively. The equilibrium distribution function of a component, σ, is computed from the composite macroscopic velocity as

$$f^{eq}_{\sigma,i} = \omega_i \rho_\sigma(\mathbf{x}, t) \left[1 + 3\frac{\mathbf{e}_i \cdot \mathbf{u}'}{c^2} + \frac{9}{2}\frac{(\mathbf{e}_i \cdot \mathbf{u}')^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}'^2}{c^2} \right], \tag{5.3}$$

where weights ω_i are $4/9$ for $i = 1$, $1/9$ for $i = 2, \cdots, 5$, and $1/36$ for $i = 6, \cdots, 9$ for the 2D simulation and $1/3$ for $i = 1$, $1/18$ for $i = 2, \cdots, 7$, and $1/36$ for $i = 8, \cdots, 19$ for the 3D simulation, respectively, and c is the basic speed on the lattice ($1\ lu \cdot ts^{-1}$ in general). The composite macroscopic velocity, $\mathbf{u}'$, differs from the macroscopic uncoupled velocities which is defined as

$$\mathbf{u}_\sigma = \frac{1}{\rho} \sum_{i} f_{\sigma,i} \mathbf{e}_i \tag{5.4}$$

of the individual fluid. The density for each component is

$$\rho_\sigma = \sum_{i} f_{\sigma,i}. \tag{5.5}$$

The force on fluid component σ is

$$\mathbf{F}_\sigma(\mathbf{x}) = -G\psi_\sigma(\mathbf{x}, t) \sum_{i} \omega_i \psi_{\bar{\sigma}}(\mathbf{x} + \mathbf{e}_i \Delta t, t) \mathbf{e}_i, \tag{5.6}$$

where $\bar{\sigma}$ indicates the other fluid component, ψ_σ is the effective mass function of fluid component σ and commonly taken as the density, $\psi_\sigma = \rho_\sigma$, $\psi_{\bar{\sigma}} = \rho_{\bar{\sigma}}$ and G is the Green function. If only the nearest neighbor interactions were considered, we have

$$G_{\sigma\bar{\sigma}} = \begin{cases} 0, & |\mathbf{x} - \mathbf{x}'| > c \\ g_{\sigma\bar{\sigma}}, & |\mathbf{x} - \mathbf{x}'| = c \ . \end{cases} \tag{5.7}$$

The interaction force term $\tau_\sigma \mathbf{F}_\sigma$ is added to the momentum $\rho_\sigma \mathbf{u}'$ to obtain the velocity for use in the computation of f_σ^{eq} as

$$\mathbf{u}_\sigma^{eq} = \mathbf{u}' + \frac{\tau_\sigma \mathbf{F}_\sigma}{\rho_\sigma}. \tag{5.8}$$

Finally, Eq. (5.3) can be simplified as

$$f_{\sigma,i}^{eq} = \omega_i \rho_\sigma(\mathbf{x}, t) \left[1 + 3\frac{\mathbf{e}_i \cdot \mathbf{u}_\sigma^{eq}}{c^2} + \frac{9}{2}\frac{(\mathbf{e}_i \cdot \mathbf{u}_\sigma^{eq})^2}{c^4} - \frac{3}{2}\frac{(\mathbf{u}_\sigma^{eq})^2}{c^2} \right]. \tag{5.9}$$

5.3 Hybrid Lattice Boltzmann Method for Bubble Simulation

5.3.1 Hybrid Lattice Boltzmann Method for Bubble Simulation

As explained in Chapter 3.3, liquid simulation using the LSM enables smooth surface representation but demands a huge computational cost because of the global pressure correction step to solve the Poisson equation. On the other hand, liquid simulation using the LBM has an efficient basic algorithm that preserves mass as discussed in Sec. 4.2. However, it suffers from a small time step restriction and a high memory requirement [69].

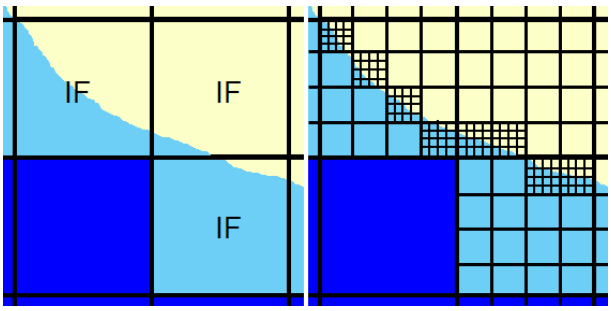


Figure 5.1: The multi-resolution density calculation up to 3[rd] level, where the left figure shows the original profile of cells and the right figure shows the multi-resolution density calculation. Symbols IF denotes the interface cell between two fluids, and the blue and yellow regions represent lattices of fluid types 1 and 2, respectively.

In this section, we propose a hybrid algorithm that integrates the LBM with the PLSM for more realistic and faster bubble simulation.

To combine the MCMP-LBM with the PLSM, we first find the macroscopic velocity field to advect the level set function and particles. The composite macroscopic velocity of each cell can be calculated using Eq. (5.2) and the distance from the center of each cell to the fluid interface can be calculated using the marching cube algorithm [40]. Thus, the level set function can be advected using the macroscopic velocity field, and the semi-Lagrangian advection scheme [60] is used for the advection method.

The multicomponent-multiphase hybrid lattice Boltzmann method (MCMP-HLBM) is described below.

- Step 1: Run the MCMP-LBM solver, where the streaming and the collision steps are performed using Eq. (5.1). The distribution function of fluid component σ in the next time step, $f_{\sigma,i}(\mathbf{x}, t + \Delta t)$, for each lattice is calculated.

- Step 2: Calculate composite macroscopic velocities in the current time step, $\mathbf{u}'(\mathbf{x}, t)$, using Eq. (5.2).

- Step 3: Advect the level set function, $\phi(\mathbf{x}, t)$, and particles, $\mathbf{p}_k(t)$, by the composite macroscopic velocity field, which is calculated in the previous step. The level set function is advected using the semi-Lagrangian advection scheme and particles are advected using the 3^{rd} order TVD-RK method.

- Step 4: Correct errors of the level set function using advected particles in the PLSM method.

- Step 5: Calculate two different density fields obtained from the MCMP-LBM and the PLSM solvers. For the MCMP-LBM part, the density of each cell, ρ_σ^{LB}, can be calculated by the sum of distribution functions using Eq. (5.5). For the PLSM part, the density of each cell, ρ_σ^{PLS}, is calculated using the multi-resolution density calculation scheme up to the 3^{rd} level as described in [36]. Fig. 5.1 also shows the 2D example of the density calculation method.

- Step 6: Add the density difference between the MCMP-LBM and the PLSM solvers to distribution functions to correct errors of the MCMP-LBM as

$$f_{\sigma,i}^{HLBM}(\mathbf{x}, t+\Delta t) = f_{\sigma,i}(\mathbf{x}, t+\Delta t)+\omega_i \left\{\rho_\sigma^{PLS}(\mathbf{x}, t+\Delta t) - \rho_\sigma^{LB}(\mathbf{x}, t+\Delta t)\right\}, \quad (5.10)$$

where $f_{\sigma,i}$ and $f_{\sigma,i}^{HLBM}$ represent distribution functions from the MCMP-LBM and the HLBM of component σ, respectively. In Eq. (5.10), ρ_σ^{LB} and ρ_σ^{PLS} are densities calculated from the MCMP-LBM and the PLSM for component σ, respectively. Note that ω_i is the weight for each directional phase used in Eq. (5.3).

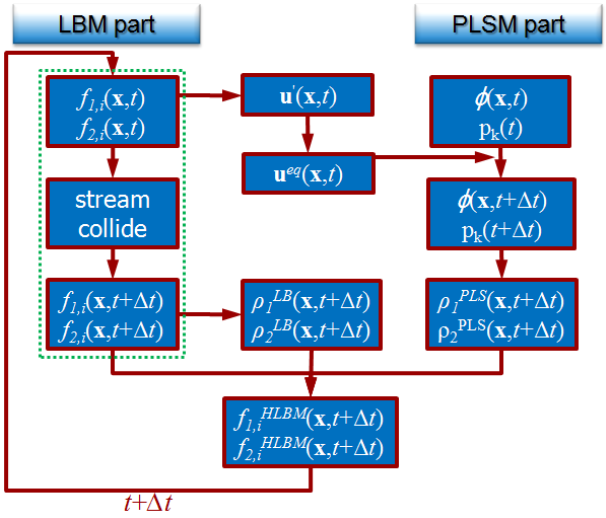


Figure 5.2: An overview of the MCMP hybrid lattice Boltzmann method for bubble simulation, where the dotted box (green) represents the overview of the MCMP-LBM only. In the next time step, $f_{\sigma,i}^{HLBM}(\mathbf{x}, t + \Delta t)$ is used as an input distribution function instead of $f_{\sigma,i}(\mathbf{x}, t + \Delta t)$.

Fig. 5.2 shows a schematic overview of the MCMP-HBLM algorithm, where $\mathbf{p}_k(t)$ represents the particle with id k at time t. In the next time step, $f_{\sigma,i}^{HLBM}(\mathbf{x}, t + \Delta t)$ is used as an input distribution function instead of $f_{\sigma,i}(\mathbf{x}, t + \Delta t)$.

5.3.2 Parameterization of MCMP-HLBM

To check the validity of the MCMP-HLBM's simulation, we compare simulation results obtained by the MCMP-LBM and the MCMP-HLBM with the ground truth. In this chapter, all ground truth data are generated using the MCMP-LBM which has a much higher resolution than other methods. For the simulation of bubbles, there are four main physical values to be considered:

ν' $[m^2/s]$ - kinematic viscosity of fluid

L' $[m]$ - characteristic length of real fluid field

U' $[m/s]$ - characteristic velocity

g' $[m/s^2]$ - strength of external force (*e.g.*, gravity)

The above physical values are converted to dimensionless lattice values via

ν - lattice viscosity

L - lattice characteristic length of real fluid

U - lattice characteristic velocity

g - lattice force

Reynolds number Re is a dimensionless number that gives a measure of the ratio of the inertial force to the viscous force and consequently quantifies the relative importance of these two types of forces for given flow conditions [13]. In order for the two flows to be similar, they must have the same geometry. Thus, they have the same Reynolds number. For all simulations used in this chapter, the Reynolds number can be written as

$$\mathrm{Re} = \frac{\mathrm{U'L'}}{\nu'}. \tag{5.11}$$

Given, L', L, U, ν_1' and ν_2', the spatial discrimination Δx can be written as

$$\Delta x = \frac{L'}{L}, \tag{5.12}$$

where ν_1' and ν_2' represent kinematic viscosity of fluid components 1 and 2, respectively. Then, U' can be calculated as

$$U = \frac{\mathrm{Re}_1 \times \nu_1}{L'}, \tag{5.13}$$

where Re_1 and Re_2 are the Reynolds numbers of fluid components 1 and 2, respectively. The time step size Δt is calculated as

$$\Delta t = \frac{L' \times U}{L \times U'}. \tag{5.14}$$

Then, ν and g can be parameterized using Δx and Δt as

$$\nu_1 = \nu_1' \frac{\Delta t}{\Delta x^2}, \tag{5.15}$$

$$g = g' \frac{\Delta t^2}{\Delta x}. \tag{5.16}$$

Re_2 can be fixed as

$$\mathrm{Re}_2 = \frac{U' \times L'}{\nu_2'}, \tag{5.17}$$

because U' and L' are identical for both components. Also, ν_2 can be calculated as

$$\nu_2 = \nu_2' \frac{\Delta t}{\Delta x^2}. \tag{5.18}$$

Finally, the relaxation time of component 1 and component 2 can be calculated as

$$\tau_1 = 0.5 + 3\nu_1, \tag{5.19}$$

$$\tau_2 = 0.5 + 3\nu_2. \tag{5.20}$$

Using the above parameterization technique, the ground truth simulation is physically similar to low resolution simulation.

5.4 Experimental Results

For fluid animation, the Boltzmann equation needs to be solved numerically using a system called the fluid solver. We use the MCMP-LBM solver based on the S-C model for this purpose [56]. For the hybrid MCMP-LBM (MCMP-HLBM) solver, we added the PLSM modules such as level set functions, particles, and error correction to the MCMP-LBM solver. After that, we scaled the resulting density matrix ρ_σ, which came from the MCMP-HLBM solver, using 2D data interpolation for 2D simulations. Finally, the scaled density matrix can be visualized using the marching cube algorithm [40] which visualizes density data. For 3D simulations, the interpolation part was skipped because scaling is possible with a 3D obj file which comes from the marching cube algorithm. Simulation results in this chapter were obtained using a PC with a 2.2GHz CPU and 4GB RAM.

5.4.1 2D Case

To show the validity of the 2D MCMP-HLBM solver, we tested two and three bubbles coalescence simulation. Fig. 5.3 shows 4 frames from two bubbles coalescence simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a grid resolution of 45×45. Fig. 5.4 shows 4 frames from three bubbles coalescence simulations using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a grid resolution of 60×60.

We show 4 frames from 2D single-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) in Fig. 5.5. Both results were run at a resolution of 40×80. Then, we show 4 frames from 2D two-bubble rising simulation

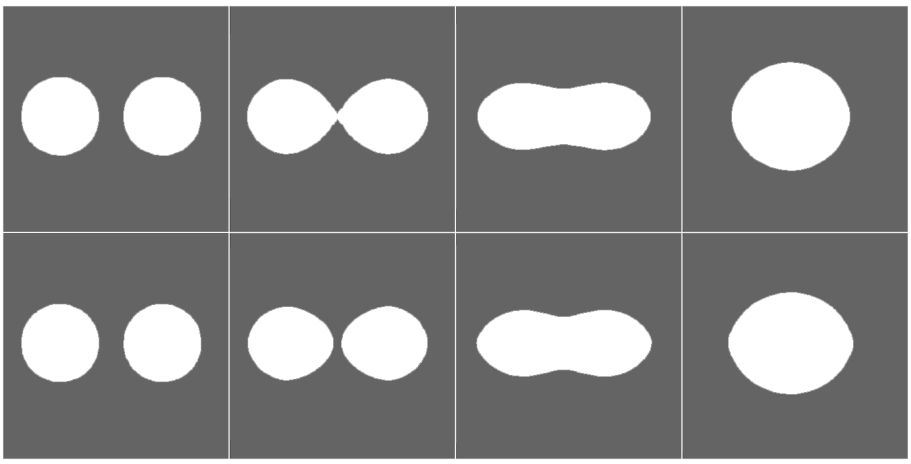


Figure 5.3: The 2D two bubbles coalescence simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of 45×45. White and gray regions represent bubble and liquid, respectively. The characteristic length of the fluid is 1 cm, and the density ratio is 2.66 in all simulations. The columns from the left to the right represent the 1^{st}, the 100^{th}, the 200^{th}, and the 500^{th} frames, respectively. The time interval of the simulation is 0.0111 sec.

using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) in Fig. 5.6. Both the MCMP-LBM and the MCMP-HLBM were run at a resolution of 40×80. Finally, we show 4 frames from 2D three-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) in Fig. 5.7. Both the MCMP-LBM and the MCMP-HLBM were run at a resolution of 60×120.

To quantify the visual improvement of the MCMP-HLBM over the MCMP-LBM, we first obtained the ground truth results for the two/three bubbles coalescence and single/two/three-bubble rising examples using the LBM with a very high resolution grid (*i.e.*, four times larger for each axis) and the same initial and boundary conditions. Then, we got simulation results using the MCMP-LBM and the MCMP-HLBM with grids of lower and middle resolutions and scale their densities, ρ_σ, to match the ground truth simulations using the 2D data interpolation. Finally, we used the normalized absolute

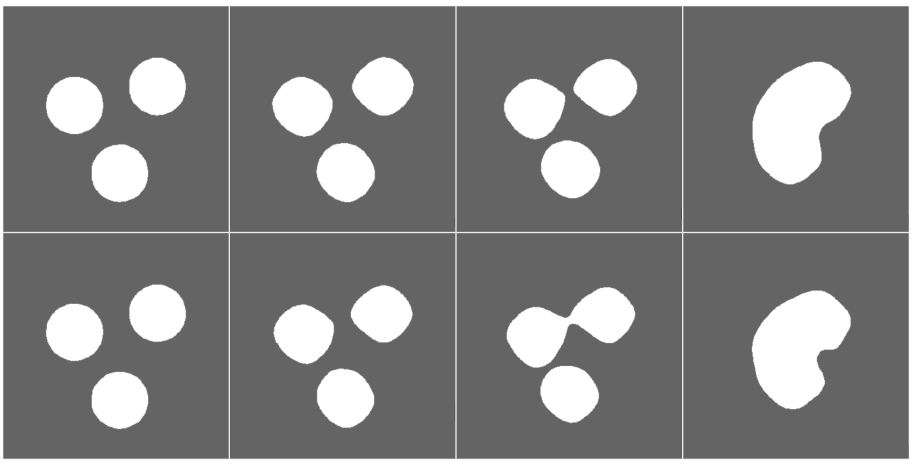


Figure 5.4: The 2D three bubbles coalescence simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of 60×60. White and gray regions represent bubble and liquid, respectively. The columns from the left to the right represent the 1^{st}, the 100^{th}, the 200^{th}, and the 500^{th} frames, respectively. The time interval of the simulation is 0.0082 sec.

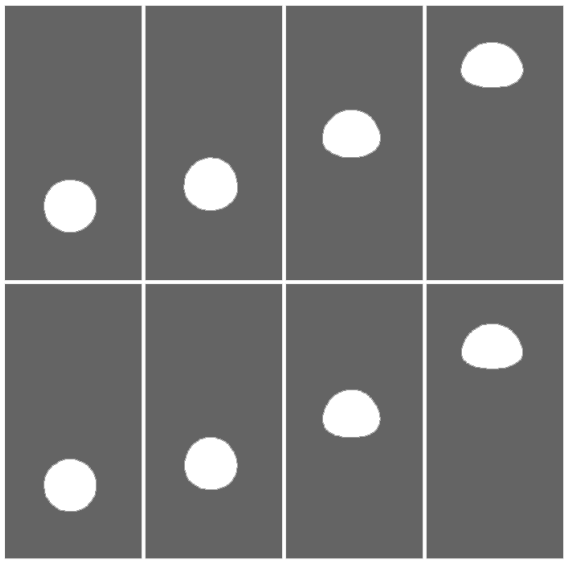


Figure 5.5: The 2D single-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of 40×80. White and gray regions represent bubble and liquid, respectively. The columns from the left to the right represent the 1^{st}, the 100^{th}, the 200^{th}, and the 300^{th} frames, respectively. The time interval of the simulation is 0.0125 sec.

difference to compare the geometric distances between the computed results and their ground truth values; namely,

$$\frac{1}{M \times N} \sum_{m,n} \left| a_{m,n} - a'_{m,n} \right|, \tag{5.21}$$

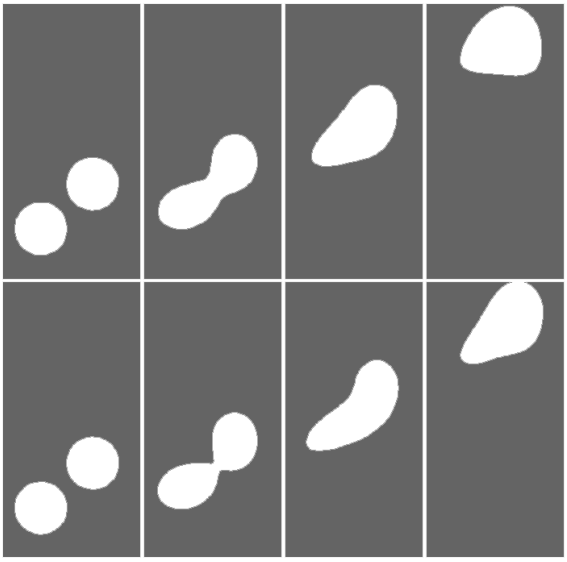


Figure 5.6: The 2D two-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of 40×80. White and gray regions represent bubble and liquid, respectively. The columns from the left to the right represent the 1^{st}, the 100^{th}, the 200^{th}, and the 300^{th} frames, respectively. The time interval of the simulation is 0.0125 sec.

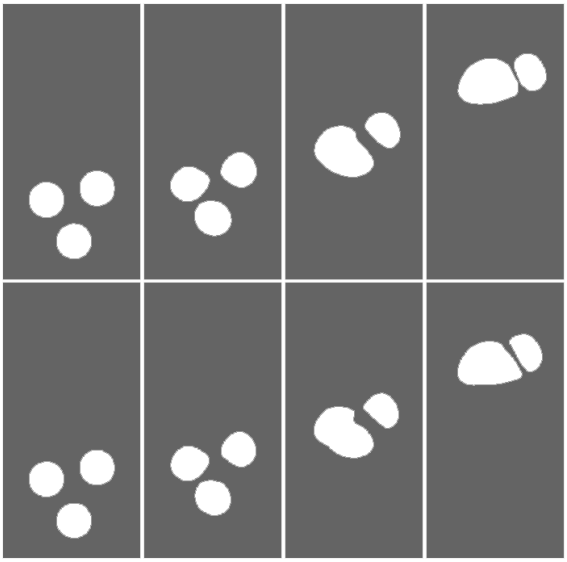
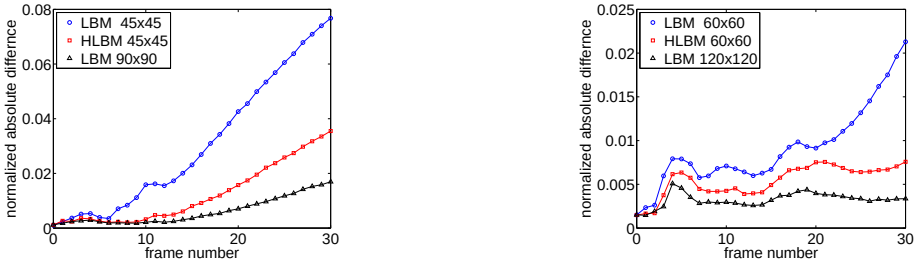


Figure 5.7: The 2D three-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of 60×120. White and gray regions represent bubble and liquid, respectively. The columns from the left to the right represent the 1^{st}, the 150^{th}, the 300^{th}, and the 450^{th} frames, respectively. The time interval of the simulation is 0.0083 sec.

frame number			1	50	100	150	200	250	300
2 Bub. Coal.	LBM	45×45	0.0011	0.0039	0.0158	0.0231	0.0426	0.0605	0.0768
	HLBM	45×45	0.0011	0.0025	0.0032	0.0081	0.0157	0.0258	0.0355
	LBM	90×90	0.0012	0.0023	0.0021	0.0036	0.0071	0.0118	0.0169
3 Bub. Coal.	LBM	60×60	0.0015	0.0079	0.0071	0.0067	0.0091	0.0132	0.0213
	HLBM	60×60	0.0015	0.0063	0.0043	0.0049	0.0075	0.0064	0.0076
	LBM	120×120	0.0015	0.0046	0.0030	0.0032	0.0040	0.0034	0.0034
1 Bub. Rising	LBM	40×80	0.0063	0.0061	0.0076	0.0104	0.0149	0.0223	0.0294
	HLBM	40×80	0.0033	0.0031	0.0039	0.0057	0.0085	0.0130	0.0187
	LBM	80×160	0.0018	0.0016	0.0021	0.0032	0.0046	0.0074	0.0111
2 Bub. Rising	LBM	40×80	0.0130	0.0128	0.0180	0.0254	0.0420	0.0585	0.0735
	HLBM	40×80	0.0065	0.0061	0.0077	0.0106	0.0176	0.0259	0.0373
	LBM	80×160	0.0032	0.0035	0.0038	0.0048	0.0076	0.0115	0.0176
3 Bub. Rising	LBM	60×120	0.0086	0.0097	0.0103	0.0136	0.0139	0.0167	0.0213
	HLBM	60×120	0.0044	0.0060	0.0056	0.0081	0.0088	0.0101	0.0126
	LBM	120×240	0.0023	0.0039	0.0033	0.0044	0.0049	0.0056	0.0068

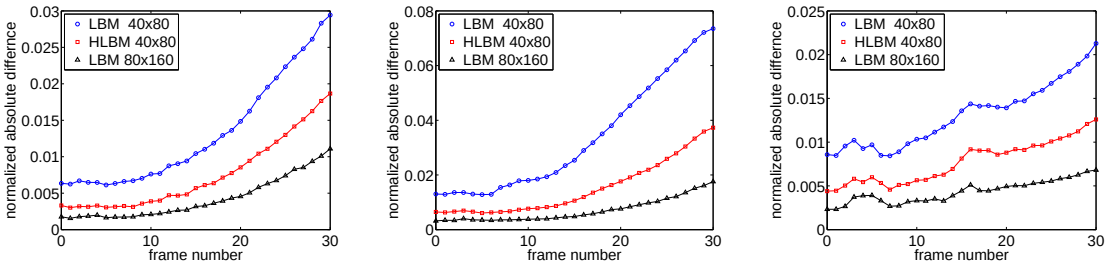
Table 5.1: The normalized absolute difference to the ground truth, where results were obtained using the MCMP-LBM with a grid resolution of $M \times N$, the MCMP-HLBM with a grid resolution of $M \times N$, and the MCMP-LBM with a grid resolution of $2M \times 2N$, and normalized absolute differences were calculated for every fifty frame.

where $a_{m,n}$ and $a'_{m,n}$ are the (m, n) component of the low resolution simulation and the ground truth, respectively, and M and N are numbers of lattices for the x-axis and the y-axis, respectively. $a_{m,n}$ and $a'_{m,n}$ are binary images with the water and the bubble regions set to one and zero, respectively. The normalized absolute differences of the two/three bubbles coalescence and single/two/three-bubble rising simulations are given in Table 5.1. Based on the data in Table 5.1, we plot the normalized absolute difference as a function of the frame number in Figs. 5.8 and Figs. 5.9 for the 2D bubble coalescence and the 2D bubble rising cases, respectively. As shown in Figs. 5.8, the normalized absolute differences of 2D two- and three-bubble coalescence simulations using the MCMP-HLBM are 61.50% and 36.50% lower than that of the MCMP-LBM with same grid resolution, respectively. Figs. 5.9 show that the normalized absolute differences of 2D single-, two- and three-bubble rising simulations using the MCMP-HLBM are 44.93%, 56.02%, and 40.62% lower than that of the MCMP-LBM with the same grid resolution, respectively.



(a) Two-bubble coalescence (b) Three-bubble coalescence

Figure 5.8: The normalized absolute difference as a function of the frame number for (a) two-bubble and (b) three-bubble coalescence simulations, where the MCMP-LBM simulation with a grid resolution 60×60 (the blue circle), the MCMP-HLBM simulation with a grid resolution 60×60 (the red square), and the MCMP-LBM simulation with a grid resolution 120×120 (the black triangle) are compared with the ground truth simulation with a grid resolution 240×240.



(a) Single-bubble rising (b) Two-bubble rising (c) Three-bubble rising

Figure 5.9: The normalized absolute difference as a function of the frame number for (a) the single-bubble, (b) the two-bubble, and (c) the three-bubble rising simulations. In (a), we compare the MCMP-LBM simulation with a grid resolution 40×80 (the blue circle), the MCMP-HLBM simulation with a grid resolution 40×80 (the red square), and the MCMP-LBM simulation with a grid resolution 80×160 (the black triangle) with the ground truth simulation with a grid resolution 160×320. In (b), we compare the MCMP-LBM simulation with a grid resolution 40×80 (the blue circle), the MCMP-HLBM simulation with a grid resolution 40×80 (the red square), and the MCMP-LBM simulation with a grid resolution 80×160 (the black triangle) with the ground truth simulation with a grid resolution 160×320. In (c), we compare the MCMP-LBM simulation with a grid resolution 60×120 (the blue circle), the MCMP-HLBM simulation with a grid resolution 60×120 (the red square), and the MCMP-LBM simulation with a grid resolution 120×240 (the black triangle) with the ground truth simulation with a grid resolution 240×480.

The mean simulation time of the 2D bubble coalescence and rising simulations is given in Table 5.2. For 2D bubble coalescence cases, the mean simulation time of the MCMP-HLBM is 36.25% larger than that of the MCMP-LBM. For 2D bubble rising

	resolution	LBM(sec)	HLBM(sec)	ratio(%)
2 Bub. Coal.	45×45	0.0165	0.0223	35.21
3 Bub. Coal.	60×60	0.0272	0.0373	37.28
1 Bub. Rising	40×80	0.0250	0.0340	36.19
2 Bub. Rising	40×80	0.0253	0.0343	35.54
3 Bub. Rising	60×120	0.0511	0.0741	44.94

Table 5.2: The mean simulation time using the 2D LBM and the 2D HLBM with the same grid resolution.

cases, the mean simulation time of the MCMP-HLBM is 38.89% larger than that of the MCMP-LBM. From Table 5.1 and Table 5.2, we can observe that the MCMP-HLBM bubble simulation improves the quality of the output by reducing the normalized absolute difference than the MCMP-LBM bubble simulation. On the other hand, the MCMP-HLBM bubble simulation requires more simulation time than the MCMP-LBM bubble simulation.

5.4.2 3D Case

Bubble rising simulations are also performed in the 3D space using the 3D MCMP-HLBM solver. We first show 5 frames from the 3D single-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) in Fig. 5.10. Both the MCMP-LBM and the MCMP-HLBM were run with a resolution of 20×20×40. Then, we show 5 frames from 3D two-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) in Fig. 5.11. Both the MCMP-LBM and the MCMP-HLBM were run with a resolution of 20×20×40. Finally, we show 4 frames from 3D three-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) in Fig. 5.12. Both the MCMP-LBM and the MCMP-HLBM were run with a resolution of 20×20×40.

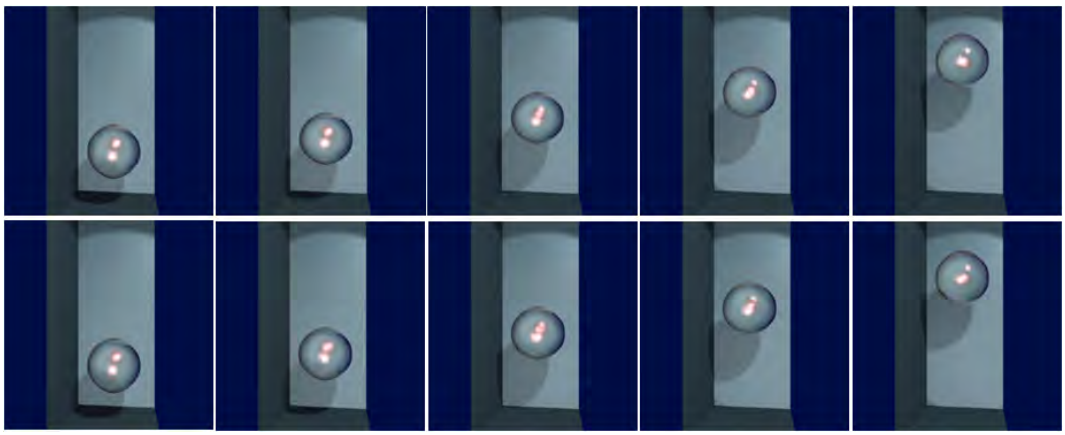


Figure 5.10: The 3D single-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of $20 \times 20 \times 40$. The columns from the left to the right represent the 1000^{th}, the 6000^{th}, the 11000^{th}, the 16000^{th}, and the 21000^{th} frames, respectively. The time interval of the simulation is 0.025 sec.

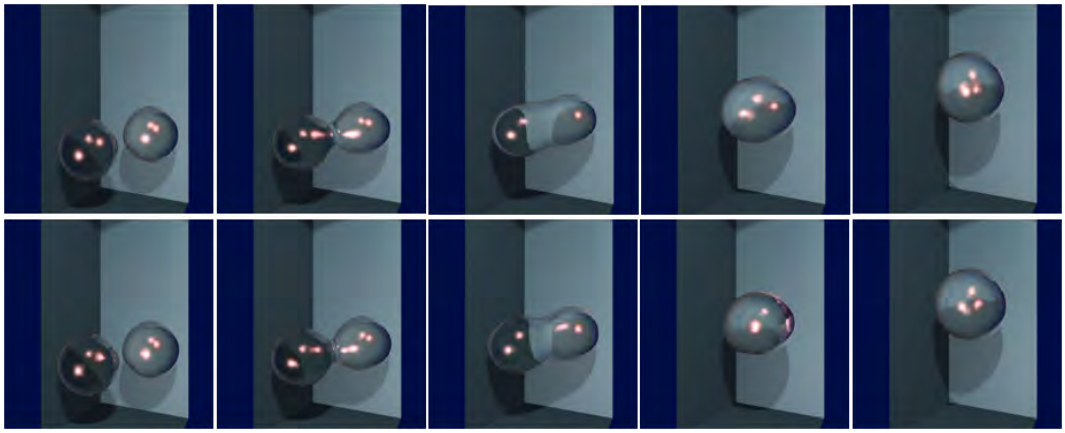


Figure 5.11: The 3D two-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of $20 \times 20 \times 40$. The columns from the left to the right represent the 1000^{th}, the 3000^{th}, the 5000^{th}, the 7000^{th}, and the 9000^{th} frames, respectively. The time interval of the simulation is 0.025 sec.

To quantify the visual improvement of the MCMP-HLBM over the MCMP-LBM, we first obtained the ground truth for bubble rising examples using the MCMP-LBM with a very high resolution grid (*i.e.*, 80×80×160) and the same initial and boundary conditions. Then, we get simulation results using the MCMP-LBM and the MCMP-HLBM with grids of lower resolution. They include: the MCMP-LBM with the 20×20×40 grid, the

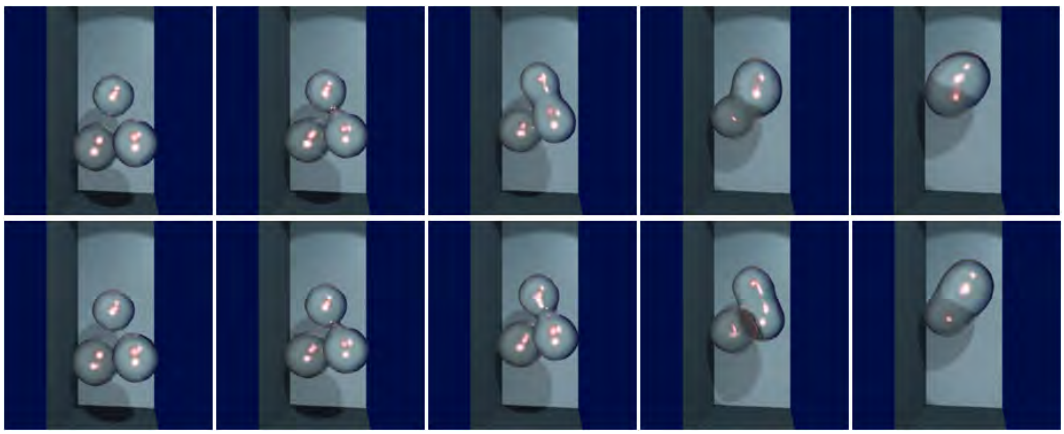


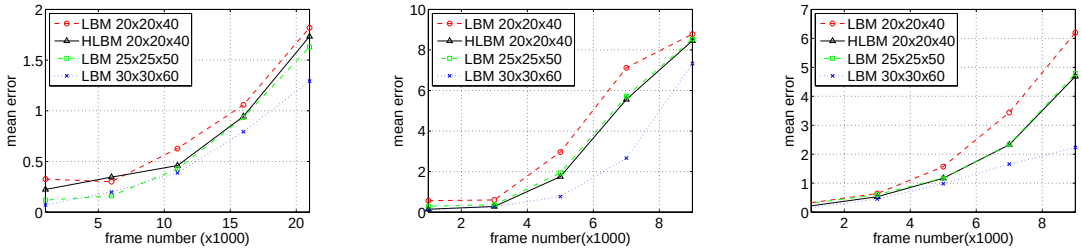
Figure 5.12: The 3D three-bubble rising simulation using the MCMP-LBM (the top row) and the MCMP-HLBM (the bottom row) with a resolution of $20 \times 20 \times 40$. The columns from the left to the right represent the 1000^{th}, the 3000^{th}, the 5000^{th}, the 7000^{th}, and the 9000^{th} frames, respectively. The time interval of the simulation is 0.025 sec.

MCMP-LBM with the 25×25×50 grid, the MCMP-LBM with the 30×30×60 grid, and the MCMP-HLBM with the 20×20×40 grid. Finally, we use the MeshDev [52] to compare the geometric distances between the computed results and their ground truth values.

The means of the geometrical distances are given in Table 5.3. Based on the data in Table 5.3, we plot the mean of the geometrical error as a function of the frame number in Figs. 5.13 for 3D single-, two-, and three-bubble rising cases, respectively. For all bubble

frame number			1000	6000	11000	16000	21000
1 Bub. Rising	LBM	20×20×40	0.3261	0.3002	0.6277	1.0584	1.8198
	HLBM	20×20×40	0.2240	0.3449	0.4592	0.9450	1.7334
	LBM	25×25×50	0.1185	0.1607	0.4283	0.9340	1.6291
	LBM	30×30×60	0.0712	0.1977	0.3870	0.7932	1.2925
frame number			1000	3000	5000	7000	9000
2 Bub. Rising	LBM	20×20×40	0.5656	0.6034	2.9652	7.1229	8.7807
	HLBM	20×20×40	0.1459	0.2773	1.7533	5.5581	8.4566
	LBM	25×25×50	0.2927	0.3622	1.9398	5.7156	8.5382
	LBM	30×30×60	0.1171	0.2671	0.7676	2.6610	7.3293
3 Bub. Rising	LBM	20×20×40	0.3267	0.6413	1.5727	3.4407	6.1982
	HLBM	20×20×40	0.2174	0.5319	1.1710	2.3263	4.6874
	LBM	25×25×50	0.3110	0.6029	1.1734	2.3400	4.7860
	LBM	30×30×60	0.1564	0.4531	0.9817	1.6571	2.2302

Table 5.3: The mean of the geometrical distance to the ground truth, where results were obtained using the MCMP-LBM with a grid of resolution 20×20×40, the MCMP-HLBM with a grid of resolution 20×20×40, the MCMP-LBM with a grid of resolution 25×25×50, and the MCMP-LBM with a grid of resolution 30×30×60.



(a) 3D single-bubble rising (b) 3D two-bubble rising (c) 3D three-bubble rising

Figure 5.13: The mean of the geometrical error as a function of the frame number for (a) single-, (b) two-, and (c) three-bubble rising three dimensional simulations. We compare the MCMP-LBM simulation with a grid of resolution 20×20×40 (the red circle dashed line), the MCMP-HLBM simulation with a grid of resolution 20×20×40 (the black triangle solid line), the MCMP-LBM simulation with a grid of resolution 25×25×50 (the green square dash-dot line), and the MCMP-LBM simulation with a grid of resolution 30×30×60 (the blue cross dotted line) with the ground truth obtained from a grid of resolution 80×80×160.

rising simulations, the mean of the geometrical error using the MCMP-HLBM with a grid of resolution 20×20×40 is lower than the mean error using the MCMP-LBM with a grid of resolution 20×20×40 but higher than the mean error using the MCMP-LBM with a grid of resolution 30×30×60 except for the 6000^{th} frame of the single-bubble rising simulation. For the case of the MCMP-LBM with a grid of resolution 25×25×50, the mean error is almost same as the MCMP-HLBM with a grid of resolution 20×20×40 except for the early part of single-bubble rising simulation. AS shown in Figs. 5.13, the mean of the geometrical error of 3D single-, two-, and three-bubble rising simulations using the MCMP-HLBM are 11.75%, 38.95%, and 26.57% lower than that of the MCMP-LBM with same grid resolution, respectively. Also the mean of the geometrical error of 3D two- and three-bubble rising simulations using the MCMP-HLBM with a resolution of $20 \times 20 \times 40$ are 11.38% and 8.94% lower than that of the MCMP-LBM with a resolution of $25 \times 25 \times 50$, respectively. But the mean of the geometrical error of 3D single-bubble

rising simulation using the MCMP-HLBM with a resolution of $20 \times 20 \times 40$ is 43.69% higher than that of the MCMP-LBM with a resolution of $25 \times 25 \times 50$.

We observe from other experiments that the performance of the MCMP-HLBM is better than the MCMP-LBM when the geometry of the simulation is complex. Note that the geometry of the single-bubble rising simulation is very simple and does not have any coalescence. Also, the mean error is much more smaller than two- or three-bubble rising cases. Thus, the MCMP-HLBM does not have significant performance advantage over the MCMP-LBM in the single-bubble rising simulation. However, the MCMP-HLBM improved the quality of the simulation with complex geometry without increasing the number of grid points. For example, the MCMP-HLBM bubble rising simulation with a grid of resolution $20\times20\times40$ has a similar mean error to the MCMP-LBM bubble rising simulations with a grid of resolution $25\times25\times50$.

As to computational complexity, the simulation time of the MCMP-LBM with a grid of size $25\times25\times50$ demands 1.95 times more simulation time than the MCMP-LBM with a grid of size $20\times20\times40$. On the other hand, the MCMP-HLBM with a grid of size $20\times20\times40$ takes about 1.42 times more simulation time than the MCMP-LBM with a grid of size $20\times20\times40$. As a result, the proposed MCMP-HLBM improves the quality of the simulation without increasing the computational cost much. Finally, as mentioned in Sec. 5.2, the MCMP-LBM is not suitable for high resolution fluid simulation because of its high memory requirement. In contrast, the MCMP-HLBM enables high resolution fluid simulation with the aid of the MCMP-PLSM.

5.5 Conclusion

In this chapter, we developed a new MCMP-HLBM method for bubble simulation. The MCMP-LBM method offers a simple and fast algorithm. However, it demands a huge memory to store distribution functions for bubble and liquid. On the other hand, the PLSM requires a high computational cost to solve the Poisson equation in the global pressure correction step. In this work, we integrated the MCMP-LBM method and the PLSM method and derived a new method, called the MCMP-HLBM, to overcome these difficulties. It was shown by experimental results that the MCMP-HLBM can offer a finer resolution simulation with the aid of the PLSM. Furthermore, it can improve the quality of fluid simulation based on the MCMP-LBM without increasing the grid size.

Chapter 6

Conclusion

The particle level set method (PLSM) and the lattice Boltzmann method (LBM) are two state-of-the-art fluid simulation methods. In this research, we proposed two more sophisticated methods; namely, hybrid LBM (HLBM) and multicomponent-multiphase hybrid LBM (MCMP-HLBM), to enhance their performance.

The LBM solves the lattice Boltzmann equation numerically by dividing it into 2 steps: the streaming step and the collision step. Furthermore, the free surface and the obstacle boundary conditions are applied for realistic fluid simulation. The computation of the LBM is more efficient than that of the PLSM. However, the LBM demands a lot of memory to store discrete velocities at each lattice. Besides, the surface of fluids looks scattered and sometimes flickering since the LBM generates surfaces locally using each lattice and its neighboring cells. To address this problem, we proposed a hybrid LBM (HLBM), in which the level set function and particles are used together for fast and realistic surface calculation and reconstruction of free surface fluid.

Multicomponent-multiphase HLBM (MCMP-HLBM) enables bubble simulation. Although the MCMP-LBM is simpler and faster than the PLSM, it demands a larger amount

of memory. To address this problem, we proposed the MCMP-HLBM, in which the level set function and particles are used together for fast and realistic surface calculation and reconstruction of liquid-bubble interface.

Bibliography

[1] D. Adalsteinsson and J. Sethian, "The fast construction of extension velocities in level set methods," *Journal of Computational Physics*, vol. 148, pp. 2–22, 1998.

[2] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge University Press, 1967.

[3] P. L. Bhatnagar, E. P. Gross, and M. Krook, "A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems," *Phys. Rev.*, vol. 94, no. 3, pp. 511–525, May 1954.

[4] Blender Foundation, "Blender - The free open source 3D content creation suite," http://www.blender.org/.

[5] M. Bouzidi, D. d'Humières, P. Lallemand, and L.-S. Luo, "Lattice Boltzmann equation on a two-dimensional rectangular grid," *J. Comput. Phys.*, vol. 172, no. 2, pp. 704–717, 2001.

[6] M. Carlson, P. J. Mucha, and G. Turk, "Rigid fluid: Animating the interplay between rigid bodies and fluid," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 377–384, 2004.

[7] H. Chen, S. Chen, and W. H. Matthaeus, "Recovery of the navier-stokes equations using a lattice-gas Boltzmann method," *Phys. Rev. A*, vol. 45, no. 8, pp. 5339–5342, 1992.

[8] S. Clavet, P. Beaudoin, and P. Poulin, "Particle-based viscoelastic fluid simulation," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM Press, 2005, pp. 219–228.

[9] D. Enright, F. Losasso, and R. Fedkiw, "A fast and accurate semi-Lagrangian particle level set method," *Computers and Structures*, vol. 83, pp. 479–490, 2005.

[10] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A hybrid particle level set method for improved interface capturing," *J. Comput. Phys.*, vol. 183, no. 1, pp. 83–116, 2002.

[11] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 736–744, 2002.

[12] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher, "A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the Ghost Fluid Method)," *J. Comput. Phys.*, vol. 152, no. 2, pp. 457–492, 1999.

[13] C. Fletcher, *Computational Techniques for Fluid Dynamics*. Springer, 1991.

[14] N. Foster and R. Fedkiw, "Practical animations of liquids," in *SIGGRAPH 2001, Computer Graphics Proceedings*, E. Fiume, Ed. ACM Press / ACM SIGGRAPH, 2001, pp. 23–30.

[15] N. Foster and D. Metaxas, "Realistic animation of liquids," in *Graphical Models and Image Processing*, vol. 1, 1996, pp. 471–483.

[16] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivert, "Lattice gas hydrodynamics in two and three dimensions," in *Complex Systems*, vol. 1, 1987, pp. 649–707.

[17] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron, "Benchmark computations based on Lattice–Boltzmann, finite element and finite volume methods for laminar flows," *Comput. Fluids*, 2004.

[18] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang, "A second-order-accurate symmetric discretization of the Poisson equation on irregular domains," *J. Comput. Phys.*, vol. 176, no. 1, pp. 205–227, 2002.

[19] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw, "Coupling water and smoke to thin deformable and rigid shells," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 973–981, 2005.

[20] A. Gupta and R. Kumar, "Lattice boltzmann simulation to study multiple bubble dynamics," *International Journal of Heat and Mass Transfer*, vol. 51, no. 21-22, pp. 5192 – 5203, 2008.

[21] W. Hackbusch, *Multi-grid Methods and Applications.* Springer-Verlag, 1985.

[22] J. Hardy, O. de Pazzis, and Y. Pomeau, "Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions," *Phys. Rev. A*, vol. 13, no. 5, pp. 1949–1961, May 1976.

[23] F. H. Harlow and E. J. Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," *Phys. Fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.

[24] S. Harris, *An Introduction to the Theory of the Boltzmann Equation.* Holt, Rinehart andWinston Inc., 1971.

[25] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, "Uniformly high order accurate essentially non-oscillatory schemes," *J. Comput. Phys.*, vol. 71, no. 2, pp. 231–303, 1987.

[26] X. He and L.-S. Luo, "Lattice Boltzmann model for the incompressible Navier-Stokes equations," *J. Stat. Phys.*, 88:927.944, 1997.

[27] ——, "A priori derivation of the lattice Boltzmann equation," *Phys. Rev. E*, vol. 55, no. 6, pp. R6333–R6336, Jun 1997.

[28] C. Hirsch, *Numerical Computation of Internal and External Flows.* Wiley, 1988, vol. 1.

[29] C. Hirt and B. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries," *J. Comp. Phys.*, vol. 39, pp. 201–225, 1981.

[30] J.-M. Hong and C.-H. KIM, "Animation of bubbles in liquid," in *Computer Graphics Forum*, vol. 22, 2003, pp. 253–463.

[31] T. Inamuro, T. Ogata, and F. Ogino, "Numerical simulation of bubble flows by the lattice boltzmann method," *Future Gener. Comput. Syst.*, vol. 20, no. 6, pp. 959–964, 2004.

[32] T. Inamuro, S. Tajima, and F. Ogino, "Lattice boltzmann simulation of droplet collision dynamics," *International Journal of Heat and Mass Transfer*, vol. 47, no. 21, pp. 4649 – 4657, 2004, festschrift issue of International Journal of Heat and Mass Transfer honouring Professors Echigo, Hirata and Tanasawa.

[33] J. James Edward Pilliod and E. G. Puckett, "Second-order accurate volume-of-fluid algorithms for tracking material interfaces," *J. Comput. Phys.*, vol. 199, no. 2, pp. 465–502, 2004.

[34] G.-S. Jiang and D. Peng, "Weighted ENO schemes for Hamilton–Jacobi equations," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 2126–2143, 2000.

[35] P. Kundu and I. Cohen, *Fluid Mechanics.* Elsevier Academic Press, 2004.

[36] Y. Kwak, C.-C. J. Kuo, and A. Nakano, "Hybrid lattice-Boltzmann/level-set method for liquid simulation and visualization," *International Journal of Computational Science*, vol. 3, no. 6, pp. 579–592, 2009.

[37] W. Li, X. Wei, and A. Kaufman, "Implementing lattice Boltzmann computation on graphics hardware," *The Visual Computer*, 2003.

[38] G. R. Liu and S. S. Quek, *Finite Element Method: A Practical Course.* Butterworth Heinemann, 2003.

[39] D. P. Lockard, S. Li-Luo, and B. Singer, "Evaluation of the lattice-Boltzmann equation solver powerflow for aerodynamic applications," Tech. Rep., 2000.

[40] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[41] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," in *ACM Trans. Graph. (SIGGRAPH Proc.)*, 2004.

[42] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw, "Multiple interacting liquids," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers.* New York, NY, USA: ACM Press, 2006, pp. 812–819.

[43] A. Masselot and B. Chopard, "A lattice Boltzmann model for particle transport and deposition," *Europhysics Letters*, vol. 42, no. 3, pp. 259–264, may 1998.

[44] G. R. McNamara and G. Zanetti, "Use of the Boltzmann equation to simulate lattice-gas automata," *Phys. Rev. Lett.*, vol. 61, no. 20, pp. 2332–2335, Nov 1988.

[45] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 154–159.

[46] W. F. Noh and P. Woodward, "SLIC (simple line interface calculation)," *in Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*, 1976.

[47] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[48] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[49] T. Pohl, M. Kowarschik, J. Wilke, K. Iglberger, and U. Rüde, "Optimization and profiling of the cache performance of parallel lattice Boltzmann codes in 2D and 3D," Lehrstuhl für Informatik 10 (Systemsimulation), University of Erlangen-Nuremberg, Germany, Tech. Rep. 03–8, July 2003.

[50] P. Raad, S. Chen, and D. Johnson, "The introduction of micro cells to treat pressure in free surface flow problems," in *in Proceedings of the Fluids Engineering Division*, vol. 202, no. 4, 1994, pp. 43–54.

[51] D. H. Rothman and S. Zaleski, "Lattice-gas cellular automata: Simple models of complex hydrodynamics," *Computers in Physics*, vol. 12, no. 6, pp. 576–576, 1998.

[52] M. Roy, S. Foufou, and F. Truchetet, "Mesh comparison using attribute deviation metric," *Int. J. Image Graphics*, vol. 4, no. 1, pp. 127–, 2004.

[53] J. Sethian, "A fast marching level set method for monotonically advancing fronts," in *Proc. Nat. Acad. Sci.*, vol. 93, no. 4, 1996, pp. 1591–1595.

[54] J. Sethian and J. Strain, "Crystal growth and dendritic solidification," 1992, *Journal of Computational Physics* 98, 1992.

[55] J. A. Sethian, "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[56] X. Shan and H. Chen, "Lattice Boltzmann model for simulating flows with multiple phases and components," *Phys. Rev. E*, vol. 47, no. 3, pp. 1815–1819, Mar 1993.

[57] M. Shinya and A. Fournier, "Stochastic motion-motion under the influence of wind," *Comput. Graph. Forum*, vol. 11, no. 3, pp. 119–128, 1992.

[58] J. Smagorinsky, "General circulation experiments with the primitive equations. I. The basic experiment," *Mon. Weather Rev.*, vol. 91, pp. 99–164, 1963.

[59] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford Applied Mathematics and Computing Science Series, 1986.

[60] J. Stam, "Stable fluids," in *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128.

[61] ——, "Real-time fluid dynamics for games," in *The Game Developer Conference*, 2003.

[62] J. Stam and E. Fiume, "Turbulent wind fields for gaseous phenomena," in *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM, 1993, pp. 369–376.

[63] H. Struchtrup and Y. Zheng, "Burnett equations for the ellipsoidal statistical BGK model," *Cont. Mech. Thermodyn.*, 16(1-2):97.108, 2004.

[64] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, 2001.

[65] M. Sussman, P. Smereka, and S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," *J. Comput. Phys.*, vol. 114, pp. 146–159, 1994.

[66] M. R. Swift, W. R. Osborn, and J. M. Yeomans, "Lattice Boltzmann simulation of nonideal fluids," *Phys. Rev. Lett.*, vol. 75, p. 830, 1995.

[67] M. R. Swift, E. Orlandini, W. R. Osborn, and J. M. Yeomans, "Lattice Boltzmann simulations of liquid-gas and binary fluid systems," *Physical Review E*, vol. 54, no. 5, pp. 5041–5052, November 1996.

[68] N. Thürey, "El'Beem - Free Surface Fluid Simulation with the Lattice Boltzmann Method," http://elbeem.sourceforge.net/.

[69] ——, "Physically based animation of free surface flows with the lattice boltzmann method," *PhD thesis*, Mar 2007.

[70] N. Thürey, R. Keiser, U. Rüde, and M. Pauly, "Detail-preserving fluid control," *Proceedings of the 2006 Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pp. 7–15, Jun 2006.

[71] N. Thürey, T. Pohl, U. Rüde, M. Öchsner, and C. Körner, "Optimization and Stabilization of LBM Free Surface Flow Simulations using Adaptive Parameterization," *Computers and Fluids*, vol. 35 [8-9], pp. 934–939, September-November 2006.

[72] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan, "A front-tracking method for the computations of multiphase flow," *J. Comput. Phys.*, vol. 169, no. 2, pp. 708–759, 2001.

[73] X. Wei, Y. Zhao, Z. Fan, W. Li, S. Yoakum-Stover, and A. Kaufman, "Blowing in the wind," in *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2003, pp. 75–85.

[74] Wikipedia, "Claude-louis navier — wikipedia, the free encyclopedia," 2008. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Claude-Louis_Navier&oldid=226679149

[75] ——, "Einstein notation — wikipedia, the free encyclopedia," 2008, [Online; accessed 7-August-2008]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Einstein_notation&oldid=223474765

[76] ——, "George gabriel stokes — wikipedia, the free encyclopedia," 2008. [Online]. Available: http://en.wikipedia.org/w/index.php?title=George_Gabriel_Stokes&oldid=233682551

[77] ——, "Newtonian fluid — wikipedia, the free encyclopedia," 2008. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Newtonian_fluid&oldid=233531140

[78] ——, "Gauss-seidel method — wikipedia, the free encyclopedia," 2010. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Gauss%E2%80%93Seidel_method&oldid=347789763

[79] D. A. Wolf-Gladrow, *Lattice-gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Springer, 2000.

[80] S. Wolfram, *A New Kind of Science*. Wolfram Media, 2002.

[81] H. Yu, S. S. Girimaji, and L.-S. Luo, "Lattice Boltzmann simulations of decaying homogeneous isotropic turbulence," *Phys. Rev. E*, vol. 71, no. 1, pp. 016 708–+, January 2005.

[82] ——, "DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method," *J. Comput. Phys.*, vol. 209, no. 2, pp. 599–616, 2005.

[83] Z. Yu and L.-S. Fan, "An interaction potential based lattice boltzmann method with adaptive mesh refinement (amr) for two-phase flow simulation," *J. Comput. Phys.*, vol. 228, no. 17, pp. 6456–6478, 2009.

[84] H. Zhao, T. Chan, B. Merriman, and S. Osher, "A variational level set approach to multiphase motion," 1996, *J. Comput. Phys.*, v. 127, pp. 179-195.

[85] O. C. Zienkiewicz and R. L. Taylor, *Finite Element Method: Volume 2- Solid Mechanics*. Butterworth Heinemann, 2000.