# Lattice Boltzmann Method and Vortex Methods for Fluid Animation

by

© *Xue Cui*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Science*

Department of *Computer Science*

Memorial University of Newfoundland

*June, 2017*

St. John's Newfoundland

## Abstract

Most fluid solvers for visual effects are based on splitting Navier-Stokes (N-S) PDEs. In the N-S solver, a projection step is often applied to correct the minor errors from advection and produces divergence free constraint for incompressible fluid flow in order to output flows with swirls. It is one of the key steps in classic N-S solver, giving us turbulence visual effects. We follow the conventional N-S based solver, but mainly focus on a growing popular method named as LBM (Lattice Boltzmann Method). Many visual effect researches have proposed to utilize LBM's straightforward parallelizable feature but neglected its capabilities for generating appealing visual effects. By investigating popular vortex methods of classic N-S solvers, we presented hybrid methods for enriching visual complexity for LBM. They require additional steps for vortex calculation and boundary conditions processing on macroscopic and microscopic levels, but are able to provide rich details for the simualtion with acceptable computation cost. Compared to introducting a complex turbulence model for LBM, our approach is easy-to-implement and straightforward.

## Acknowledgements

I would like to express my sincere thanks to my supervisor, Dr.Minglun Gong. He has been providing me the possible research direction of fluid animation, detail discussions, and encouraging innovative ideas and critical thinking. All these are the key points guiding me to be an independent researcher.

I would like to acknowledge that Computer Vision lab, Computer Science, Memorial University of Newfoundland, which is managed by Dr.Minglun Gong, and School of Graduate Study, Memorial University of Newfoundland, provided me continuous financial support.

Additionally, I would like to give my special thanks to the researchers for details discussions from industries and other universities, they provided professional suggestions of programming, scientific computing, and math derivations.

Last but not the least, I would like to thank my parents who have been supporting me from all sides.

# Contents

**Abstract** **ii**

**Acknowledgements** **iii**

**List of Tables** **vi**

**List of Figures** **vii**

**1 Introduction** **1**

1.1 Fluid Animation in the Film Industry . . . . . 1

1.2 Basis of Fluid Animation . . . . . 2

1.2.1 Methods of Fluid Simulation . . . . . 3

1.2.2 Turbulence . . . . . 4

1.2.3 The Arts of Fluid . . . . . 6

1.3 Contributions . . . . . 8

**2 Related Works** **10**

**3 N-S, LBM, and Turbulence Background** **16**

3.1 N-S is the Cornerstone . . . . . 16

3.1.1 Numerical Solution . . . . . 23
3.2 A Mesoscopic Method — LBM . . . . . 27
3.2.1 Foundation of LBM . . . . . 27
3.3 Vortex Methods . . . . . 31
3.3.1 Spinning Wheels . . . . . 31
3.3.2 More Natural Spinnin Wheels . . . . . 34

**4 Algorithms of LBM and Vortex Methods 37**
4.1 Why it is possible to mix . . . . . 37
4.2 How to mix (Algorithms) . . . . . 38
4.2.1 Vorticity Confinement LBM (VCLBM) . . . . . 39
4.2.2 IVOCK-LBM . . . . . 42
4.3 Implementation Discussion . . . . . 43

**5 Conclusion and Results 45**

**6 Future Work 57**

**Bibliography 60**

**A Demo and Code 64**

# List of Tables

5.1 Evaluation for the effects of swirls based on different methods of curl calculations . . . . . 48

5.2 3D Performance of algorithms in different resolution for each step . . 48

# List of Figures


1.1 Common CFD products structure . . . . . 3

1.2 Laminar and Turbulence in real world . . . . . 5

1.3 MAYA smoke rises. (a) is the procedure from Laminar to Turbulence; (b) with hotter generator, smoke does not show any Laminar flow. . . 7

1.4 Two masterpieces about turbulence. . . . . 8

2.1 Resolution: $192 \times 192$. Left side is the semi-Lagrangian advection with IVOCK, while right side is the pure semi-Lagrangian advection scheme. 11

2.2 Resolution: $192 \times 192$, particles clearly depicts the different part from classic method. The right side semi-Lagrangian scheme illustrates a very plain effect without any curls. In the supporting videos could be seen more clearly. . . . . 12

2.3 Resolution:$128x192x128$, we output the smoke density into files and illuminated the scene through volumetric path tracing. . . . . 13

2.4 Resolution: $128x192x128$, the gas lifted from the bottom and hit the boundary. . . . . 14

3.1 Euler view (a) stores the fluid properties in each cell. When any material of fluid moves over the grid, these properties will be changed. Lagrangian view (b) keeps the properties of fluid in the particles, given an appropriate collide model such as the smooth function of SPH method, the particles change their properties accordingly. . . . . 18
3.2 A tiny boat does not have engine! . . . . . 19
3.3 Measure the region of flow . . . . . 21
3.4 Two methods for splitting computational domain . . . . . 25
3.5 Stream the central velocity to 9 directions. . . . . 28
3.6 The Spinning Wheels appears around the upward flowing fluid . . . . 32
3.7 Lower value of $\epsilon$ will provide fewer wheels. . . . . 32
3.8 IVOCK does not have chaotic rotaional wheels in the field but provides natural and reliable shape of the simulation. The support video could be found in the appendix video link. . . . . 35
3.9 **Left:** Velocity field before IVOCK; **Middle:** IVOCK corrections; **Right:** After IVOCK advection . . . . . 36

4.1 General N-S workflow . . . . . 38
4.2 Vorticity Confinement is applied when velocity has been obtained. . . 39
4.3 IVOCK is applied for changing the advection process . . . . . 39
4.4 General processing steps of LBM . . . . . 40
4.5 Velocity has been exported for curl VC processing . . . . . 41
4.6 Workflow of IVOCK-LBM . . . . . 43

5.1 A real experimental lid driven cavity flow, and the detail curls are marked with black curves. (The video could be found through supporting video sites; see Appendix A) . . . . . 46

5.2 Resolution: $32 \times 32 \times 32$, Lid Driven Cavity Flow without vortex methods. It is quite rough to observe the detail of flow . . . . . 49

5.3 Resolution: $128 \times 128 \times 128$, Lid Driven Cavity Flow without vortex methods.By increasing the resolution, more irregular and some turbulence effects could be found around the central rotational part. . . . . 50

5.4 Resolution: $32 \times 32 \times 32$, Lid Driven Cavity Flow with IVOCK and Vorticity Confinement update. Also, the low resolution could give very limited details. However, potential turbulence could be found around the central rotational area. . . . . 51

5.5 Resolution: $128 \times 128 \times 128$, Lid Driven Cavity Flow with IVOCK and Vorticity Confinement update. With a high resolution, turbulence could be simply captured. . . . . 52

5.6 Resolution: $128 \times 128 \times 128$, Lid Driven Cavity Flow with Vorticity Confinement only. The parameter $\epsilon$ is set to 0.008. . . . . 53

5.7 Flow passes over a black tilted boundary. Carefully observe the third frame for both original LBM and VCLBM, the VCLBM has more chaotic patterns appearing after the black panel. . . . . 54

# Chapter 1

# Introduction

## 1.1 Fluid Animation in the Film Industry

Since the very early age of films, fluid was often represented by low cost and easy-to implement methods. For instance, in early Walt Disney cartoons, the overlapped, staggered, and wavy paper walls was used to represent water waves. With personal computer being more popular, visual effects in films have become more complicated and attractive. A milestone of fluid animation in non-cartoon films firstly appeared in Titanic 1997, audiences were totally convinced by the fluctuated ocean waves. Post-process of a huge boat with charaters moving around these 3D models, gives a strong impact visually.

However, our keen eyes will grab any flaws when watching a film on a huge screen. Shortly after, with the revolution of super-computing, physics based fluid animation has gained popularity. Many mathematicians, physicists, and computer scientists were involved in the research. Except for fluid animation, researchers started ob-

serving other natural phenomena as well, and making effort to simulate the world. Therefore, it is not a surprise that after a decade, it is now very difficult for the audiences who do not have related background to judge whether what they have seen on the screen is true or not.

Compared to the classic engineering research such as CFD (Computational Fluid Dynamics), fluid simulation in film industry does not require as much accuracy as CFD. More importantly, controllability and attractiveness are the two key features for film industry.

## 1.2 Basis of Fluid Animation

In the industrial visual effects area, fluid animation is crucial in artists creating spectacular scenes. Computer scientists have been studying fluids from reality, computational cost, and controllability aspects for decades. Audiences have keen vision when fluid motion behaviors in an unnatural manner, even though they could not point out where the problems are.

Researchers grab ideas for fluid animation from engineering CFD (Computational Fluid Dynamics) studies, mixing different ideas to give fluid more reality, controllability and less computation cost features. Regarding reality, turbulence is one of the key factors strongly affecting the visual sense of fluid motion. Unfortunately, there is no exact solution to fully solve turbulence. Turbulence mystery has been remaining to CFD study that the direction of the turbulence could be barely approximated. Optimistically, however, many approximating methods of turbulence have been proposed with highly developed N-S based solvers. Therefore, computer animation could also

benefit from the related researches.

### 1.2.1 Methods of Fluid Simulation

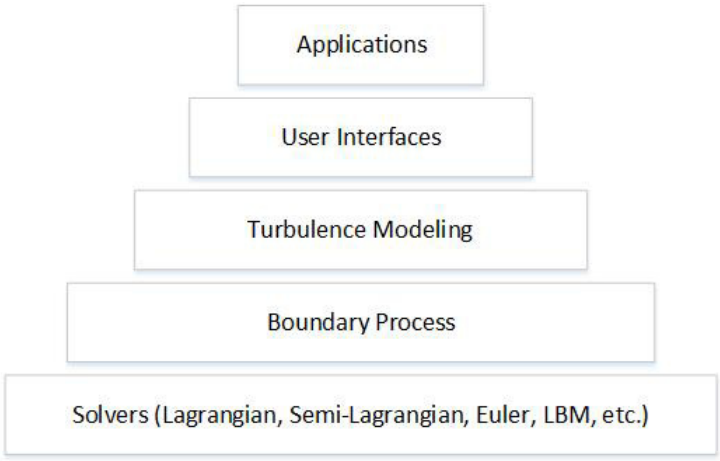



Figure 1.1: Common CFD products structure

The methods of the fluid simualtion are usually named as solvers. Solvers are at the bottom of a CFD product structure pyramid (See fig. 1.1). When building a fluid analyzing software, choosing an appropriate fluid solver is the first step. Then, related boundary processing techniques should be decided in order to reduce errors and to ensure the stability of simulation. Modeling turbulence comes in the next step for more accurate simulation of complex fluid flows. Finally, software engineers will release the interfaces to users and a complete application now is made.

N-S (Navier-Stokes) solver is a very common and popular solution to many fluid problems. N-S is a top down method which starts from splitting the PDEs' (Partial Differential Equations) operators. Computational Science has contributed a lot in solving the PDEs, many ready-made and mature calculation methods could be easily

implemented. Also, many third party software libraries could be imported. Generally, N-S depicts fluid from a high level based on macroscopic concept. It includes fluid viscosity, advection, pressure, and external force. Moreover, N-S based solver has two main branches to solve the group of equations. One is called Lagrangian view, which keeps the particle all around and stores the fluid properties on particles. The other is called Euler view, which starts from splitting the problem domain into tiny cells. These cells would save and update the variables of fluid at every time step.

In addition to the N-S based solver, LBM (Lattice Boltzmann Method) is another approach which updates fluid states at a mesoscopic level by observing tiny units collision and the possibility of directions that these units are likely moving towards. With the highly developed parallel device, GPU, this approach has been extensively studied in its compatibility with concurrent device. This can greatly reduce computational cost while maintain a certain level of accuracy. Progressively, industrial institutions tend to use LBM as their product's infrastructure. However, LBM is still in an early stage in visual effects area. We focus on the turbulent visual effects of LBM which we found has not been sufficiently addressed in literatures previously. For this reason, we start from analyzing N-S based methods and its famous vortex solutions appeared in recent years, then we propose two interfaces mixing vortex methods and LBM in different perspectives to improve its visual effects.

### 1.2.2 Turbulence

In this thesis, we try to introduce all of the concepts as simple as possible so we tend to avoid using many terminologies while to describe the scientific information and

(a) Laminar to Turbulence

(b) Chaos (Turbulence)

Figure 1.2: Laminar and Turbulence in real world

natural phenomena directly. For example, the states of fluid flow (Laminar, between Laminar and Turbulence, Turbulence) are usually distinguished by the term called ***Reynolds number*** which states:

$$Re = \frac{uL}{\nu} \tag{1.1}$$

where $\vec{u}$ is the velocity of the fluid with respect to the object, $L$ is a characteristic linear dimension (imagine a pipe filled up with water, then the $L$ is the diameter of the pipe, or the $L$ is the chord length when calculating Reynolds number of flow of airfoils, so $L$ depends on specific situtaion), and $\nu$ is the kinematic viscosity of the fluid. However, this is trivial to artists because they do not often apply the numbers to get the fluid effects while observe the results visually. Since turbulence has always been a very difficult problem in terms of how visually natural it is or how numerically accurate it is. What fluid animation in films usually does is to refer to the turbulence models from CFD area, but simplify the part of process in order to make it easier to implement, controllable, and visually appealing.

Approximately, the flow of fluid is differed by Laminar flow and Turbulence flow. Laminar flow states the movement of fluid is smooth and stable, swirls and strong flucutation should not be observed in this type of flow. Turbulence is visually defined as chaos in the flow with bunch of rotating parts of fluid, the motion at next second could not be predicted accurately. Moreover, a phnomenon between the pure Laminar and pure Turbulence is more common in daily basis such as the smoke of cigarettes, it will slowly change the states from the one to the other; See fig. 1.2 and fig. 1.3 for illustrations.

Intuitively, turbulence gives us more details of the motion, these details are often illustrated by the little rotating parts of the flow. To audiences, more complicated and chaostic effects would present more interesting scenes and provide more realistic appearances. To analyze how interesting the fluid is, is also a common question for researchers. Honestly, this question has never been solved precisely, the reason is simple, people have different tastes! Therefore, find an appropriate approach to simulate turbulence effects is more crucial than only focus on numerical preciseness for computer animation purpose.

### 1.2.3 The Arts of Fluid

It is worth nothing, Fluid simulation appears not only in classic CFD engineering, but also appears in many art masterpieces. For example, scientists found that the Vincent's painting in The Starry Night (See (a) of fig. 1.4). matches the real world flow patterns (See the article [17]). Also, The Great Wave of Kanagawa by Japanese painter Hokusai. (See (b) of fig. 1.4)) depicts a huge wave moving by, while mountain

(a) Laminar to turbulence

(b) Stronger turbulence

Figure 1.3: MAYA smoke rises. (a) is the procedure from Laminar to Turbulence; (b) with hotter generator, smoke does not show any Laminar flow.

Fuji is a background bringing out how big the wave is. Interesting part is the details of the waves, an exaggerated approach is used to draw those small turbulence on top of the big waves. Sprays also could be found under the top of the wave.

There are many other masterpieces about fluid, but are not listed in this article. They are all from the artists' observation of mysterious, common, and interesting natural phenomena. According to a story, German theoretical physicist Werner Heisenberg was asked what he would ask God, given the opportunity. He said "When I meet God, I am going to ask him two questions: Why relativity? And why turbulence? I really believe he will have an answer for the first.". From Werner's words, we understand how mysterious and important the turbulence is.

(a) The Starry Night

(b) The Great Wave off Kanagawa

Figure 1.4: Two masterpieces about turbulence.

## 1.3 Contributions

In this thesis, we present a novel framework for enriching turbulence effects of LBM. Under this framework, we presented two specific approaches for inserting turbulence models to LBM. The first approach provides means to control the level of turbulence but the parameter needs to be carefully set. Blow up could happen when inappropriate parameters are provided. The second approach is capable of generating more natural turbulence but is not user adjustable. In summary, the presented algorithms have shown the following characteristics:

**Easy-to-Implement** We carefully observed the LBM process and N-S based turbulence modeling, and found the possibility to reuse part of Euler N-S vortex method inserting to LBM. Overall, LBM has streaming and collision steps calculating one timestep fluid behavior. We insert vortex method directly between streaming and collision once we have computed macroscopic velocity. Then vortex could be seen as a correction of macroscopic velocity field.

**Controllable (One of the algorithm)** The first presented algorithm is named

VCLBM (Vortex Confinement Lattice Boltzmann Method). It keeps the feature of vortex confinement that provides a parameter to control the level of turbulence. The parameter should be carefully set as unstable simulation results would happen. In our experiements, the value is between 0.001 and 0.01. Higher value is more likely leading to blow up but the turbulence is strong, whereas lower value is more stable but generates less swirls.

**More Natural (The other algorithm)** The other presented algorithm is called IVOCK-LBM (Integrated Vorticity of Convective Kinematics Lattice Boltzmann Method). Originally it has non-physics sense as IVOCK is only designed for N-S based framework. However, we found a way for connecting IVOCK to LBM. IVOCK method is parameter-free method but provides more natural motion of fluid.

The shortcomes of the algorithms are that they need extra calculation of vortex and boundary process. It rises the cost of computation by about $30\% - 50\%$. A possible solution is to implement the algorithms on parallel devices to get accelerated. We leave it as the future work discussed in chapter 6. We will go to more details of these features in Chapter 3 and 4.

# Chapter 2

# Related Works

For the basic instructions and practical use of fluid animation, Stam's [21] and Bridson's [3] books are fully recommended. The magic behind the scene could be found in the article [5], it discusses the mathematical derivations of fluid mechanics. Moreover, from the perspective of computer science, the processing speed of the implementation may be a concern. Space splitting could accelerate the neighbor searching problem if particles get involved. Hence, Onderik and Durikovic's paper [15] could be useful for extended reading.

Stam's Stable Fluids [19] has been widely used in visual effects industries to simulate fluids since early 2000s. His approach dramatically reduces the computation of traditional CFD, while in the meantime, maintains the reality of the motion. The unconditionally stable feature attracts many researchers to expand or modify his approach such as the work of smoke visualization by Fedkiw, Stam and Jensen's work [7], which introduces vorticity confinement to enrich the swirls visually. By tunning the parameter from vorticity confinement, some swirls could be created outside of

the fluid concentration area, which does not make any sense. Kim, Threy, James and Gross's work[10] provided a wavelet based method to get rid of those non-sense swirls calculated from vorticity confinement. Moreover, Kim, Tessendorf and Threy proposed a method [9] that visually improves the details of liquid surface. In recent work, Zhang, Bridson and Greif's approach [23] introduced a vortex dynamics approach for improving the turbulence, which is a stand-alone method for inexpensively enriching the vorticities of classic N-S framework.

As the motivation of the research, we demonstrate the results based on the IVOCK with N-S solver through our 2D prototype simulation. As shown in fig. 2.1 and fig. 2.2, we use particles as tracers to show more clear results (Vorticies are more obvious). The hot smoke rises upwards with buoyancy. The red marked area is a minor phenomenon called "Leap Frog" (See fig. 2.1), which shows the ability of IVOCK to obtain more details of flow than original semi-Lagragian method which only has velocity self-advection.

We also reproduce the IVOCK in 3D, see the frame sequence fig. 2.3. Besides,

Figure 2.1: Resolution: $192 \times 192$. Left side is the semi-Lagrangian advection with IVOCK, while right side is the pure semi-Lagrangian advection scheme.

Figure 2.2: Resolution: $192 \times 192$, particles clearly depicts the different part from classic method. The right side semi-Lagrangian scheme illustrates a very plain effect without any curls. In the supporting videos could be seen more clearly.

we add internal sphere boundary to our reproduced IVOCK 3D experiment, see fig. 2.4. In addition, there is a more classic and accurate turbulence model called $k-\epsilon$, which is very popular in traditional CFD research. Pomerantz and Dawn studied $k-\epsilon$ turbulence model from a detailed physical and theoritical view [18]. The high computational cost of $k-\epsilon$ model prohibited its application in fluid animation until the work from Tobias et al.[16], which efficiently solved two-equations from $k-\epsilon$. Nevertheless, they incorporated with particle system to reduce the computation in order to make their method real-time. The number of particles is a parameter that controls the details of fluids.

These works demonstrated multiple choices for modeling turbulence in visual effects field. Most of them are based on N-S solver. We consider a combination of LBM and one of the turbulence model to enrich the details of LBM animation.

From original LBM view, insufficient visual turbulence is one of the shortcoming

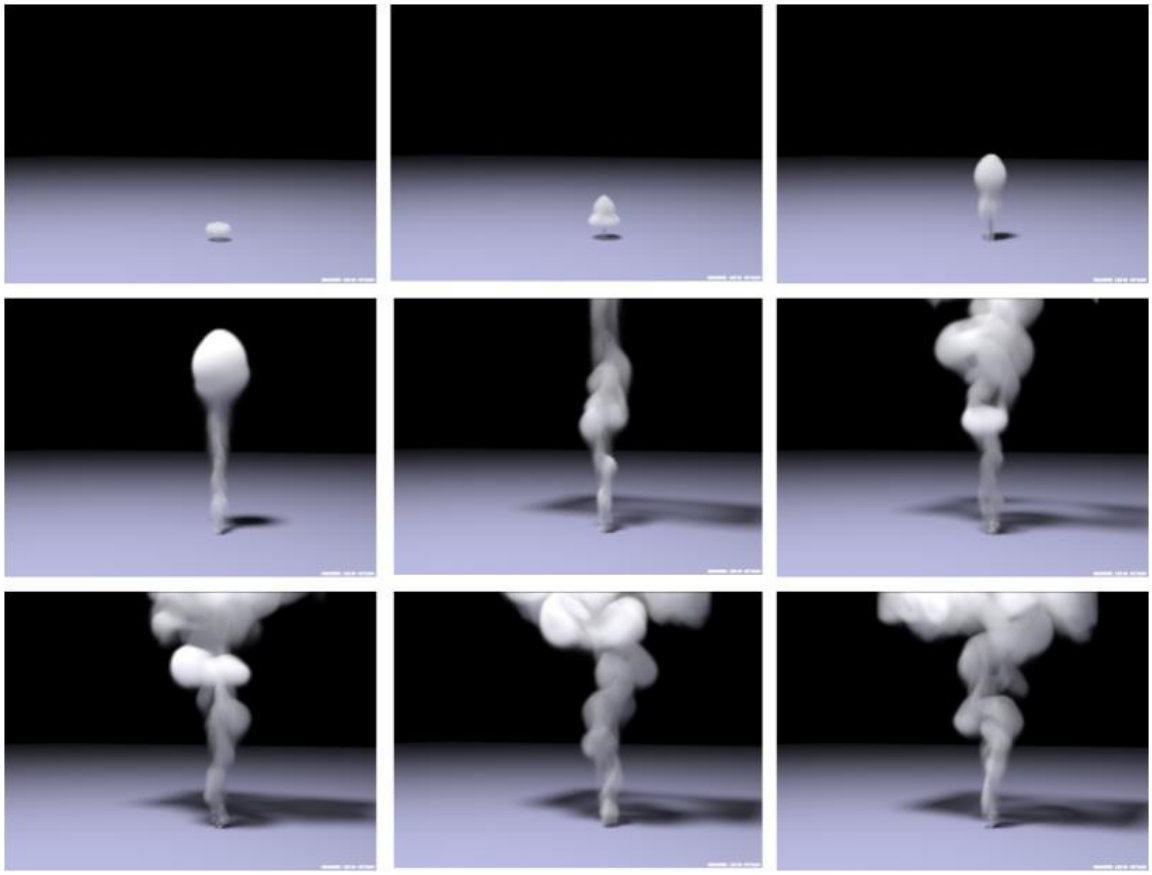


Figure 2.3: Resolution:$128x192x128$, we output the smoke density into files and illuminated the scene through volumetric path tracing.

that LBM left. Inadequate works that integrating turbulence method into LBM have been addressed from visual effects community. Zhao et al. presented a locally-refined LBM [24] for small visual details. However, it still suffers from unnatural turbulence behavior as no vortex methods have been introduced.

We leave the LBM in its original form to solve the particle kinematic energy in mesocopic, but reproduce the work from [23] and [7] by mixing the methods in macroscopic level to enrich the results from LBM solver.

However, as the basis of the research, many other approaches were also investi-

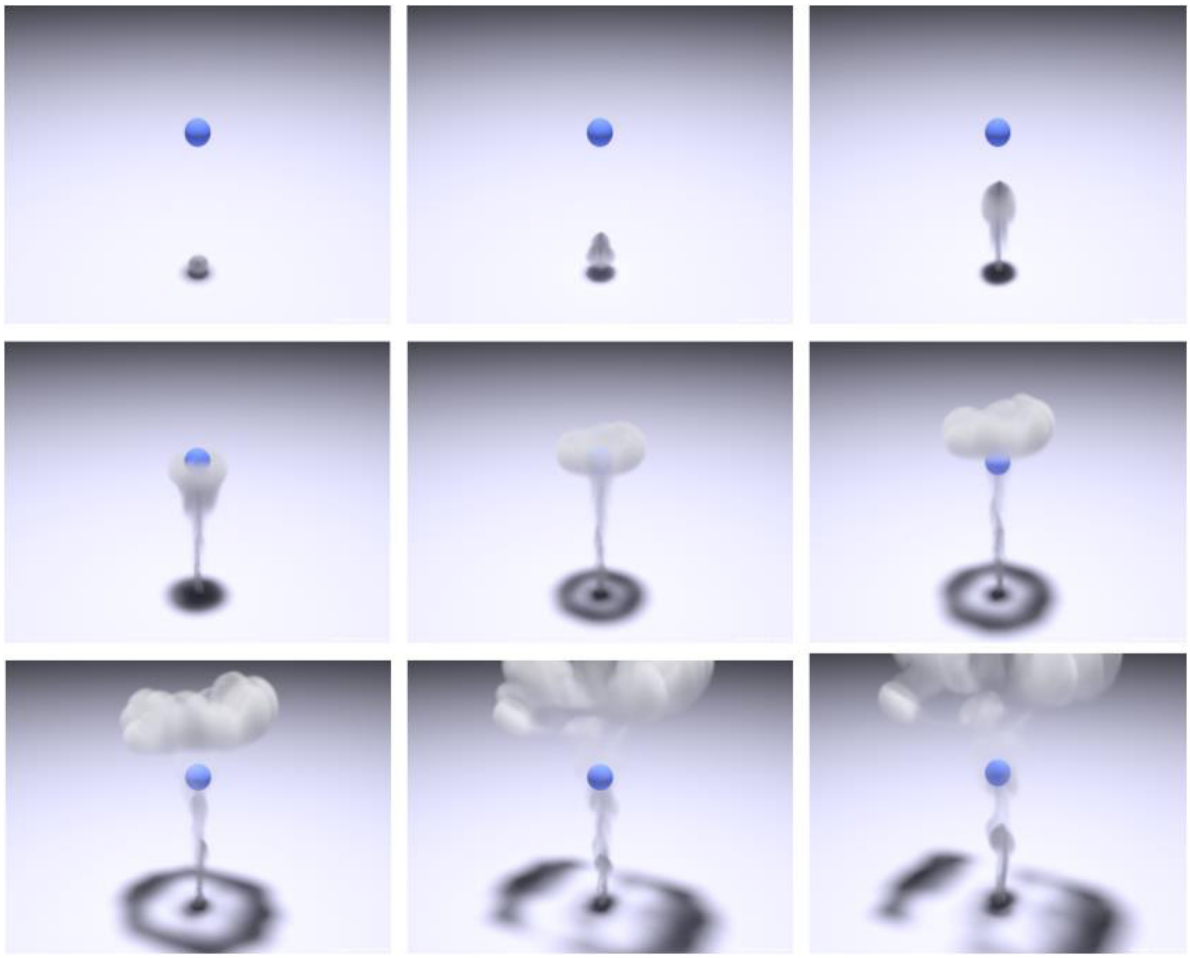


Figure 2.4: Resolution: $128x192x128$, the gas lifted from the bottom and hit the boundary.

gated. For example, Matthias et al. discussed Position Based Dynamics for physics based simulation [14]. Subsequently, Macklin et al. built a unified particle dynamics framework [13] based on PBD. Muller and Matthias introduced impressible fluid simulation by using PBD and compared with classic SPH (Smoothed Particle Hydrodynamics) method [12].

The number of literatures which have been reviewed for this research are far beyond the above articles. The start point of this research was a classic and anonymous

demo based on SPH method which is not included in the following chapters. Additionally, many art albums also can be considered as important literatures of this research.

# Chapter 3

# N-S, LBM, and Turbulence Background

## 3.1 N-S is the Cornerstone

In computer graphics, simulating fluids are usually based on solving N-S PDEs. The classic N-S solvers commonly include advection which describes the inertia of the fluid, diffusion which says the fluid is spreading in all directions, and projection is a step to ensure our fluid is incompressible, which requires to solve the Eq. 3.2 to make the divergence of velocity field is free. Generally, divergence free simply demonstrates the amount of inflow is the same as the amount of outflow, we will discuss specifically in later articles.

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{f} + \nu \nabla \cdot \nabla \vec{u} \tag{3.1}$$

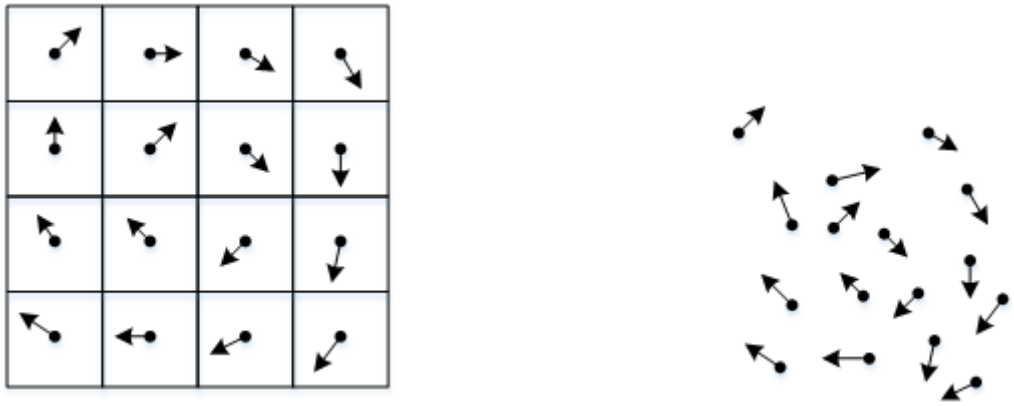
$$\nabla \cdot \vec{u} = 0 \tag{3.2}$$

where $\vec{u}$ is the velocity of fluid, $\rho$ is the fluid density, $p$ represents the fluid pressure, $f$ is the term describes external force such as gravity, buoyancy, etc., and $\nu$ is kinematic viscosity. Fluid motion is updated by the self-advection step which is the second term in the left side of Eq. 3.1. In the advection step, semi-Lagrangian scheme and its modified versions such as BFECC(The Back and Forth Error Compensation and Correction) [8], MacCormack [11], are often perfomed to solve the term. Advection is a terminology showing fluid globe inertia which says the velocity of fluid is moved by itself. From the macroscopic observation, it is difficult to imagine this phenomenon. However, if we consider sand as another type of fluid, we can see intuitively is a systematic motion of the sand under external forces. Zooming in on the motion shows how sand particles collide which helps each other, which leads macroscopic behavior change.

As mentioned in Chapter 1, solving N-S has two main approaches, Lagrangian and Euler view. Accordingly, the fluid properties could be either stored in moving particles or fixed position cells. Please see fig. 3.1.

Note that N-S is orignally non-linear because of the advection term $\vec{u} \cdot \nabla \vec{u}$. This is quite tricky from the Euler view as it is hard to find a solution for efficiently solving non-linear equations. Therefore, in fluid animation, a method commonly called the semi-Lagrangian scheme would be applied. Even though we split the fluid domain into the Euler computational grid, when dealing with advection, particles are still required to trace the velocities (Particles do not have an issue in handling the non-linear problem).

Semi-Lagrangian was introduced to eliminate the problem for solving non-linear equations. Stam's article [20] discussed more specific information about how to im-



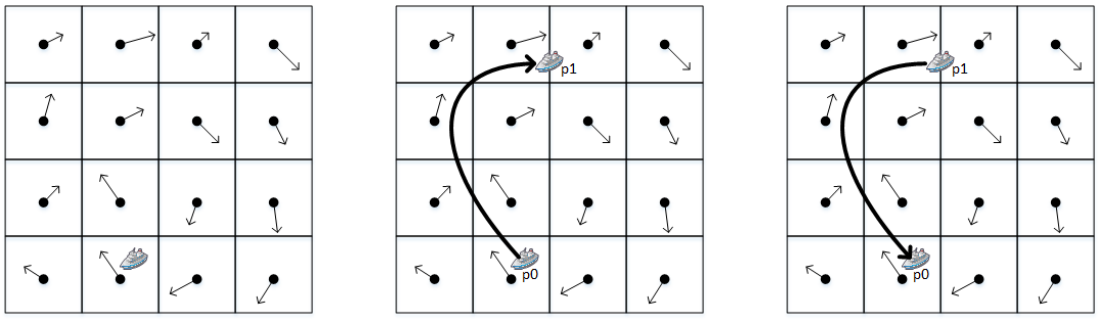
(a) Euler Cells

(b) Lagrangian Particles

Figure 3.1: Euler view (a) stores the fluid properties in each cell. When any material of fluid moves over the grid, these properties will be changed. Lagrangian view (b) keeps the properties of fluid in the particles, given an appropriate collide model such as the smooth function of SPH method, the particles change their properties accordingly.

plement backward tracing for particles. However, it is important to emphasize that there are no particles created visually. Instead, the particles exist conceptually for helping find out the updated velocity of the cell where the particle is coming from. As shown in fig. 3.2, an unpowered tiny boat moves along the ocean current or wind. We expect to find out where the boat is coming from and what velocity it has. This is achieved by backward tracing: $x = i - \delta t\vec{u}_{ij}$, $y = j - \delta t\vec{v}_{ij}$, where $i$ and $j$ are the horizontal and vertical indices to Euler grid cells, respectively. Moreover, the boat does not need to be located at the center of the voxel. If it is located at a non-central area, we should be able to interpolate the coordinates to get a smooth value.

Particles based methods such as SPH are not going to be included in this thesis. We are interested in the Euler grid method. Although LBM borrowed particle methods' concept, grid is still the source of LBM. Before getting into LBM, it is nec-




(a) Initial State (b) Move to p1 (c) Backward Trace

Figure 3.2: A tiny boat does not have engine!

essary to introduce the numerical methods of solving the PDEs and how to deal with incompressible constraint in N-S.

Foremost, deriving a more theoretical view from physics for N-S would be very helpful for understanding later discussions. Start from Newton's second law:

$$\vec{F} = m\vec{a} \tag{3.3}$$

where $\vec{F}$ is the vector, represents force, $m$ is a mass of related object, and $\vec{a}$ is acceleration. Now we would like to extend Newton's Second Law to the fluid motion. Primarily, the forces in the fluid system are usually known as a collection of different sources. They are shown with the expression in Newton's Second Law:

1. **Pressure Force** $\rho\vec{a}_{pressure} = \vec{F}_{pressure}$
2. **Viscosity Force (Dynamic Viscosity)** $\rho\vec{a}_{viscosity} = \vec{F}_{viscosity}$
3. **External Force** $\rho\vec{a}_{external} = \vec{F}_{external}$

Please note the volume has set to a unit. Then we replace the mass $m$ with density $\rho$. External force could be gravity, buoyancy, or user specified force. Moreover, specific

expression of acceleration is the partial differential of velocity (time is not the only variable of velocity function):

$$\vec{a} = \frac{\partial \vec{u}}{\partial t} \tag{3.4}$$

If we look at Eq. 3.1 and simply rearrange the terms, we could have a short one:

$$\rho \frac{\partial \vec{u}}{\partial t} + \rho \vec{u} \cdot \nabla \vec{u} = \sum_{force} \vec{F}_{force} \tag{3.5}$$

where the left side is usually called ***Material Derivative***. It describes that a fluid property is not only changing in time, but also changing in space along the velocity field as well, please see Eq. 3.6:

$$DT = \frac{\partial T}{\partial t} + \vec{u} \nabla T = 0 \tag{3.6}$$

where $T$ is a fluid property, $\vec{u}$ is fluid velocity. So the Eq. 3.5 denotes that the fluid density times the material derivative of fluid velocity should be equal to the summation of all the forces applied on the fluid, see Eq. 3.7.

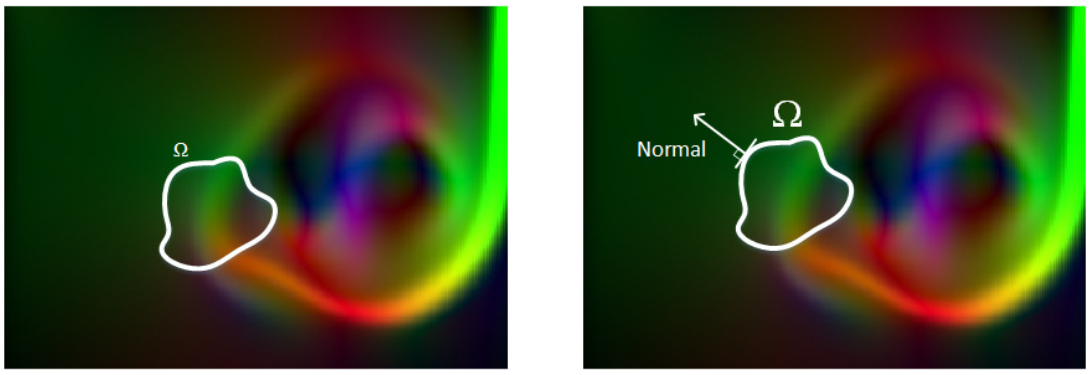
$$\rho \frac{\partial D}{\partial t} = \sum_{force} \vec{F}_{force} \tag{3.7}$$

The subscript $force$ represents the various forces mentioned in the previous context.

Additionally, note Eq. 3.2 is the constraint for incompressible fluid. Before the deriving it, we must clarify the terminology compressible and incompressible. Incompressible fluid is an ideal model to eliminate the difficulty for analyzing the real world fluid. In incompressible fluid simulation, we assume that fluid density is not going to change. As we learned from the basis of physics, in most cases, the density of solid, liquid, and gas is a constant unless external forces are applied to compress them. From the common sense of knowledge, we know that gas can be compressed

with pressure. However, liquid and solid in some cases, such as firing a gun under water are also instantaneously compressible within at a very short period. The water molecules in front of the gun's muzzle gets closer to each other, which is the phenomenon of compressed liquid. Similar cases could be found for compressed solid as well.

Therefore, to make fluid incompressible, in and out of flow to a specific region should be equal. Please refer to fig. 3.3 mathematically, the integral of the boundary




(a) Fluid Boundary $\Omega$ (b) The normal $\vec{n}$

Figure 3.3: Measure the region of flow

$\Omega$ should be zero:

$$\int_{\partial\Omega} \vec{u} \cdot \vec{n} = 0 \tag{3.8}$$

Applying divergence theorem to measure the flux over a randomly curved area:

$$\iint \nabla \cdot \vec{u} = 0 \tag{3.9}$$

Since the whole fluid problem domain follows the same rule, we have:

$$\nabla \cdot \vec{u} = 0 \tag{3.10}$$

which is Eq. 3.2. This is usually called the divergence free constraint for incompressible fluid.

Processing divergence free constraint would have pressure gradients involved. This is usually done by ***Helmholtz-Hodge Decomposition***, which states that any vector field $\vec{u}$ can be decomposed into a divergence free vector field $\vec{u}_{rot}$ and gradients of a scalar field $\nabla q$:

$$\vec{u} = \vec{u}_{rot} + \nabla q \tag{3.11}$$

Commonly, the $\vec{u}_{rot}$ is called ***Solenoidal Vector Field*** which has zero divergence $\nabla \cdot \vec{u}_{rot} = 0$. In addition, $\vec{u}_{rot}$ can also be a curl of a vector potential $\vec{A}$:

$$\vec{u}_{rot} = \nabla \times \vec{A} \tag{3.12}$$

We will use Eq. 3.12 in IVOCK algorithm to deduce the velocity field from vorticity.

During the step for solving pressure, we want to find the divergence free field, therefore, we just switch the terms:

$$\vec{u}_{rot} = \vec{u} - \nabla q \tag{3.13}$$

Then we take $\nabla$ operator for both sides:

$$\nabla \cdot \vec{u}_{rot} = \nabla \cdot \vec{u} - \nabla \cdot \nabla q \tag{3.14}$$

thus, we have:

$$\nabla \cdot \vec{u} = \nabla^2 q \tag{3.15}$$

The process of finding the scalar field $q$ becomes solving ***Poisson Equation***. Once q is solved, the divergence free field can be calculated. The Helmholtz-Hodege Decomposition is a step getting pressure involved which leads to more swirling flows.

Therefore, we rewrite Helmholtz-Hodge Decomposition:

$$\vec{u}^{target} = \vec{u}^{current} - \vec{F}_{pressure} \tag{3.16}$$

where $\vec{u}^{target}$ is the Solenoidal Vector Field which has zero divergence, $\vec{u}^{current}$ is a known term which is computed from Semi-Lagrangian step and $\vec{F}_{pressure}$ is a pressure gradient force $\nabla p$.

Now we have the complete derivation. However, all of these equations in the context are not resolvable so far as they are all based on the real world, in terminology, they are in continuous space and time. So we need to discretize them in some way.

### 3.1.1 Numerical Solution

We already have exposure in the previous paragraph for numerically solving the advection term by using particles. More specific discussion is going to be how does advection work numerically.

Foremost, semi-Lagrangian particles are not created visually. There is no need to visualize these particles, but they do conceptually get involved in the computation for helping find the particle locations. Note that we are still in the Euler view to resolve the problem, hence what we are looking for is the value of the cell. Backwards semi-Lagrangian basically tells us how much velocity we need so that we could push the particle to the current location. If we do forward in time, starting from a grid point, the particles move forward $\vec{x}_{ij} + \vec{u}_{ij}\delta t$, the particles are more likely landing in an non-center point of the cell, then we need to correct their locations onto the grid point again for finding out the transferred energy. Therefore, backwards is more preferable because of its convenience, straightforward, and unconditionally-stable. In

summary, the numerical solution to the advection term uses the semi-Lagrangian method as particles could easily handle the non-linear part. In a 2D situation, assume $\vec{u} = (u, v)$, $i$ and $j$ are horizontal and vertical indices of the computational grid, respectively:

1. $\boldsymbol{x_{real} = i - \delta t \vec{u}^{old}_{i,j}}$, $\boldsymbol{y_{real} = j - \delta t \vec{v}^{old}_{i,j}}$
2. $\boldsymbol{i_0 = truncateError(x)}$, $\boldsymbol{j_0 = truncateError(y)}$
3. $\boldsymbol{k = x_{real} - i0}$, $\boldsymbol{l = y_{real} - j0}$
4. $\boldsymbol{Bilerp(u^{new}_{i,j}, k, l)}$, $\boldsymbol{Bilerp(v^{new}_{i,j}, k, l)}$

where Bilerp denotes the bilinear interpolation of the 2D Euler grid.

Recall that we have divergence free constraint which leads us to solve the pressure (Poisson Equation). Again, by using Newton's Second Law and replacing the acceleration with Eq. 3.4, then we will have the form for pressure gradient force:

$$\rho \frac{\partial \vec{u}}{\partial t} = -\nabla p \tag{3.17}$$

Discretize Eq. 3.17 in time:

$$\vec{u}^n = \vec{u}^* - \frac{\delta t}{\rho} \nabla p \tag{3.18}$$

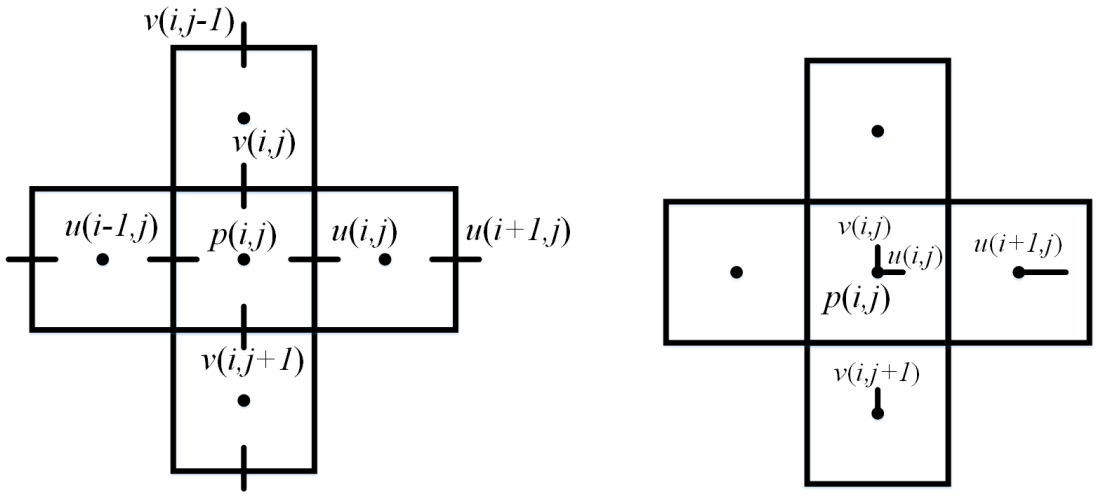
It is necessary to clarify that $\vec{u}^n$ is the target field we expect to ensure the divergence is free. Nevertheless, $\vec{u}^{n-1}$ is not coming from the previous time step, but is a result of advection. Semi-Lagrangian advection is very likely to break the divergence free rule due to the bilinear interpolation or Trilinear interpolation operation and potential constant vector fields.

If we insert Eq. 3.18 into Eq. 3.2, as previously described, this will lead to ***Poisson Equation***:

$$\nabla \cdot \left( \vec{u}^{n-1} - \frac{\delta t}{\rho} \nabla p \right) = 0$$
$$k \nabla^2 p = \nabla \cdot \vec{u}^{n-1} \tag{3.19}$$

where $k$ is a constant that represents $\frac{\delta t}{\rho}$.

We have already discretized in time but have not done for space yet. From the Eq. 3.19, we have ***Laplacian*** operator $\nabla^2$ and divergence of velocity $\nabla \cdot \vec{u}^{n-1}$ which need to be discretized in space. We use the finite difference method to approximate the operator. In most referenced information, staggered grid is commonly used for reducing numerical dissipation as the results of calculation for velocity would match the position where the pressure is located. See (a) of fig. 3.4. If the horizontal index




(a) Staggered Grid (b) Co-Located Grid

Figure 3.4: Two methods for splitting computational domain

of pressure $p$ is $i$, then many related research materials such as Bridson's book [3]

discussing staggered grid used $i-\frac{1}{2}$ and $i+\frac{1}{2}$ to locate the derivative of velocities. This representation can be confusing from the programming point of view, as you cannot use indices $i-\frac{1}{2}$ or $i+\frac{1}{2}$. The trick is to assign different lengths of array for $u$ and $v$, respectively. For example in 2D: $\boldsymbol{u} = [\boldsymbol{N}][\boldsymbol{N+1}]$, $\boldsymbol{v} = [\boldsymbol{N+1}][\boldsymbol{N}]$.

For explanation, however, we prefer to use co-located scheme to avoid confused process of index. Therefore, the components of velocity, pressure, and other properties would have the same size.

The discrete ***Poisson Equation*** would become a matrix-vector form which requires a linear solver to solve the system. Assume that the cell space is $\boldsymbol{h}$, velocity $\vec{\boldsymbol{u}} = (\boldsymbol{u}, \boldsymbol{v})$, and $\boldsymbol{k} = \frac{\boldsymbol{\delta t}}{\boldsymbol{\rho}}$, the discrete form shows as:

$$
\begin{aligned}
k\nabla^2 p &= \nabla \cdot \vec{u}^{n-1} \\
k\Big(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\Big) &= \frac{\partial \vec{u}(u,v)}{\partial x} + \frac{\partial \vec{u}(u,v)}{\partial y} \\
k\left(\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{h^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{h^2}\right) &= \frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{v_{i,j+1} - v_{i,j-1}}{2h} \\
k\left(\frac{4p_{i,j} - p_{i+1,j} - p_{i-1,j} - p_{i,j+1} - p_{i,j-1}}{h^2}\right) &= -\left(\frac{u_{i+1,j} - u_{i-1,j} + v_{i,j+1} - v_{i,j-1}}{2h}\right)
\end{aligned}
\tag{3.20}
$$

Please note we use a multiplier $(-1)$ for both sides. Different linear solvers could be applied to solve pressure $p$. We will present the regarded results in later chapters.

Once pressure has been computed, recall the ***Helmholtz-Hodge Decomposition*** which implies that using $\vec{u}^{n-1}$ to subtract the pressure $p$ will result in a divergence free field which is our destination.

$$
\vec{u}^{rot} = \vec{u}^{n-1} - \nabla p \tag{3.21}
$$

## 3.2 A Mesoscopic Method — LBM

N-S based solver is a downwards method which starts from the macroscopic mathematical description and gradually goes into the details analysis such as pressure solve for incompressible constraint. This usually has the advantage of capturing macroscopic scale motion of fluid but leaves the difficulty to capture small scales. Also, the implementation of N-S based solver requires a certain CFD background such as finite difference, which raises the learning curve from the programming perspective.

If we look at the fluid from microscopic level, we would find that it is exactly a particle system. Actually this is the idea behind the LBM, from the detail to general description for fluid, or in terminology, from microscopic behavior to recover the macroscopic motion. LBM in CFD area is not a brand new topic, but for the computer animation field, it offers a novel perspective and is becoming more popular with the increasing development of parallel device. More practically, it has less code than the N-S solver.

### 3.2.1 Foundation of LBM

LBM is an extension of ***Boltzmann Transport Equation*** as Eq. 3.22.

$$\frac{\partial f}{\partial t} + \vec{u} \cdot \nabla f = \Omega. \tag{3.22}$$

Where $f(\vec{x}, t)$ is the particles distribution function, which collects the possibility that molecules are more likely moving towards. $\vec{u}$ is the molecular velocity, and $\Omega$ is the collision operator which describes the behavior after the molecules collide with each other.

Observing a single molecule's motion does not make sense, since its motion is almost random. However, the overall motion of the fluid is the result of the movements from all of the molecules. Hence, after putting all these chaotic movements together by following the particles' velocity distribution function, their motion becomes more clear and directional. The full implementation of Eq. 3.22 is computaionally expensive. The solution to this problem is to apply lattices for splitting the computational domain and confining the particles onto the nodes of lattice. This turns original microscopic Boltzmann Equation to a new level which is named as Mesoscopic. It does not focus on individual particle, but instead observes behaviors of a group of particles which is commonly called fluid parcels. Each lattice has multiple nodes. There is a term called ***DmQn*** such as ***D2Q9*** which stands for two dimensional and nine possible directions, please refer to fig. 3.5. LBM still keeps sufficient physically-based information. This ensures to achieve the fluids' overall behavior but with acceptable computation cost.

Additionally, the collision term must be a computationally acceptable process.

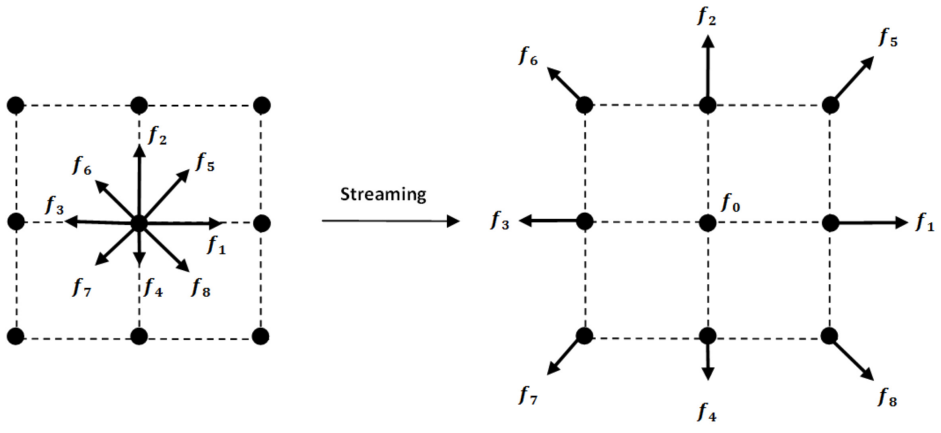



Figure 3.5: Stream the central velocity to 9 directions.

In LBM, Bhatnagar-Gross-Krook (BGK) collision model [2] is usually performed as an efficient model to collide particles. It is motivated by the fact that the original description of collision is based on a pair of particle integral. However, this two-particles model barely influences the macroscopic behavior of fluid states. Consequently, a simpler form for modeling collision step is adopted in a relaxation manner which is named as Maxwellian Distribution.

Boltzmann equation with BGK collision operator is described as Eq. 3.23.

$$\frac{\partial f_i}{\partial t} + c_i \cdot \nabla f_i = \frac{1}{\tau}(f_i^{eq} - f_i), \tag{3.23}$$

where $\tau$ is a parameter to control the frequency of equilibrium distribution, and also is commonly used to control the kinematic viscosity of simulation. Eq. 3.23 is still in a continuous form on the left side, so we need to discretize it in space and time. When choosing an appropriate lattice space $\delta x$ and the time step $\delta t$, the lattice speed is defined as $c_i = \frac{\delta x}{\delta t}$, which allows the properties at location $x$ move to next lattice by $x + c_i \delta t$. Discretize the Eq. 3.23, we get the Lattice BGK form:

$$f_i(\vec{x} + \vec{c}_i \delta t, t + \delta t) - f_i(\vec{x}, t) = \frac{1}{\tau}(f_i^{eq} - f_i). \tag{3.24}$$

where $\tau$ is related to fluid kinematic viscosity $\nu$. In D2Q9 model, it has a form as

$$\tau = \frac{3\delta t \nu}{\delta x} + \frac{1}{2} \tag{3.25}$$

where $\delta t$ and $\delta x$ are time step and cell space, respectively. For the collision process in Eq. 3.24, BGK's equilibrium distribution $f_i^{eq}$ is defined by:

$$\vec{f}_i^{eq} = \omega_i \rho + \rho s_i(\vec{u}^{new}), \tag{3.26}$$

The left side of Eq. 3.24 is commonly named as streaming, while the right side is collision. From an implementation perspective, and on a microscopic level, energy

transfer does not change the macroscopic level velocity directly, however, it gives new value to the lattice on ready-to-update locations. When collision is performed, macroscopic velocity components will be computed by microscopic velocities' summation weighted by $\vec{e}_i$ as Eq. 3.27.

$$\vec{u}(\vec{x},t) = \frac{1}{\rho}\sum_{i=1}^{8}\frac{\delta x}{\delta t} f_i \vec{e}_i. \tag{3.27}$$

It is necessary to note that in the D2Q9 model, the 9 directions are defined as a group of vectors, see Eq. 3.28, and these vectors would be used as a multiplier to calculate the velocity in a specific lattice nodes according to the index as fig. 3.5.

$$\vec{e}_i = \begin{cases} (0,0) & \text{if i=0} \\ (1,0),(0,1)(-1,0),(0,-1) & \text{if i=1,2,3,4} \\ (1,1),(-1,1),(-1,-1),(1,-1) & \text{if i=5,6,7,8} \end{cases} \tag{3.28}$$

Moreover, in Eq. 3.26 $s_i$ is defined as

$$s_i(\vec{u}) = \omega_i\Big[3\frac{\vec{e}_i\cdot\vec{u}}{c} + \frac{9}{2}\frac{(\vec{e}_i\cdot\vec{u})^2}{c^2} - \frac{3}{2}\frac{\vec{u}^2}{c^2}\Big], \tag{3.29}$$

with the weights $\omega_i$ in 9 directions defined by

$$\omega_i = \begin{cases} 4/9 & \text{if i=0} \\ 1/9 & \text{if i=1,2,3,4} \\ 1/36 & \text{if i=5,6,7,8} \end{cases} \tag{3.30}$$

In Eq. 3.26, macroscopic density is defined by the summation from microscopic velocity distribution of location and time.

$$\rho(\vec{x},t) = \sum_{i=0}^{8} f_i(\vec{x},t) \tag{3.31}$$

Regarding the incompressible constraint of the LBM, the computational field is not guaranteed divergence free, however, it automatically matches the incompressible constraint only if the ***Mach Number*** (which is defined by $M = \frac{\vec{U}}{\vec{C}}$, where $\vec{U}$ is the local flow velocity with respect to the boundaries while $\vec{C}$ is the speed of sound in the specific medium) is less than 0.3 which indicates that the change of density is approaching constant. This may turn LBM to be qualitatively equivalent to artificial compressible approaches.

At this stage, all of the backgrounds about fluid simulator we have tested in our experiments have been fully introduced. The remaining of this chapter discusses two main vortex methods that are used, whereas the next chapter presents the details of our algorithms which are the core of our contributions.

## 3.3 Vortex Methods

### 3.3.1 Spinning Wheels

Spinning Wheels is a phenomena that we would visually see in a streamline style visualization from the ***Vorticity Confinement*** method. Please refer to fig. 3.6 and fig. 3.7.

Vorticity confinement is originally proposed by Steinhoff and Underhill [22]. It uses a single parameter to add the swirls (vorticity force) in the fluid domain. When implementing the vorticity confinement in N-S solver, it actually works at a macroscopic level to correct the velocity field. How to combine the vorticity confinement with LBM is discussed next. Our idea is straightforward, treating vorticity force on

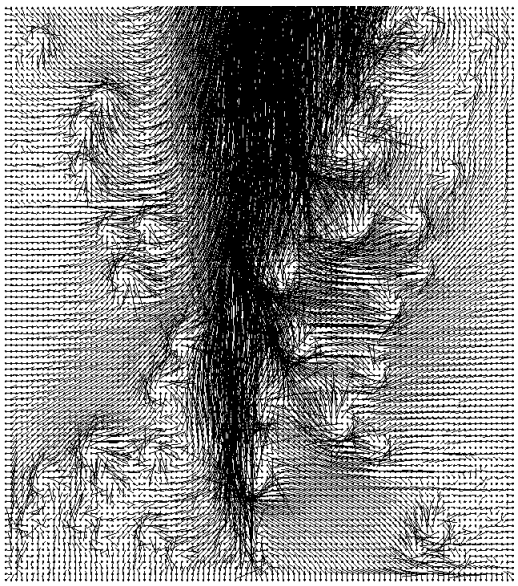


Figure 3.6: The Spinning Wheels appears around the upward flowing fluid

Figure 3.7: Lower value of $\epsilon$ will provide fewer wheels.

macroscopic velocities which is similar to how it works in the N-S solver.

Regarding the algorithm of Vorticity Confinement, firstly, the swirls $\omega$ are computed through performing curls of macroscopic velocites.

$$\omega = \nabla \times \vec{u} \tag{3.32}$$

In the implementation of vorticity computation, we obtain the vorticity on uniform LBM grid through Richardson extrapolation (discrete form):

$$\omega_{i,j} = \frac{1}{12\delta x}(-v_{i+2,j}+8v_{i+1,j}-8v_{i-1,j}+v_{i-2,j})-\frac{1}{12\delta y}(-u_{i,j+2}+8u_{i,j+1}-8u_{u,ij-1}+u_{i,j-2}) \tag{3.33}$$

which has less truncation errors than the other two methods called Stokes Theorem and Least Square. Then vorticity force is defined as Eq. 3.34 with vorticity concentration from low to high: $\vec{N} = \frac{\boldsymbol{\eta}}{|\boldsymbol{\eta}|}$, where ($\boldsymbol{\eta} = \nabla|\omega|$).

$$\vec{f}_{vort} = \epsilon h(N \times \omega), \tag{3.34}$$

Where $\epsilon$ is the vorticity confinement control parameter, usually set between 0 and 1 in N-S solver but between 0 and 0.001 in our algorithms. $h$ is the grid space between two chunks. It is important to be aware of the 2D case when handling the cross product $\frac{\nabla|\omega|}{|\nabla|\omega||} \times \omega$, mathematically speaking, it is undefined. Therefore, we set $\omega$ as the last component of a 3D vector $\vec{vort}^3(0, 0, \omega)$. We need to set $\vec{N}$ components in 3D style $\vec{N}^3(N.x, N.y, 0)$ as well in order to match the cross product definition.

### 3.3.2 More Natural Spinnin Wheels

***IVOCK*** is the other algorithm we are interested in. It is orignated from ***Vortex Dynamics***, and measures the vorticity errors before and after the advection, then velocity correction will be computed from vorticities. The feature of IVOCK is its stand-alone ability that leaves flexibility to change either the boundary process or advection scheme. Compared with the vorticity confinement, IVOCK does not provide parameters to tune the effects. While it generates more reliable results (See fig. 3.8), this could be a double-edged blade from the artists point of view, as it loses controllability but gains reality.

The general algorithm of IVOCK has the steps as Algorithm 1.

**Algorithm 1** IVOCKAdvection($\delta t$, $\vec{u}^n$, $\tilde{\vec{u}}$)

1: $\omega^n \leftarrow \nabla \times \vec{u}^n$
2: $\tilde{\omega} \leftarrow stretch(\omega^n)$
3: $\tilde{\omega} \leftarrow advect(\delta t, \tilde{\omega})$
4: $\tilde{\vec{u}} \leftarrow advect(\delta t, \vec{u}^n)$
5: $\omega^* \leftarrow \nabla \times \tilde{\vec{u}}$
6: $\delta\omega \leftarrow \tilde{\omega} - \omega^*$
7: $\Psi \leftarrow solvePoisson(-\delta\omega)$
8: $\delta\vec{u} \leftarrow \nabla \times \Psi$
9: $\tilde{\vec{u}} \leftarrow \tilde{\vec{u}} + \delta\vec{u}$

The boundary process is left out of the IVOCK scheme as the orignal paper [23] suggested that it should be done in pressure correction step, and IVOCK could be unstable when vorticities are moving around the boundaries and getting values accumulated. Also, ***step 2 and 3*** are summarized by ***advect*** function as IVOCK does not rely on specific advection scheme. This gives us flexibility for our contribution that we will discuss in the next chapter.

Figure 3.8: IVOCK does not have chaotic rotaional wheels in the field but provides natural and reliable shape of the simulation. The support video could be found in the appendix video link.

Please note in step 7 when solving Poisson for stream function $\Psi$, the same

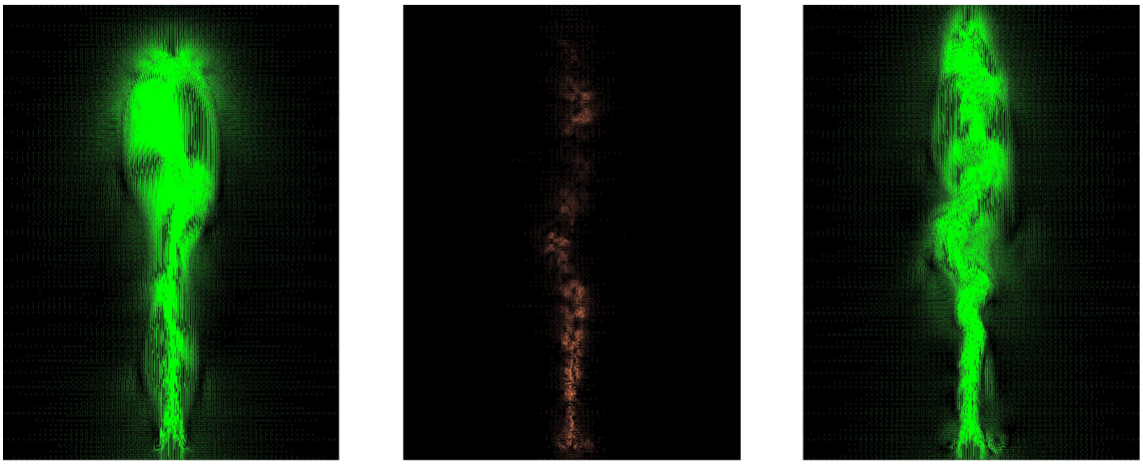


Figure 3.9: **Left:** Velocity field before IVOCK; **Middle:** IVOCK corrections; **Right:** After IVOCK advection

linear solver used for projection step in traditional N-S solver can be reused. We visualize this process by velocity streamlines, please see fig. 3.9. The middle image is the velocity correction, whereas the left figure shows velocity without corrections. When add the corrections back to the field, we get the results in right side figure which shows more complex effects. Also, step 2 will only appear in the 3D situation, because stretch is a procedure that drags vorticies in the direction perpendicular to the plane where the vorticities were generated. In the step of solving the Poisson Equation, an optimized linear solvers is required. According to our experiments on different linear solvers, multigrid solver has the best performance.

# Chapter 4

# Algorithms of LBM and Vortex Methods

## 4.1 Why it is possible to mix

From the experiments we found the possibility to extract vortex methods from N-S based solvers. We propose two algorithms to improve the visual effects of LBM.

Vorticity Confinement and IVOCK are two straightforward methods used in N-S based solvers. Both of them have been proven for their ability to generate sufficient details but with different features. In this study, we mix Vorticity Confinement and IVOCK with LBM, respectively. We found that these two methods are generally working on the macroscopic level of fluids, and hence they can be applied once the macroscopic level velocity has been obtained. Regardless, whether we apply the N-S solver or LBM solver, both of them generate macroscopic velocity at the end, giving the interface where we are ready to apply vortex methods on it. It is necessary to

clarify that our mixed approach have not been proven fully from a physical perspective. However, the algorithms provide rich turbulence which produces strongly visual impact in animation.

Although LBM and N-S analysis start from different points to depict fluid behavior, the intrinsic connection between these two methods can not be ignored — LBM can recover the equation that N-S describes. [4] explained the detail of derivation from LBM to N-S.

## 4.2 How to mix (Algorithms)

According to the classic N-S approach, see fig. 4.1, after the pressure correction step, we eventually will have a divergence free field ready for moving the fluid. Vorticity Confinement usually is applied once the self-advection has been performed, as the macroscopic velocity at this stage is defined. Vorticity Confinement force is treated as an external force in the N-S based solver, and is similar to apply the other forces such as buoyancy and gravity.

On the other hand, IVOCK is slightly different from Vorticity Confinement. It

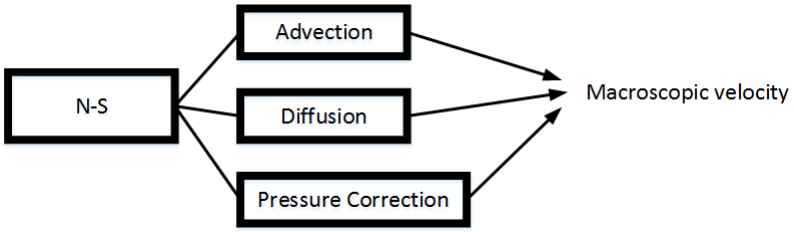



Figure 4.1: General N-S workflow

substitutes the classic solver process for vorticity error measurement. However, it

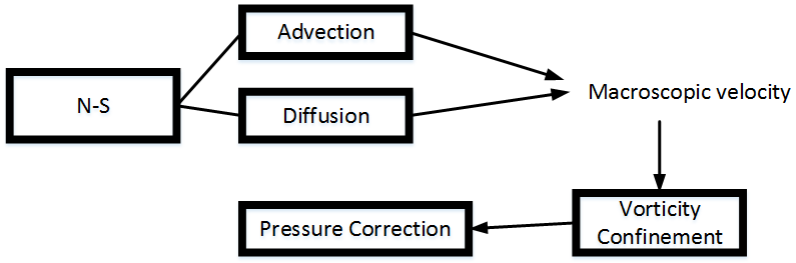



Figure 4.2: Vorticity Confinement is applied when velocity has been obtained.

focuses on the calculations from vortex dynamics, hence leaves space for switching advection schemes and the techniques of processing boundary conditions. Please refer to fig. 4.3 for details of the IVOCK.

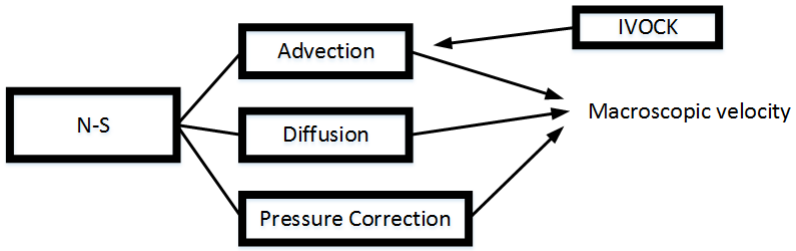



Figure 4.3: IVOCK is applied for changing the advection process

### 4.2.1 Vorticity Confinement LBM (VCLBM)

Vorticity Confinement, as dicussed in the previous chapter, has a single parameter to tune the effects. It can provide a strong and stable visual effects with sufficient swirls when an appropriate $\epsilon$ has been chosen. We summarize the general calculation steps in LBM, and find the entry point where to apply Vorticity Confinement.

Please refer to fig. 4.4 for a general processing of LBM. Note that before applying the equilibrium distribution calculation, the velocity actually has been obtained.

This velocity is used for updating the distribution funciton and then the microscopic velocity at specific directions will be replaced with new values. If we quickly review how vorticity confinement works, we would find that curl is calculated from the macroscopic velocity, and the N-S based solver uses the velocity to advect other fluid properties, hence it is the entry that we are looking for.

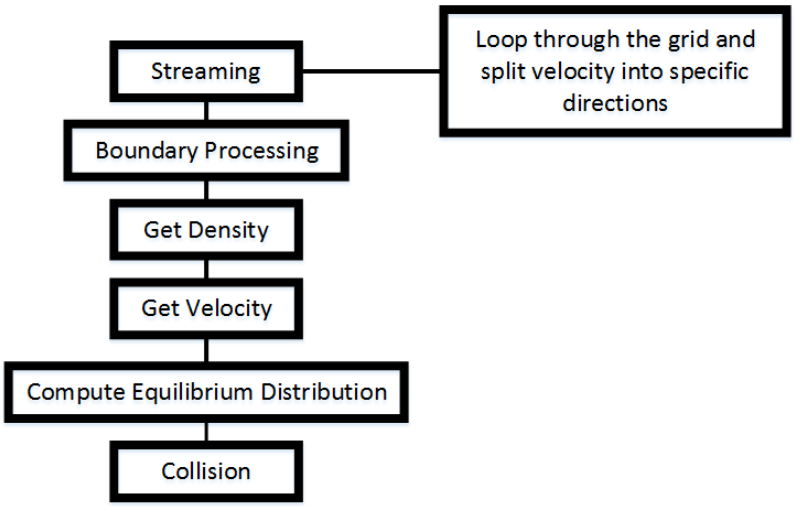



Figure 4.4: General processing steps of LBM

After we obtain the intermediate velocity from vorticity confinement calculations, the equilibrium distribution should follow the intermediate velocity to update its values.

A more specific process of VCLBM is stated by Algorithm 2.

In step 4, Eq. 3.28 is implemented. Thus the density $\rho$ has been split into $x$ and $y$ components. These two components will be used for the velocity components calculation at step 5, respectively. Moreover, $f_i$ represents the velocity distributions, as e.g. in D2Q9, where $f_i$ is the distribution of 9 individual directional velocities from

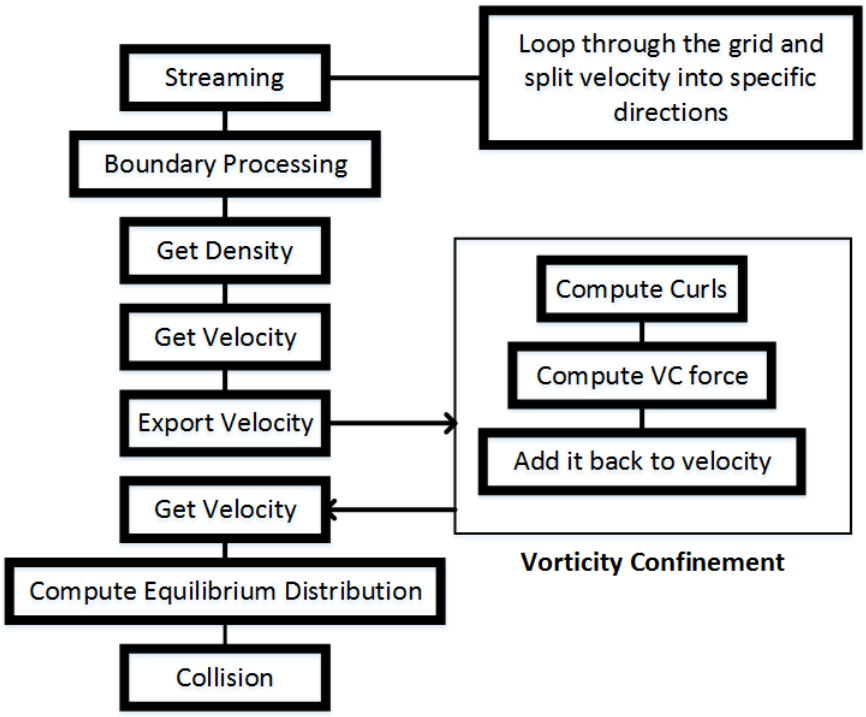



Figure 4.5: Velocity has been exported for curl VC processing

**Algorithm 2** VCLBM Workflow 2D (D2Q9)

1: $f_i = stream(\vec{u}^{n-1})$
2: $applyBCs(f_i)$
3: $\rho = sum(f_i)$
4: $(\rho_x, \rho_y) = sum(\rho \cdot \vec{e}_i)$
5: $\vec{u}_x^{n-1} = \frac{\rho_x}{\rho}; \quad \vec{u}_y^{n-1} = \frac{\rho_y}{\rho}$
6: $\bar{\vec{u}} = export(\vec{u}_x^{n-1}, \vec{u}_y^{n-1})$
7: $\omega = computeCurl(\bar{\vec{u}})$
8: $\vec{f}_{vort} = computeVortForce(\omega)$
9: $\bar{\vec{u}} = \bar{\vec{u}} + \vec{f}_{vort} \cdot \delta t$
10: $\vec{u}^{n-1} = import(\bar{\vec{u}})$
11: $computeDistribution(\vec{u}^{n-1})$
12: $\vec{u}^{n} = collideBGK()$

streaming.

### 4.2.2 IVOCK-LBM

We have illustrated the original IVOCK scheme in Algorithm 1 and we have discussed its abilities to stand-alone from other advection methods. Thus, the target is straightforward — replace the advection part with a LBM approach. However, IVOCK costs more computational resources than VCLBM due to the process of Poisson Equation Solving for deducing the velocity corrections. This step gets a linear solver involved. Hence choosing an appropriate linear solver should be carefully considered.

Recalling the IVOCK general steps shown in Algorithm 1, the calculation of vorticity is the first step and requires at least macroscopic velocity to be obtained. Therefore, we still need to go through the streaming process and get individual velocities, also similar to the situation before step 6 in Algorithm 2. The general workflow of IVOCK-LBM is shown in fig. 4.6. The initial state of IVOCK-LBM requires storing of velocity which has not been solved yet. Thus, Algorithm 3 starts by exporting the previous solved time step velocity. It then takes two seperated paths: advect velocity and advect vorticity. Similar to the original IVOCK, we also have not fixed the advection scheme of vorticity, while we fixed the velocity advection based on LBM. Vorticity advection is supposed to adapt to either a semi-Lagrangian solver or a LBM solver. After velocity has been solved, we take the curl($\nabla$) of it again. We then measure the differences from these two paths, and get ready to solve the Poisson Equation for deducing the velocity corrections. Subsequently, like VCLBM, we added corrections back (which should not be scaled by time step, as it is not a force) to the velocity

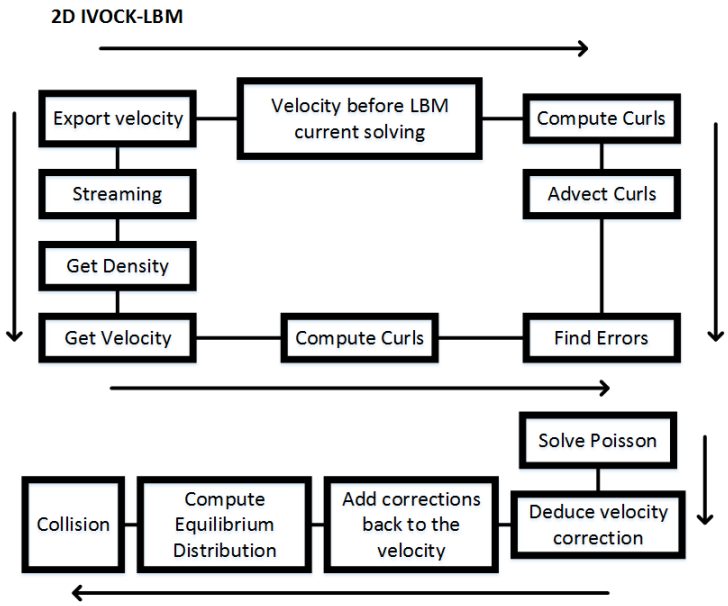



Figure 4.6: Workflow of IVOCK-LBM

field. Then, we used corrected velocity field to update equilibrium distributions.

It is important to note that initialization is required for velocity distributions $f_i$. Hence we could obtain the vorticity through velocity field , which is computed from Eq. 3.27

## 4.3 Implementation Discussion

We setup an implementation environment with very few third-party-libraries in order to help later researcheres to simply reproduce the experiments. We choose $\boldsymbol{C/C++}$ as the programming language which is very efficient for numerical solutions. On the other hand, visualization only relies on ***OpenGL*** with its extension ***GLUT***. In the results of our experiments, we generate sequential images to illustrate the compar-

**Algorithm 3** IVOCK-LBM Workflow 2D (D2Q9)

1: $(\rho_x, \rho_y) = sum(\rho \cdot \vec{e}_i)$
2: $\vec{u}_x^{n-1} = \frac{\rho_x}{\rho}; \quad \vec{u}_y^{n-1} = \frac{\rho_y}{\rho}$
3: $\omega = \nabla \times \vec{u}^{n-1}$
4: $\bar{\omega} = advect(\omega)$
5: $f_i = stream(\vec{u}^{n-1})$
6: $applyBCs(f_i)$
7: $\rho = sum(f_i)$
8: $(\rho_x, \rho_y) = sum(\rho \cdot \vec{e}_i)$
9: $\vec{u}_x^{n} = \frac{\rho_x}{\rho}; \quad \vec{u}_y^{n} = \frac{\rho_y}{\rho}$
10: $\bar{\vec{u}} = export(\vec{u}_x^n, \vec{u}_y^n)$
11: $\omega^* = \nabla \times \bar{\vec{u}}$
12: $\delta\omega = \bar{\omega} - \omega^*$
13: $\Psi = SolvePoisson(\delta\omega)$
14: $\bar{\vec{u}} = \bar{\vec{u}} + \nabla \times \Psi$
15: $\vec{u}^n = import(\bar{\vec{u}})$
16: $computeDistribution(\vec{u}^n)$
17: $\vec{u}^{n+1} = collideBGK()$

sions with different parameters and different methods in both LBM and N-S.

Moreover, most of our experiments are two dimensional and we have not presented the three dimensional situation yet. Nevertheless, the implementation can be easily extended to three dimensions by adding another component to the vector properties, but it could be cumbersome when debugging for numerical issues.

So far, we have fully introduced all of the algorithm and the foundation of simulation method. In the final chapter, we are heading to demonstrate the results gained from our experiments.

# Chapter 5

# Conclusion and Results

In this chapter, we present both 2D and 3D results from our experiments. Besides LBM results, we also present the original N-S based solver with the two vortex methods we used for LBM. Moreover, the corresponding video clips can be found in the support video link (See Appendix A).

A common question in this research has always been "How would you convince people that the results are more natural or interesting than the previous works". The answer is to find a similar and convincible test samples in CFD field or even in real life. It is difficult to produce a real world experiment and compare it with the simulations. This is because in the real world, parameters that vary over time will be much more complex than in these simulations, and the patterns of fluid is challenging to fully match the simulations that only have a few impacting factors. On the other hand, regarding the visual effects in the film or game industries, a fully accurate simulation is not the primary goal, whereas lower consumption of computation resources, more controllable and more interesting become the foremost factors to consider.

We borrow the experiment which is commonly used to test new code or new features in CFD field to examine our approaches. It is also a real world experiment named as ***Lid Driven Cavity Flow*** (LDCF) which could be seen as a ground truth, please refer to fig. 5.1.

Basically, LDCF assumes that a flow passes by the top of a container without

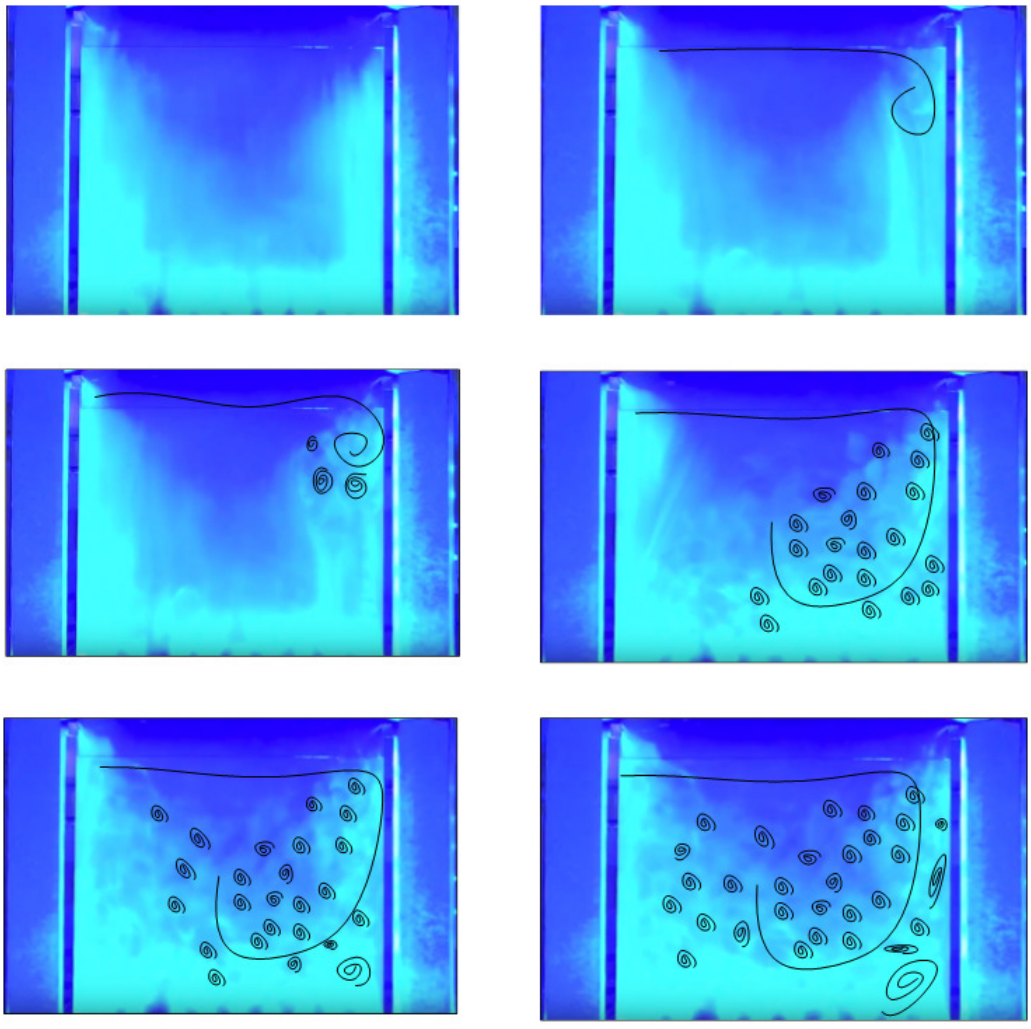


Figure 5.1: A real experimental lid driven cavity flow, and the detail curls are marked with black curves. (The video could be found through supporting video sites; see Appendix A)

top lid, which has all others faces (boundaries) with non-slip conditions. As time goes by, a special pattern will be generated. Specifically, a general big curl will be created at the center of the container over time. Moreover, details can be found later on at the bottom-right corner that small curls have been generated. In the real experiment of LDCF, we can see that the turbulence is very complicated and countless curls are produced around the high fluid concentration.

We reproduced the LDCF experiment in a LBM solver with the two differnt vortex methods. Firstly, a LBM solver with no vortex method is shown in fig. 5.2. The simulation is three dimensional and the resolution is $32 \times 32 \times 32$, with the velocity generated at the top in only horizontal direction from left to right.

As shown in fig. 5.2, the general shape of LDCF can be captured. However, as the resolution is low, the detail flow is hard to be obtained. Then, when we raise the resolution of the simulation to $128 \times 128 \times 128$, more interesting and complex pattern appear, as shown in fig. 5.3.

Compared to the real experiment, the general pattern of the Lid Driven Cavity Flow has been obtained through our simulation. We can logically conjecture that with the resolution increased, more details will be captured and the simulation is getting closer to the real experimental results. However, vortex method requires more computational cost and memory, which lead to unacceptable time consumption.

By adding two vortex methods to the LBM approach, we get results as shown in fig. 5.4 with resolution of $32 \times 32 \times 32$, and fig. 5.5, with a resolution of $128 \times 128 \times 128$. When the resolution increases, turbulence effects become more clear. However, if we carefully observe fig. 5.5, we can see stripe texture pattern. This indicates that the simulation has a potential to blow-up and if we raise the $\epsilon$ of Vorticity Confinement to

0.01 or higher, the simulation would crash. Crashes often happen when inappropriate parameters have been chosen.

We tested three different methods to compute curls and get different visual results. From summary, please see Table 5.1.

| Method | VCLBM | IVOCK-LBM |
|---|---|---|
| Stokes Theorem | Minor curls | Does not work |
| Least Square | Sharp curls, unattached | Sharp curls |
| Richardson | Smooth curls | Smooth curls |

Table 5.1: Evaluation for the effects of swirls based on different methods of curl calculations

The time consuming statistics for different resolutions are shown in Table 5.2

| Resolution | LBM | VCLBM | IVOCK-LBM |
|---|---|---|---|
| $64 \times 64 \times 64$ | $0.08(s)$ | $0.11(s)$ | $0.3(s)$ |
| $96 \times 96 \times 96$ | $0.1(s)$ | $0.2(s)$ | $0.37(s)$ |
| $192 \times 192 \times 192$ | $0.7(s)$ | $1.0(s)$ | $2.1(s)$ |

Table 5.2: 3D Performance of algorithms in different resolution for each step

Additionally, we also tested our algorithm through another very popular experiment which says that the fluid pass through a tilted boundary. For instance, the flow passes over an airfoil. See fig. 5.7. If we look at the third frame of both sequences, it is very clear that VCLBM generates more turbulence than original method. As

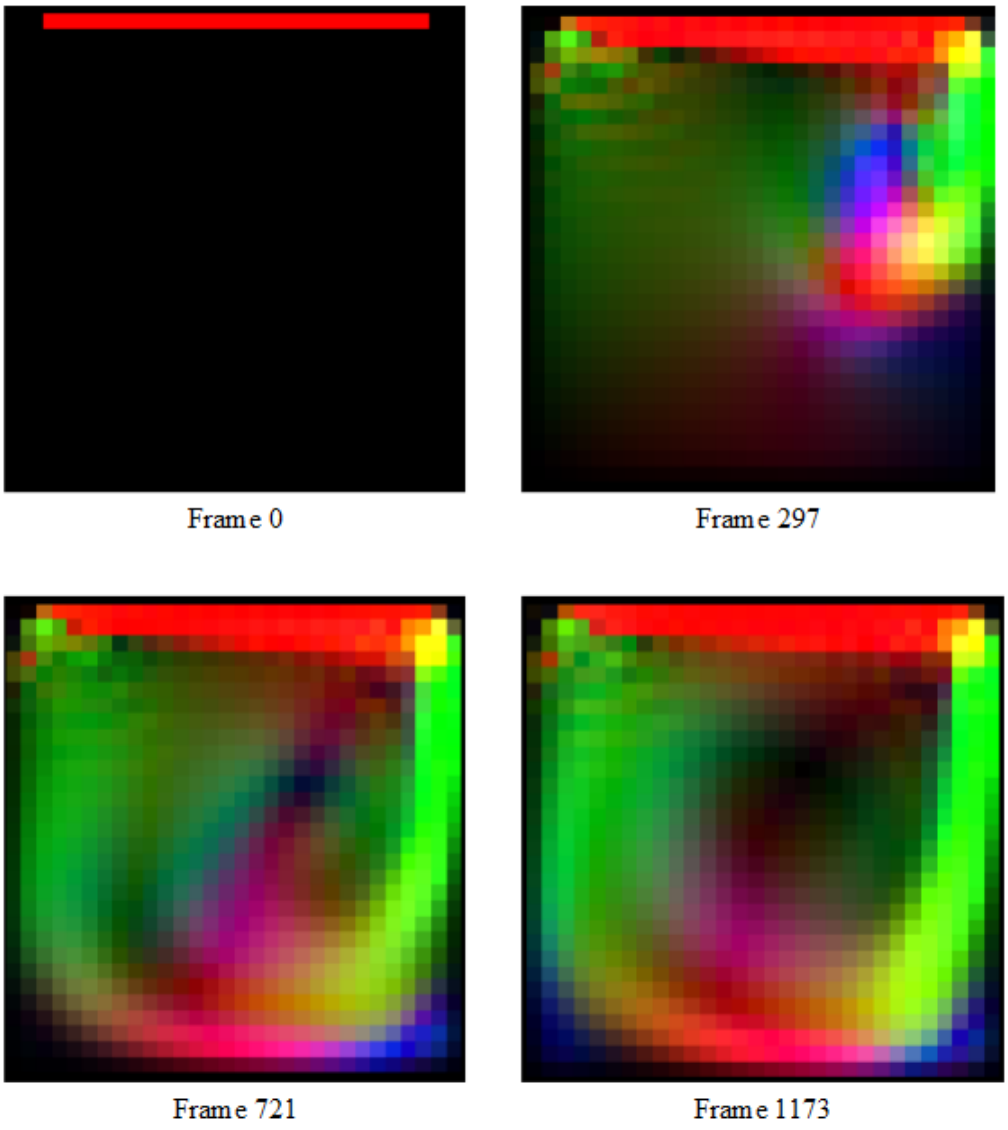



Figure 5.2: Resolution: $32 \times 32 \times 32$, Lid Driven Cavity Flow without vortex methods. It is quite rough to observe the detail of flow

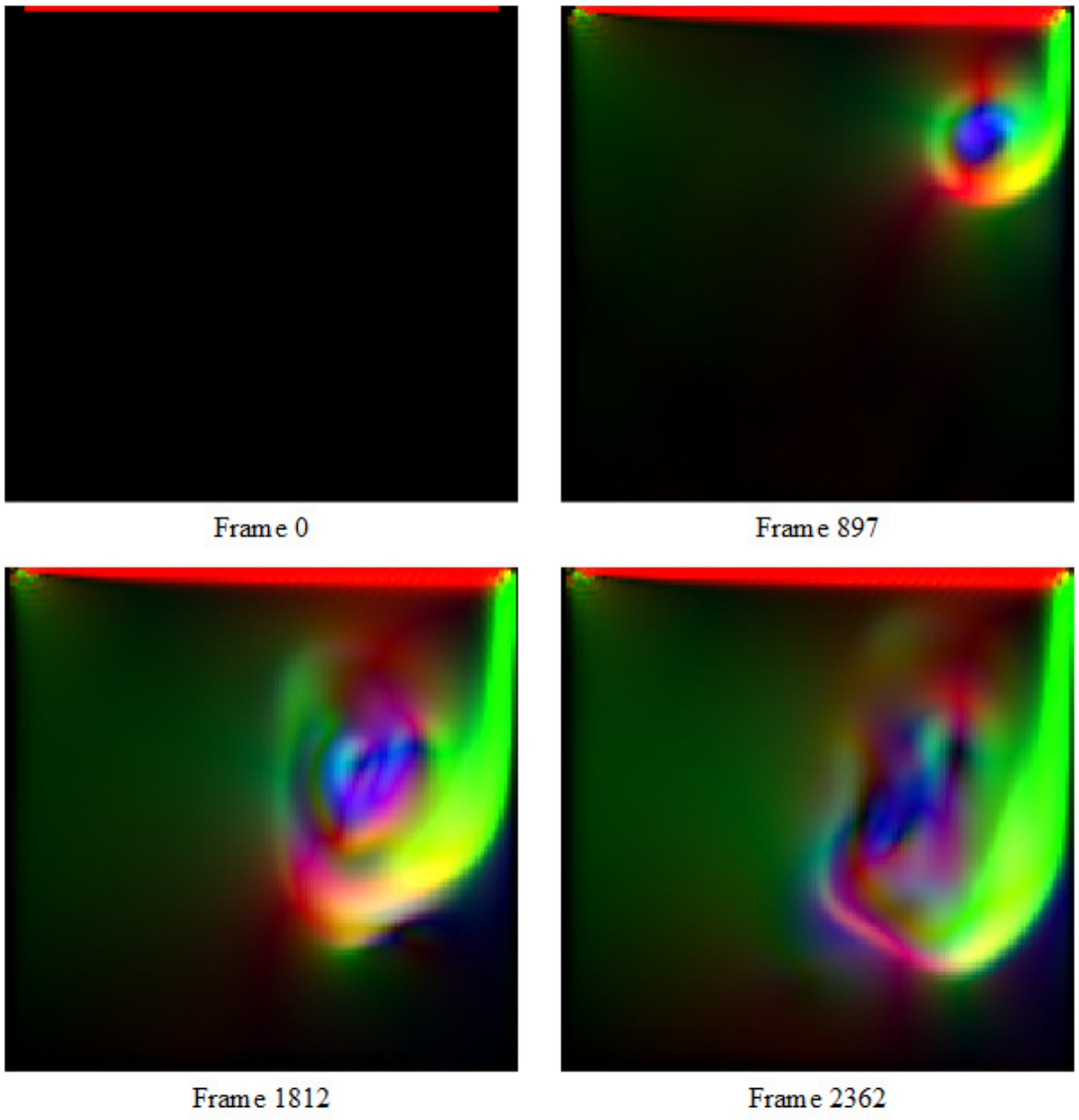



Figure 5.3: Resolution: $128 \times 128 \times 128$, Lid Driven Cavity Flow without vortex methods.By increasing the resolution, more irregular and some turbulence effects could be found around the central rotational part.

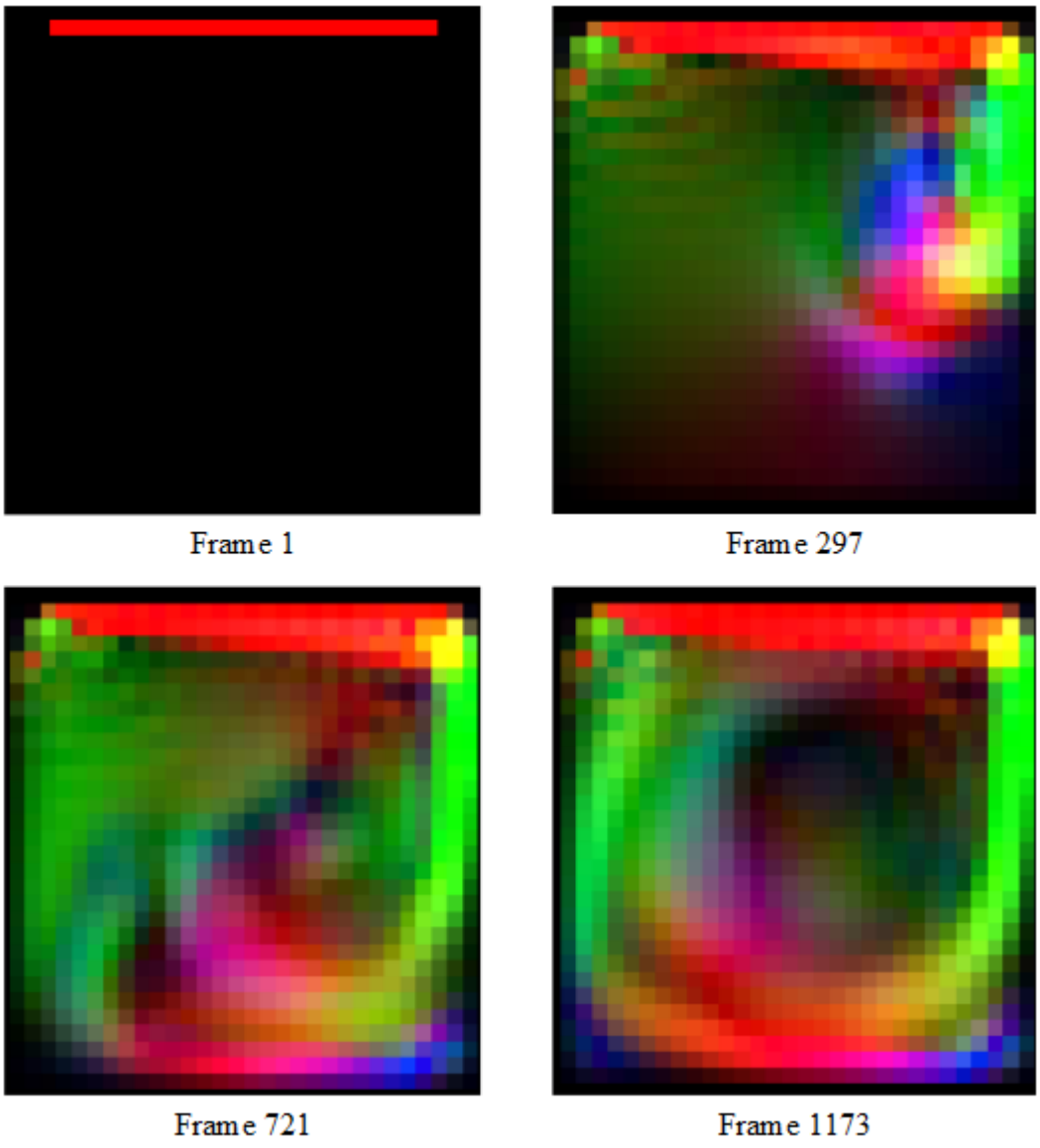



Figure 5.4: Resolution: $32 \times 32 \times 32$, Lid Driven Cavity Flow with IVOCK and Vorticity Confinement update. Also, the low resolution could give very limited details. However, potential turbulence could be found around the central rotational area.

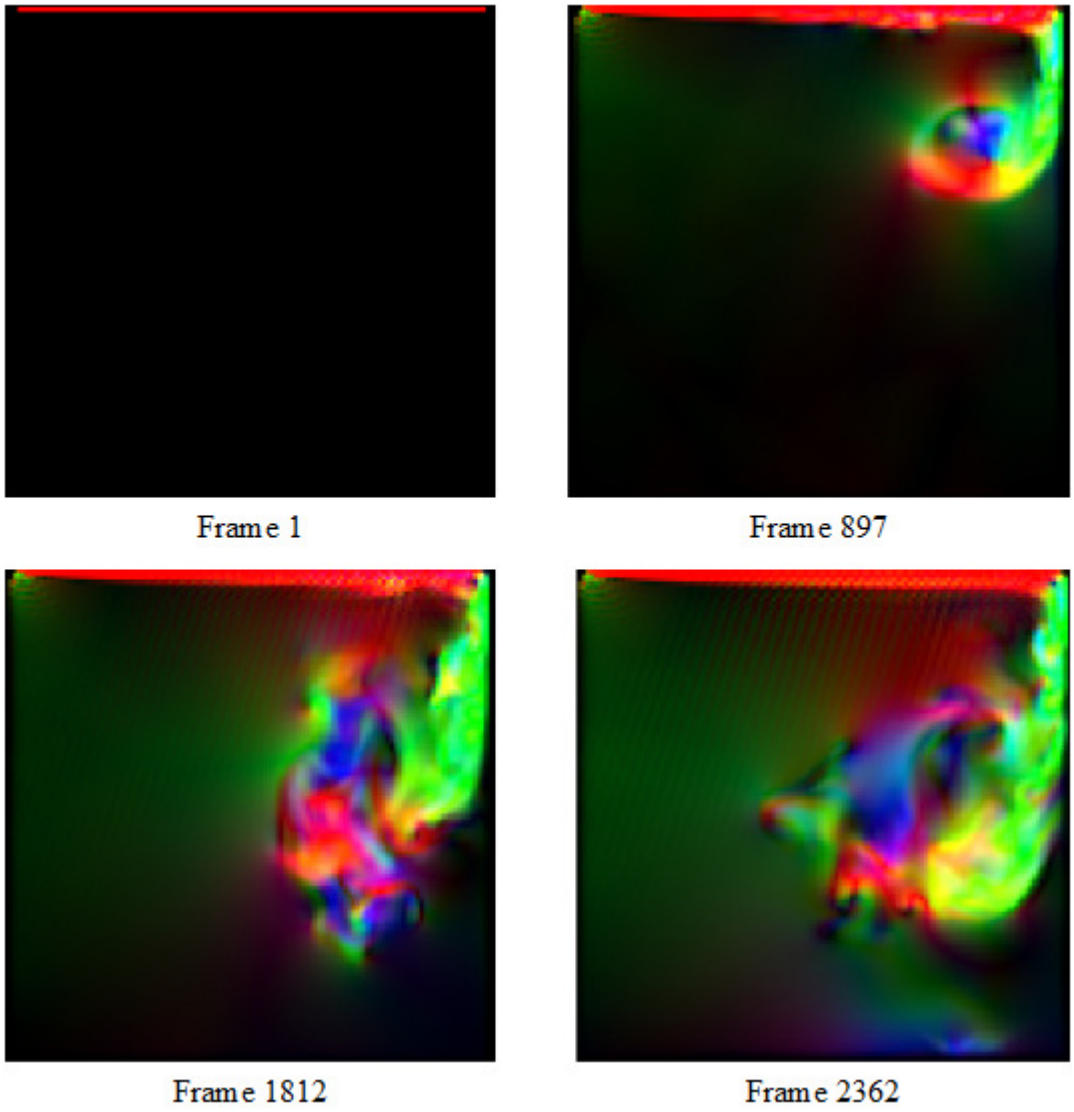



Figure 5.5: Resolution: $128 \times 128 \times 128$, Lid Driven Cavity Flow with IVOCK and Vorticity Confinement update. With a high resolution, turbulence could be simply captured.

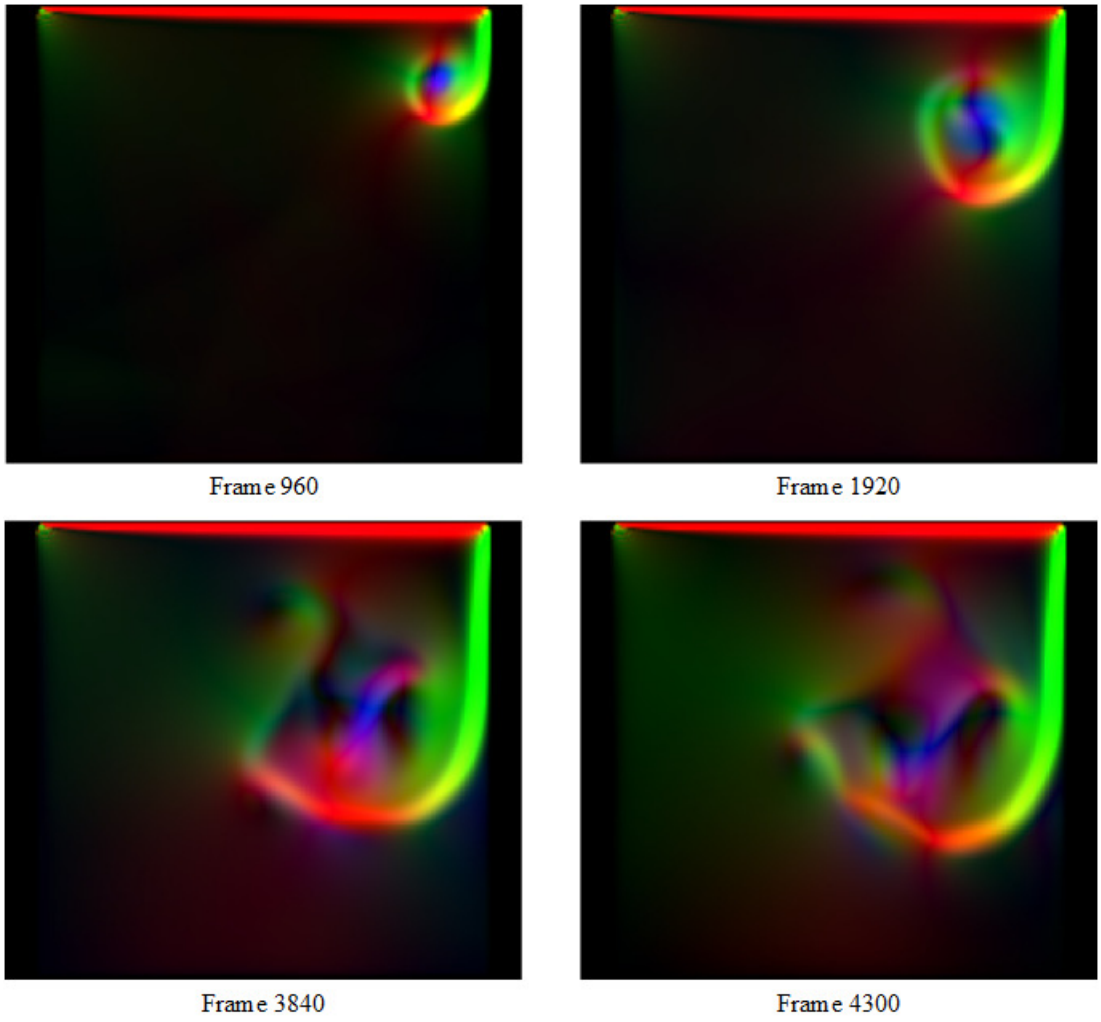



Figure 5.6: Resolution: $128 \times 128 \times 128$, Lid Driven Cavity Flow with Vorticity Confinement only. The parameter $\epsilon$ is set to 0.008.

we have mentioned that LBM is sensitive to parameters, by inserting our algorithm to LBM is not a perfect solution. The last frame in VCLBM of fig. 5.7 has shown potential unstable patterns which are some repetitive textures.

It is necessary to clarify that from fig. 5.2 to fig. 5.6 are 3D simulation but are visualized in 2D orthogonal mode. We render every 20 simulation steps into the disc and the movie player should be set to 480 frames per second for fluent animation.

Recall that we are required to solve a ***Poisson Equation*** to get vector potential

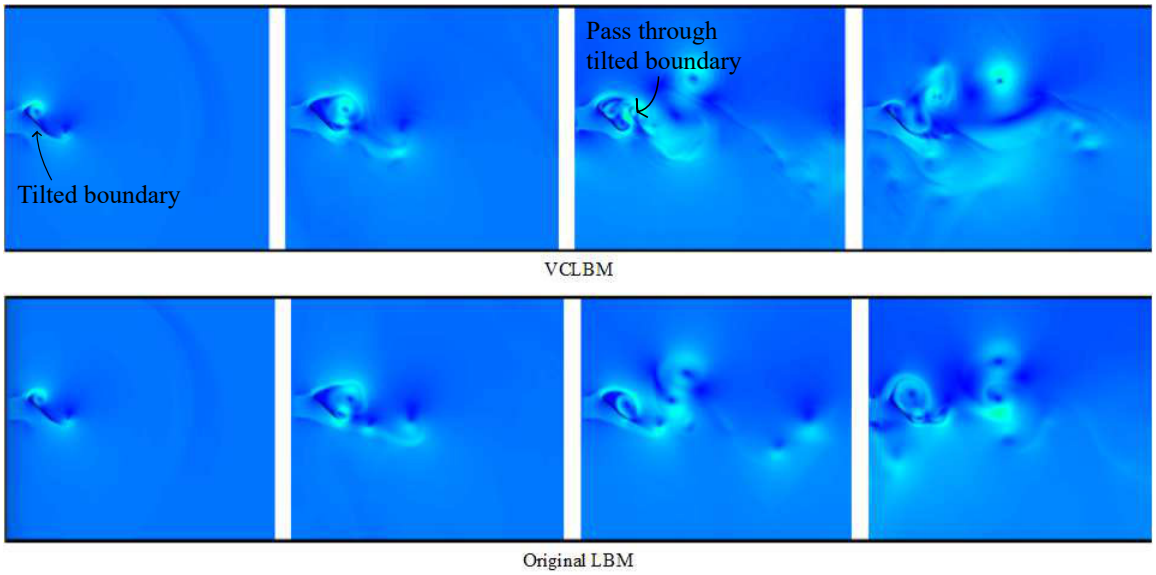



Figure 5.7: Flow passes over a black tilted boundary. Carefully observe the third frame for both original LBM and VCLBM, the VCLBM has more chaotic patterns appearing after the black panel.

$\Psi$, and then compute the curl of $\Psi$ to get velocity corrections, please refer to Algorithm 1. We examined Gauss Seidel, Red-Black Gauss Seidel, Jacobi and Multigrid linear solvers for solving the Poisson Equation, however, we did not get much visual

difference from these different solvers. On the other hand, from the computational speed point of view, only multigrid method illustrates faster solving while all the other three methods speed are similar.

We developed most of experiments based on our own code base which we build from scratch. Moreover, we also tested our approach by inserting the vortex algorithms to exsiting LBM code. After these experiments for both N-S and LBM, we made following conclusions:

- **Relationships between N-S and LBM** N-S describes fluid based on Newton's second law, while LBM is derived from statistical mechanics. We would like to call LBM as a Semi-Lagrangian Method as we analyze fluid states from particles' view but a grid is also required to confine and to align the particle's velocities. Although LBM is derived from a totally differnt view, N-S equations could be recovered from LBM, we provide the reference describing how N-S is recovered by LBM [4].

- **LBM — Easy To Implement** Intuitively, the number of lines in code for LBM is less than N-S solvers. This can be well explained by its simple equation form, it requires less partial different equation solutions to solve the term separately.

- **LBM — Sensitive to Parameters** For each experiment shown in the figures of this chapter, we carefully tune the parameters of the simulation to have the best results visually. We found LBM heavily depends on an appropriate set of parameters.

- **Turbulence Effects of LBM** In this thesis we try to combine the two vortex

methods and LBM. It works visually correct and returns us rich turbulence effects. As we have dicussed in chapter one, turbulence effects is a very important visual effects in film and game industry giving audiences strong visual impact. However, when applying these vortex methods on LBM, we still cannot avoid to carefully tune the parameters to keep simulation stable. We believe blow-up happens because we have not specifically processed the boundary values where the curls are moving around. Furthermore, it is tedious to define how curls are stored in LBM computational grid.

# Chapter 6

# Future Work

In this research, we analyzed many experiments of fluid animation and vortex methods. We picked a classic vortex method and a most recent vortex method to test LBM. They have shown the ability to enrich turbulence with only minor impacts in simulation stability. We believe LBM is a very efficient approach to be an alternative choice for fluid animation. For example, in a widely-used open-source 3D-Modeling software — Blender, liquid solver is desgined on in LBM, whereas the smoke solver is still based on in N-S. It could be explained by the fact that liquid animation requires more computational cost for surface rendering which is not included in this thesis, but it is also a crucial topic for fluid animations.

Our approach is based on a general view of macroscopic level for enriching the complexity of visual effects for fluids. We argue that the algorithm VCLBM and IVOCK-LBM are straightforward ways to modify original approach. Even though the view of simulation is essentially different from N-S, we focus on a general vision to observe the states change. It brings us the key ideas that insert vortex methods

to LBM on macroscopic velocity, but leave the mesocopic level as LBM's original procedure. As an alternative option, LBM has its own style of adding external force, which would introduce one more term in the corresponding equations. We encourage readers taking our overall procedures as an introduction to examine other vortex methods. In addition, collision operator is limited to BGK in this thesis. Many other approaches could be performed, which are also likely to provide turbulence effects. A good introduction to this topic could be find in Aslan, Taymaz and Benim's work [1].

Our approach does not specifically address boundary conditions. This usually leads the solver to blow-up and potential dissipation. In our algorithms, the boundary condition of curls retains to the original position where we process the velocity field boundaries. Non-sense value could appear at the location of boundary points when vorticity related calculations are performed. This leaves a room for more investigations on how to keep the simulation stable over time.

From the view of physics, LBM is growing to be a more popular choice of fluid simulation which solves problems in a modern physical point of view. It is derived from statistical mechanics, but differs from absolute microscopic. With an appropriate modification, turning the idea to the LBM, it brings countless benefits in the aspect of fluid animation in computer graphics area such as parallel processing. Moreover, turbulence modeling have always been attracting researchers in CFD community. It is very difficult to solve such a mystery motion perfectly, but many interesting models have been proposed in past decades. We encourage readers giving these approaches a try with LBM. Finally, regarding LBM itself, it still remains the common shortcome of producing high speed flows. One proposed approach so far is to investigate the MRT (Multi-Relaxation-Time) for LBM [6]. MRT extends the stability of conven-

tional LBM and allows higher speed of flow which we could expect more swirling effects.

In addition to the methods we mentioned above, Single Phase Screen (SLM) which has been studied in atmospheric turbulence field is also a possible solution to generate turbulent effects as Implicit Large Eddy Simulation (ILES) does.

# Bibliography


[1] E. Aslan, I. Taymaz, and A. Benim. Investigation of the lattice boltzmann srt and mrt stability for lid driven cavity flow. *Intl. J. Mater. Mech. Manuf*, 2(4):317–324, 2014.

[2] 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.*, 94:511–525, May 1954.

[3] R. Bridson. *Fluid simulation for computer graphics*. CRC Press, 2015.

[4] S. Chen and G. D. Doolen. Lattice boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.

[5] A. J. Chorin, J. E. Marsden, and J. E. Marsden. *A mathematical introduction to fluid mechanics*, volume 3. Springer, 1990.

[6] D. d'Humières. Multiple–relaxation–time lattice boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.

[7] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 2001.

[8] B. Kim, Y. Liu, I. Llamas, and J. R. Rossignac. Flowfixer: Using bfecc for fluid simulation. Technical report, Georgia Institute of Technology, 2005.

[9] T. Kim, J. Tessendorf, and N. Thürey. Closest point turbulence for liquid surfaces. *ACM Transactions on Graphics (TOG)*, 32(2):15, 2013.

[10] T. Kim, N. Thürey, D. James, and M. Gross. Wavelet turbulence for fluid simulation. In *ACM Transactions on Graphics (TOG)*, volume 27, page 50. ACM, 2008.

[11] R. MacCormack. The effect of viscosity in hypervelocity impact cratering. *Journal of spacecraft and rockets*, 40(5):757–763, 2003.

[12] M. Macklin and M. Müller. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):104, 2013.

[13] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4):153, 2014.

[14] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[15] J. Onderik and R. Durikovic. Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics, Statistics and Informatics (JAMSI)*, 4(1):29–43, 2008.

[16] T. Pfaff, N. Thuerey, J. Cohen, S. Tariq, and M. Gross. Scalable fluid simulation using anisotropic turbulence particles. *ACM Transactions on Graphics (TOG)*, 29(6):174, 2010.

[17] M. Popova. The fluid dynamics of the starry night: How vincent van goghs masterpiece explains the scientific mysteries of movement and light, November 2014. [Online; posted 13-November-2014].

[18] C. D. Scott-Pomerantz. *The k-epsilon model in the theory of turbulence*. PhD thesis, University of Pittsburgh, 2005.

[19] J. Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[20] J. Stam. Real-time fluid dynamics for games. In *Proceedings of the game developer conference*, volume 18, page 25, 2003.

[21] J. Stam. *The Arts of Fluid Animation*. A K Peters/CRC Press, 2015.

[22] J. Steinhoff and D. Underhill. Modification of the euler equations for vorticity confinement: application to the computation of interacting vortex rings. *Physics of Fluids (1994-present)*, 6(8):2738–2744, 1994.

[23] X. Zhang, R. Bridson, and C. Greif. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics (TOG)*, 34(4):52, 2015.

[24] Y. Zhao, F. Qiu, Z. Fan, and A. Kaufman. Flow simulation with locally-refined lbm. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 181–188. ACM, 2007.

# Appendix A

# Demo and Code

The results include the sequences of images for LBM and Turbulence Effects of LBM. We presented multiple dimensions of the simulation to illustrate the details captured from the solver. Also, the experiments mentioned in this thesis could be found through author's personal YouTube Channel as following link: `https://www.youtube.com/playlist?list=PL6dfRoj3VxD8SkcIjxSp-SmTqkUYhQMo_`

Part of experiments code are located at the braches named as "Experiments" and "VFX":`https://github.com/Shakebones?tab=repositories`