FINITE VOLUME DISCRETE BOLTZMANN METHOD ON A CELL-CENTERED TRIANGULAR UNSTRUCTURED MESH

by

Leitao Chen

BS, East China University of Science and Technology, Shanghai, China, 2006

MS, Tongji University, Shanghai, China, 2009

Submitted to the Graduate Faculty of
the Swanson School of Engineering in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Leitao Chen

It was defended on

June 23, 2016

and approved by

Laura A. Schaefer, Ph.D., Professor, Department of Mechanical Engineering and Materials Science, University of Pittsburgh

Mark L. Kimber, Ph.D., Adjunct Assistant Professor, Department of Mechanical Engineering and Materials Science, University of Pittsburgh

Sangyeop Lee, Ph.D., Assistant Professor, Department of Mechanical Engineering and Materials Science, University of Pittsburgh

Michael Neilan, Assistant Professor, Department of Mathematics, University of Pittsburgh

Dissertation Director: Laura A. Schaefer, Ph.D., Professor, Department of Mechanical Engineering and Materials Science, University of Pittsburgh

FINITE VOLUME DISCRETE BOLTZMANN METHOD ON A CELL-CENTERED TRIANGULAR UNSTRUCTURED MESH

Leitao Chen, PhD

University of Pittsburgh, 2016

Due to its basis in kinetics, the lattice Boltzmann method (LBM) has been shown to be superior to conventional computational fluid dynamic (CFD) tools that solve the Navier-Stokes (NS) equation for complex flow problems, such as multiphase and/or multicomponent flows. However, after development for decades, the LBM still has not gained prominence over other CFD tools for a number of reasons, one of which is its unsatisfactory ability to handle complex boundary geometries. The goal of the current work is to overcome this difficulty.

In order to fully integrate the unstructured mesh for treating complex boundary geometries, the discrete Boltzmann equation (DBE), which is the Eulerian counterpart of the lattice Boltzmann equation (LBE), is chosen for the study. The finite volume (FV) method is selected to solve the governing equation due to its outstanding performance in handling an unstructured mesh and its built-in conservation. Therefore, the method in the current work is called the finite volume discrete Boltzmann method (FVDBM).

A FVDBM platform, both for isothermal and thermal models, is developed in the current work, which consists of three parts: the cell-centered (CC) triangular unstructured mesh, the FVDBM solver, and the boundary treatment, among which the latter two are the main areas of contribution. In the FVDBM solver, there are three components: the time marching scheme, the flux scheme, and the collision calculator.

The flux schemes are the major area of study because of their significance in the overall FVDBM model (they control the spatial order of accuracy) and their complexity (they calculate the spatial gradient term) on the unstructured mesh. A universal stencil is devel-

oped on the arbitrary CC triangular unstructured mesh, with which two categories of flux schemes are developed systematically: the Godunov type and the non-Godunov type. As a result, a total of five schemes (three Godunov schemes and two non-Godunov schemes) with different orders of accuracy are formulated, numerically validated and analyzed. Two major conclusions can be drawn. First, for any flux scheme, Godunov or non-Godunov, its actual order is roughly one order lower than its theoretical order for velocity solutions, due to the diffusion error introduced by the unstructured mesh and the Eulerian nature of the solver. Second, a Godunov scheme is less diffusive and more stable than a non-Godunov one if they are of the same order of accuracy.

Furthermore, a unique three-step boundary treatment is developed in detail for the current model. With the proposed boundary treatment, a variety of physical boundary conditions (velocity, density, and temperature, etc.) can be realized on the complex boundaries with the triangular unstructured mesh in a unified way. It can also incorporate different lattice models indiscriminately. With sufficient numerical testing, it is found that the boundary treatment is at least second-order accurate in space, and it can accurately preserve Dirichlet boundary conditions up to machine accuracy under different scenarios.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

The conventional computational fluid dynamic (CFD) tools that have been developed specifically to solve the Navier-Stokes (NS) equation have been extremely successful for more than a half century, especially in the area of aerodynamics for compressible and incompressible flows. However, when used to model unconventional problems such as multiphase and/or multicomponent flows and non-Newtonian flows, the NS equation is not well qualified to provide an *a priori* theory for the underlying physics. The NS equation solves only the macroscopic variables (velocities, pressure, etc.) and derives from the conservation laws. However, in a complex fluid system, more degrees of freedom than the macroscopic variables emerge, and the conservation laws are not the only governing equations. A more complex mechanism can be achieved by viewing the fluid body as an ensemble of the fluid molecules. The corresponding method solves the fluid system at the microscopic level and provides the most detailed representation. Unfortunately, tracking each individual molecule is too computationally expensive.

A question is naturally raised, is there a mesoscopic method that has a scale between the macroscale and microscale? The lattice Boltzmann method (LBM), which is anchored in kinetic theory, is the desired answer. By creating a concept called a "particle", which is a group of fluid molecules, the LBM can solve the underlying physics of complex flows as well as keep the computational overhead low. Some prior work has shown that the LBM has great potential in simulations of complex flows such as multiphase and/or multicomponent flows [43, 89, 90], free-surface flows and droplets [38, 51], particle-laden flows [6, 54], and crystallization [74]. However, the LBM still has several major defects compared to conventional CFD techniques, such as a lack of an accurate thermal model [10, 26, 44, 55, 84, 114], inadequate stability performance [3, 7, 13, 55], and poor ability to handle complex geome-

tries [39, 46, 71, 75, 99]. The current work is concentrated on improving the capabilities of the LBM to incorporate complex geometries in the realm of simple flow problems that are single-phase and of moderate Reynolds number (Re) and Knudsen number (Kn), in order to help the LBM become a more practical alternative to CFD for the problems with complex geometries such as porous media, but in such simple flow regime. It is also hoped that all of the methods developed in this work can be developed and extended by others to complex flows problems, such as multi-phase and/or high-Re flows.

1.1 FUNDAMENTALS

To be clear, the term LBM has two meanings in this work, a broad one (above) and a specific one. The former is used to distinguish any general Botzmann-equation-based method from the NS-based methods; the latter specifically denotes the method that solves the lattice Boltzmann equation (LBE) in order to be differentiated from other lattice Boltzmann techniques. The term LBM carries the specific meaning in the rest of this dissertation, unless explained otherwise.

The LBM (or LBE) has been substantially studied for the past decades, and therefore has been widely accepted as a practical simulation approach. The LBE without the body force term is defined in Eq. (1.1) such that:

$$f_i\left(\boldsymbol{x}+\boldsymbol{\xi}_i, t+1\right)-f_i\left(\boldsymbol{x}, t\right)=\Omega_i\left(\boldsymbol{x}, t\right) \tag{1.1}$$

In Eq. (1.1), $f_i\left(\boldsymbol{x}, t\right)$, which is the short notation for $f\left(\boldsymbol{x}, \boldsymbol{\xi}_i, t\right)$, is the single particle distribution function (PDF) that travels at velocity $\boldsymbol{\xi}_i$ in velocity space (the subscript i marks the ith component in velocity space), at location $\boldsymbol{x}$ in configuration space and at time t in time space. $\Omega_i\left(\boldsymbol{x}, t\right)$ is the general collision term that carries all the nonlinear kinetic physics. The concept "lattice" defines how many components of velocity space there are and the direction and magnitude of each individual component. Usually, Eq. (1.1) is solved with a repeated two-step procedure: the collision step $f_i^{+}\left(\boldsymbol{x}, t\right)=f_i\left(\boldsymbol{x}, t\right)+\Omega_i\left(\boldsymbol{x}, t\right)$ and streaming step $f_i\left(\boldsymbol{x}+\boldsymbol{\xi}_i, t+1\right)=f_i^{+}\left(\boldsymbol{x}, t\right)$, in which the superscript "+" denotes the post-

collision condition. There are many options for the collision model; however, for simple flow problems, the Bhatnagar-Gross-Krook (BGK) model [8] has been proven to be adequate. The BGK collision term is shown in Eq. (1.2), where τ is the relaxation time and $f_i^{eq}(\boldsymbol{x}, t)$ is the equilibrium PDF defined by the Maxwellian distribution function. The dynamics in the BGK collision are such that the PDFs traveling in all directions relax towards their own equilibrium states at the same rate measured by τ.

$$\Omega_i(\boldsymbol{x}, t) = -\frac{1}{\tau}\left(f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)\right) \tag{1.2}$$

Historically, there are two paths to reach the LBE (see Fig. 1), which reflect the community's two stages of insights into this topic. The first path is more application-driven, while the second one is purely mathematical. The difference in methodology gives two completely different comprehensions of the topic, which will be elaborated on in the following parts.

1.1.1 From the LGA to the LBE

The early development of the LBM was rooted in the lattice gas automata (LGA) or lattice gas cellular automata (LGCA) (see Eq. (1.3)), which pioneered the approach that models continuum flows with simple particle dynamics [28, 33, 34, 108]. In Eq. (1.3), the particle n_i has only two values: *1* or *0*, which corresponds to whether there is a particle or not. Therefore, the collision term C_i becomes a Boolean operator that transforms the pre-collision particles into the post-collision particles according to certain collision rules. In the famous FHP model [33,34], a six-velocity hexagonal lattice is applied. Therefore, a six-digit boolean string can be used to represent the particles at each site. For example, according to a two-particle collision rule, the pre-collision particles *100100* become *010010* after collision, which means that the head-on pair of two particles along one direction becomes another head-on pair along a different direction. Other collision rules may also apply, and collisions involving more than two particles are also possible. The LGA is simple and effective. However, its Boolean nature will generate a large amount of statistical noise and there is no physical foundation for the collision rules, which become the two main motives to develop the LBM. By using positive floating numbers instead of the Boolean numbers for the particles to dampen the noise, and by replacing the physically meaningless collision operator with the

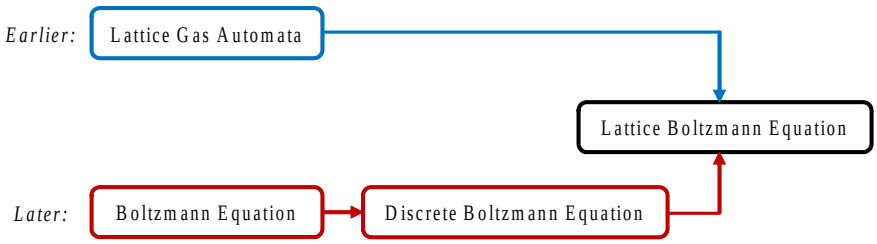


Figure 1: Two origins of the LBE

BGK collision term that contains the correct physics, one can obtain the LBM without changing other parts in the system, such as the lattice and repeated collision-streaming operation, etc. For this reason, until about two decades ago, the LBM was merely considered nothing more than a floating-number and physical successor of the LGA (one can see that the LBE (Eq. (1.1)) and LGA (Eq. (1.3)) carry exactly the same form).

$$n_i\left(\boldsymbol{x}+\boldsymbol{\xi}_i, t+1\right)-n_i\left(\boldsymbol{x}, t\right)=C_i\left(\boldsymbol{x}, t\right) \tag{1.3}$$

1.1.2 From the BE to the LBE

The LBM has been proven superior to the LGA in many aspects, ironically due to the basic modeling concepts borrowed from the LGA. However, the way of viewing the LBM as the extension of the LGA is insufficient in explaining the fundamental characteristics of the LBE, which will eventually stall its own development. Consequently, a number of researchers focused their attention on seeking the LBM's root in kinetic theory, by building the connection between the LBE and the Boltzmann equation (BE), which is accepted as the best description of kinetic theory. Later, many researchers [1, 9, 47, 48, 92] successfully derived the LBE from the BE without any help from the LGA, which became the second path to obtain the LBE (see Fig. 1). Their success is significant because they found not only a second path to the LBE, but also a rigorous and informative one that, for the first time in history, has brought our understanding of the LBM to a whole new level and also

enabled us to see a bigger picture beyond the LBM. The following mathematical derivation of the LBE from the BE is generalized from different publications [9,47,48] in a shorter but more intuitive fashion, which starts from the BE with the BGK collision:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi}\frac{\partial f}{\partial \boldsymbol{x}} = \frac{1}{\lambda}(g - f) \tag{1.4}$$

where g is the Boltzmann-Maxwellian distribution function defined as:

$$g \equiv \frac{\rho}{(2\pi RT)^{\frac{D}{2}}} \exp\left(-\frac{(\boldsymbol{\xi} - \boldsymbol{u})^2}{2RT}\right) \tag{1.5}$$

and where R is the ideal gas constant, D is the dimension of the configuration space, and ρ, $\boldsymbol{u}$ and T are the macroscopic density, velocity and temperature, respectively. The links between the f, which is the microscopic variable, with those macroscopic counterparts are defined as:

$$\rho = \int f d\boldsymbol{\xi} = \int g d\boldsymbol{\xi} \tag{1.6a}$$

$$\rho \boldsymbol{u} = \int \boldsymbol{\xi} f d\boldsymbol{\xi} = \int \boldsymbol{\xi} g d\boldsymbol{\xi} \tag{1.6b}$$

$$\rho\varepsilon = \int (\boldsymbol{\xi} - \boldsymbol{u})^2 f d\boldsymbol{\xi} = \int (\boldsymbol{\xi} - \boldsymbol{u})^2 g d\boldsymbol{\xi} \tag{1.6c}$$

It should be noted that the moment integral of either f or g is able to yield the same macroscopic variables. The variable ε is the internal energy whose relation with T is:

$$T = \frac{2}{D_0 R}\varepsilon \tag{1.7}$$

where D_0 is the degrees of freedom of a particle. It can be seen that all of the variables in Eq. (1.4) are continuous, so Eq. (1.4) is also called the continuous Boltzmann equation (CBE). The equations from (1.4) to (1.6c) form a perfect closed model that can represent any macroscopic problem intrinsically governed by the kinetic physics. However, due to the continuity in the velocity space (the continuities in the configuration and time spaces pose no difficulty), the CBE is inherently unsolvable. When a system consists of a large number of particles, where each of them has infinite possibilities of traveling velocities, the Boltzmann-Maxwellian distribution function in Eq. (1.5) and the moment integrals in Eqs. (1.6) simply cannot be calculated, since they all have $\boldsymbol{\xi}$ as a variable. (This is why it was almost a century until the importance of the Boltzmann equation was realized since Ludwig

Boltzmann[1] devised it in 1872.) As a result, the discretization of velocity space is required, which will result in a finite number of particular velocities that should well represent the continuous velocity space. The assembly of such discrete particular velocities is referred to as the lattice model in the physical point of view or abscissas in the mathematical point of view. The choice of lattice model is not arbitrary, as it should at least be geometrically symmetric. In addition, it should guarantee the numerical approximation of the moment integrals, Eqs. (1.6), up to a certain degree of accuracy, and the physics underlying Eq. (1.5) must be correct and accurate, when it is approximated with the discrete velocities. The process to obtain an *a priori* lattice model is tedious and obscure [93], and therefore will not be covered in the current work. The following development is based on the assumption that a proper lattice model is already known. Given such a lattice model, the moment integrals in Eqs. (1.6) become quadratures, namely:

$$\rho = \sum_{i=0}^{M-1} f_i = \sum_{i=0}^{M-1} g_i \tag{1.8a}$$

$$\rho \boldsymbol{u} = \sum_{i=0}^{M-1} \boldsymbol{\xi}_i f_i = \sum_{i=0}^{M-1} \boldsymbol{\xi}_i g_i \tag{1.8b}$$

$$\rho \varepsilon = \sum_{i=0}^{M-1} \left(\boldsymbol{\xi}_i - \boldsymbol{u}\right)^2 f_i = \sum_{i=0}^{M-1} \left(\boldsymbol{\xi}_i - \boldsymbol{u}\right)^2 g_i \tag{1.8c}$$

where M is the total number of discrete velocities, with each of them labeled with the subscript i. Expanding Eq. (1.5) with a Taylor series and small-velocity assumption up to the third-order yields:

$$\begin{aligned} g &= \frac{\rho}{(2\pi RT)^{\frac{D}{2}}} \exp\left(-\frac{\boldsymbol{\xi}^2}{RT}\right) \exp\left(\frac{\boldsymbol{\xi}\cdot\boldsymbol{u}}{RT} - \frac{\boldsymbol{u}^2}{2RT}\right) \\ &= \frac{\rho}{(2\pi RT)^{\frac{D}{2}}} \exp\left(-\frac{\boldsymbol{\xi}^2}{RT}\right) \left[1 + \frac{\boldsymbol{\xi}\cdot\boldsymbol{u}}{RT} + \frac{(\boldsymbol{\xi}\cdot\boldsymbol{u})^2}{2(RT)^2} - \frac{\boldsymbol{u}^2}{2RT}\right] + \mathcal{O}\left(\boldsymbol{u}^3\right) \end{aligned} \tag{1.9}$$

By truncating the third-order term of macroscopic velocity in Eq. (1.9), the equilibrium PDF with decent accuracy can be obtained:

[1] Ludwig Eduard Boltzmann (February 20, 1844 – September 5, 1906) was an Austrian physicist and philosopher specializing in non-equilibrium statistical mechanics.

$$g \simeq f^{eq} = \rho W(\boldsymbol{\xi}) \left[1 + \frac{\boldsymbol{\xi} \cdot \boldsymbol{u}}{RT} + \frac{(\boldsymbol{\xi} \cdot \boldsymbol{u})^2}{2(RT)^2} - \frac{\boldsymbol{u}^2}{2RT} \right] \tag{1.10}$$

where $W(\boldsymbol{\xi})$ is the weight coefficient function:

$$W(\boldsymbol{\xi}) = \frac{1}{(2\pi RT)^{\frac{D}{2}}} \exp\left(-\frac{\boldsymbol{\xi}^2}{RT} \right) \tag{1.11}$$

It can be seen that the equilibrium PDF can be discretized once the weight coefficient function is decomposed in velocity space. In order to do this, the integral of the moment needs to be evaluated by plugging in Eq. (1.10) into Eqs. (1.6) such that:

$$\begin{aligned} \boldsymbol{I} &= \int \Phi(\boldsymbol{\xi}) f^{eq} d\boldsymbol{\xi} \\ &= \rho \int \Phi(\boldsymbol{\xi}) W(\boldsymbol{\xi}) \left[1 + \frac{\boldsymbol{\xi} \cdot \boldsymbol{u}}{RT} + \frac{(\boldsymbol{\xi} \cdot \boldsymbol{u})^2}{2(RT)^2} - \frac{\boldsymbol{u}^2}{2RT} \right] d\boldsymbol{\xi} \\ &= \frac{\rho}{(2\pi RT)^{\frac{D}{2}}} \int \Phi(\boldsymbol{\xi}) \exp\left(-\frac{\boldsymbol{\xi}^2}{RT} \right) \left[1 + \frac{\boldsymbol{\xi} \cdot \boldsymbol{u}}{RT} + \frac{(\boldsymbol{\xi} \cdot \boldsymbol{u})^2}{2(RT)^2} - \frac{\boldsymbol{u}^2}{2RT} \right] d\boldsymbol{\xi} \end{aligned} \tag{1.12}$$

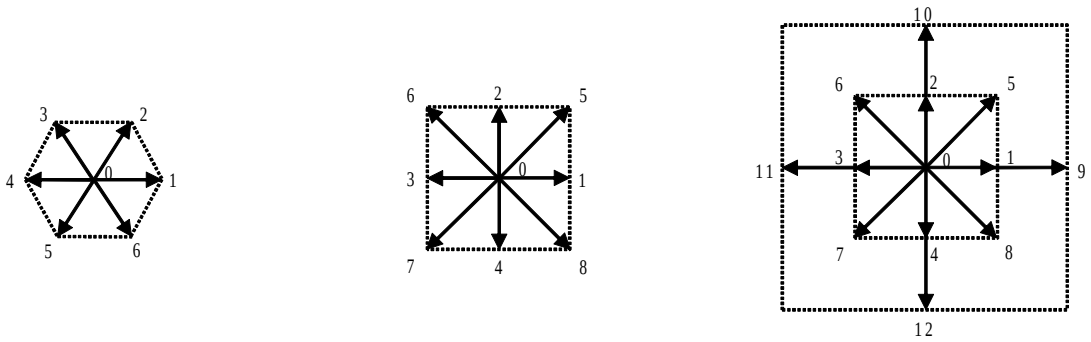
where $\Phi(\boldsymbol{\xi})$ is the polynomial of $\boldsymbol{\xi}$ with arbitrary order ($\Phi(\boldsymbol{\xi}) = 1$ gives $\boldsymbol{I} = \rho$, and $\Phi(\boldsymbol{\xi}) = \boldsymbol{\xi}$ gives $\boldsymbol{I} = \rho \boldsymbol{u}$, etc.). Equation (1.12) contains the integral $\int \Phi(\boldsymbol{\xi}) \exp\left(-\boldsymbol{\xi}^2 / RT \right) d\boldsymbol{\xi}$ that can be numerically calculated by Gaussian quadrature with a proper lattice model [23], namely:

$$\int \Phi(\boldsymbol{\xi}) \exp\left(-\frac{\boldsymbol{\xi}^2}{RT} \right) d\boldsymbol{\xi} = \sum_{i=0}^{M-1} W_i \Phi(\boldsymbol{\xi}_i) \exp\left(-\frac{\boldsymbol{\xi}_i}{RT} \right) \tag{1.13}$$

Then, by replacing g in Eq. (1.4) with Eq. (1.10) and adding subscript i to the equation to specify the discrete identifier, the discrete Boltzmann equation (DBE) with the BGK collision model reads:

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \frac{\partial f_i}{\partial \boldsymbol{x}} = \frac{1}{\lambda} (f_i^{eq} - f_i) \tag{1.14}$$

It might appear to be too easy to derive the DBE (Eq. (1.14)) from the BE (Eq. (1.4)) by just comparing those two equations, in which the major difference is the addition of the subscript i. However, in the back scene, a huge effort is required to achieve this seemingly easy advance. As just discussed, two tasks need to be finished: the discretization of the velocity space (the lattice model) and the explicit form of the equilibrium PDF. In addition, f^{eq} needs the lattice model as its constraint to determine its final explicit expression. When the lattice model changes, f^{eq} also changes. Fig. 2 shows three popular two-dimensional lattice models with different numbers of discrete velocities (or lattice velocities) and configurations.



(a) D2Q7 (b) D2Q9 (c) D2Q13

Figure 2: Three commonly used two-dimensional lattice models

A notation that indicates the numbers of dimensions and lattice velocities is applied. For example, D2Q9 indicate the two-dimensional nine-velocity lattice model.

For each of the lattice models in Fig. 2, its structure along with the definition of the resulting f^{eq} is listed as follows.

For D2Q7:

$$\boldsymbol{\xi}_i = \begin{cases} (0,0)\, c & i = 0 \\ \left(\cos\left[(i-1)\,\pi/3\right], \sin\left[(i-1)\,\pi/3\right]\right) c & i = 1-6 \end{cases} \tag{1.15a}$$

$$f_i^{eq} = \omega_i \rho \left[1 + \frac{\boldsymbol{\xi}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{\left(\boldsymbol{\xi}_i \cdot \boldsymbol{u}\right)^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2}\right] \tag{1.15b}$$

$$\omega_i = \begin{cases} 1/2 & i = 0 \\ 1/12 & i = 1-6 \end{cases} \tag{1.15c}$$

$$c_s = \frac{c}{\sqrt{4}} \tag{1.15d}$$

For D2Q9:

$$\boldsymbol{\xi}_i = \begin{cases} (0,0)\, c & i = 0 \\ \left(\cos\left[(i-1)\,\pi/2\right], \sin\left[(i-1)\,\pi/2\right]\right) c & i = 1-4 \\ \left(\cos\left[(i-5)\,\pi/2 + \pi/4\right], \sin\left[(i-5)\,\pi/2 + \pi/4\right]\right) \sqrt{2} c & i = 5-8 \end{cases} \tag{1.16a}$$

$$f_i^{eq} = \omega_i \rho \left[1 + \frac{\boldsymbol{\xi}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{\left(\boldsymbol{\xi}_i \cdot \boldsymbol{u}\right)^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2}\right] \tag{1.16b}$$

$$\omega_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i = 1-4 \\ 1/36 & i = 5-8 \end{cases} \tag{1.16c}$$

$$c_s = \frac{c}{\sqrt{3}} \tag{1.16d}$$

For D2Q13:

$$\boldsymbol{\xi}_i = \begin{cases} (0,0)\, c & i = 0 \\ \left(\cos\left[(i-1)\,\pi/2\right], \sin\left[(i-1)\,\pi/2\right]\right) c & i = 1-4 \\ \left(\cos\left[(i-5)\,\pi/2+\pi/4\right], \sin\left[(i-5)\,\pi/2+\pi/4\right]\right) \sqrt{2}c & i = 5-8 \\ \left(\cos\left[(i-9)\,\pi/2\right], \sin\left[(i-9)\,\pi/2\right]\right) 2c & i = 9-12 \end{cases} \tag{1.17a}$$

$$f_i^{eq} = \omega_i \rho \left[1 + \frac{\boldsymbol{\xi}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^3}{2c_s^6} - \frac{3\,(\boldsymbol{\xi}_i \cdot \boldsymbol{u})\,(\boldsymbol{u} \cdot \boldsymbol{u})}{2c_s^4}\right] \tag{1.17b}$$

$$\omega_i = \begin{cases} 3/8 & i = 0 \\ 1/12 & i = 1-4 \\ 1/16 & i = 5-8 \\ 1/96 & i = 9-12 \end{cases} \tag{1.17c}$$

$$c_s = \frac{c}{\sqrt{2}} \tag{1.17d}$$

In the lattice models above, c refers to a particular velocity, which is usually taken as unity, and $c_s \equiv \sqrt{RT}$ is the speed of sound. Therefore, the DBE system not only includes the DBE itself (Eq. (1.14)), but also the lattice model and corresponding equilibrium PDF description, such as Eqs. (1.15), Eqs. (1.16) or Eqs. (1.17).

The connection between the BE and DBE already has been built, so the last step is to derive the LBE from the DBE; specifically speaking, to discretize configuration and time spaces. By using the material derivative $Df/Dt = \partial f/\partial t + \boldsymbol{\xi} \cdot \boldsymbol{\nabla} f$, the DBE (Eq. (1.14)) can be translated from the Eulerian description to the Lagrangian description such that:

$$\frac{Df_i}{Dt} = \frac{1}{\lambda}\left(f_i^{eq} - f_i\right) \tag{1.18}$$

Next, it is chosen to replace the material derivative with a first-order upwind difference both in time and configuration spaces:

$$\frac{f_i\left(\boldsymbol{x} + \Delta\boldsymbol{x}, t + \Delta t\right) - f_i\left(\boldsymbol{x}, t\right)}{\Delta t} = \frac{1}{\lambda}\left[f_i^{eq}\left(\boldsymbol{x}, t\right) - f_i\left(\boldsymbol{x}, t\right)\right] \tag{1.19}$$

It is assumed that, during every time step, the distance traveled by each particle f_i is equal to $\Delta \boldsymbol{x}$, the spacing in the discretized configuration space, or mathematically speaking:

$$\Delta \boldsymbol{x} = \boldsymbol{\xi}_i \Delta t \tag{1.20}$$

By plugging Eq. (1.20) into Eq. (1.19) and making a rearrangement, we have:

$$f_i\left(\boldsymbol{x} + \boldsymbol{\xi}_i \Delta t, t + \Delta t\right) - f_i\left(\boldsymbol{x}, t\right) = \frac{\Delta t}{\lambda}\left[f_i^{eq}\left(\boldsymbol{x}, t\right) - f_i\left(\boldsymbol{x}, t\right)\right] \tag{1.21}$$

At last, by assuming $\tau = \lambda / \Delta t$ and $\Delta t = 1$, we have the LBE as follows, which is exactly the same as Eq. (1.1) with the BGK collision (Eq. (1.2)):

$$f_i\left(\boldsymbol{x} + \boldsymbol{\xi}_i, t + 1\right) - f_i\left(\boldsymbol{x}, t\right) = \frac{1}{\tau}\left[f_i^{eq}\left(\boldsymbol{x}, t\right) - f_i\left(\boldsymbol{x}, t\right)\right] \tag{1.22}$$

1.1.3 Comparison between the LBE and the DBE

To develop a more thorough comparison, the four kinetic equations that have been discussed are listed in Tab. 1. However, the only feasible candidates for flow simulations are the DBE and LBE, since the LGA lacks the correct physics and the BE is impossible to solve, as discussed before.

Next, it is necessary to identify the common features and differences between the DBE and LBE in order to help us choose the suitable one for different problems. From the rigorous derivation in Sec. 1.1.2, it can be seen that the DBE and LBE share the same origin, the BE. In addition, the DBE and LBE have the same velocity discretization, including the lattice models and definitions of equilibrium PDF, and therefore, the DBE and LBE are essentially equivalent in terms of physics and equally capable of solving the same problem. However, the two equations are also different in other aspects, especially the numerical ones. First, the DBE is Eulerian but the LBE is Lagrangian. Second, only the velocity space is discretized in the DBE, but all three spaces (velocity, configuration and time) are discretized in the LBE. Third, the time step size for the LBE is fixed ($\Delta t = 1$) while the time can be discretized with an arbitrary size for the DBE. Fourth, for the LBE, the spatial step size, or the discretization of the configuration space, is patched or coupled with the discretizations of velocity and time space (see Eq. (1.20)). Such a coupling mechanism is named multi-space coupling (MSC) in this dissertation. However, the DBE has no MSC. Fifth, and last

Table 1: Four kinetic equations for flow simulations

LGA	$n_i(\boldsymbol{x}+\boldsymbol{\xi}_i, t+1) - n_i(\boldsymbol{x}, t) = C_i(\boldsymbol{x}, t)$	Eq. (1.3)
BE with BGK	$\partial f/\partial t + \boldsymbol{\xi}\cdot\boldsymbol{\nabla} f = (g-f)/\lambda$	Eq. (1.4)
DBE with BGK	$\partial f_i/\partial t + \boldsymbol{\xi}_i\cdot\boldsymbol{\nabla} f_i = (f_i^{eq} - f_i)/\lambda$	Eq. (1.14)
LBE with BGK	$f_i(\boldsymbol{x}+\boldsymbol{\xi}_i, t+1) - f_i(\boldsymbol{x}, t) = [f_i^{eq}(\boldsymbol{x}, t) - f_i(\boldsymbol{x}, t)]/\tau$	Eq. (1.22)

but not least, due to the MSC, solving the LBE can be replaced by a repeated steaming-collision (S-C) procedure. Therefore, the LBE is very computationally efficient and can be easily parallelized. In contrast, the DBE cannot be simplified and has to be solved with standard numerical methods for the partial differential equation (PDE). The similarities and differences between the two equations are summarized in Tab. 2.

Similar to the LBM for LBE, the method that solves the DBE is named the discrete Boltzmann method (DBM). From Tab. 2, it can be concluded that the LBM and DBM are physically the same but numerically different. The LBM is nothing but a special finite-difference version of the DBM [47,48]. Therefore, we can also use the DBM to the solve the problems that can be solved by the LBM.

1.2 BACKGROUND AND MOTIVATIONS

In order to be a modern simulation tool for fluid dynamics, a method should be versatile enough to overcome various complexity of the problems. One major part of the complexity comes from the geometry, which is ubiquitous in a variety of engineering problems. An extreme example is the mass and heat transfer in porous media, which involves exceptionally complicated geometries. The features of the DBM and LBM listed in Tab. 2 are unbiasedly compared. In this section, by judging those features with detailed explanations and examples,

Table 2: Comparison between the discrete Boltzmann equation (DBE) and lattice Boltzmann equation(LBE)

		DBE	LBE
Similarities	Origin	BE	BE
	Discretization of velocity space	The same	The same
Differences	Governing equation	Lagrangian	Eulerian
	Discretization	Discretized $\boldsymbol{\xi}$, continuous $\boldsymbol{x}$ and t	Discretized $\boldsymbol{\xi}$, $\boldsymbol{x}$ and t
	Fixed time step size	No	Yes
	Multi-Space Coupling	No	Yes
	S-C procedure	No	Yes

it will be concluded that it is not only possible, but also necessary, to choose the DBM over the LBM, in order to fully be equipped with the capability to deal with complex geometries.

The features listed in Tab. 2 for the LBM have a significant consequence, which is that the mesh structure for the LBM is highly constrained. Usually, when one solves a PDE (e.g. NS) numerically, the spatial discretization, or meshing, is performed independently, and the resulting mesh information is not contained in the PDE. However, for the LBE, the meshing is coupled with the structure of the applied lattice model due to the MSC. As a result, as shown in Fig. 3, the mesh structure for the LBM is predetermined by the lattice model, and the mesh information is already contained in the LBM itself. In addition, during the streaming step, each node (the black nodes in Fig. 3) in configuration space should be able to send and receive PDFs to and from its nearby neighbor nodes. Since this happens to every node, an unbroken network for streaming must be created. Such a streaming network also contains the information where each node in the mesh should be located in the configuration space. As the result, the streaming network and the grid network or mesh are the same thing. As shown in Fig. 3, the D2Q7, a triangular lattice, requires a triangular mesh in which each triangle is equilateral; and the mesh yielded by the square D2Q9 lattice is square-shaped as well. The D2Q13 is another square lattice, so the corresponding mesh for it is also square-shaped. Each line in Fig. 3 both represent the streaming path of PDFs and the geometric

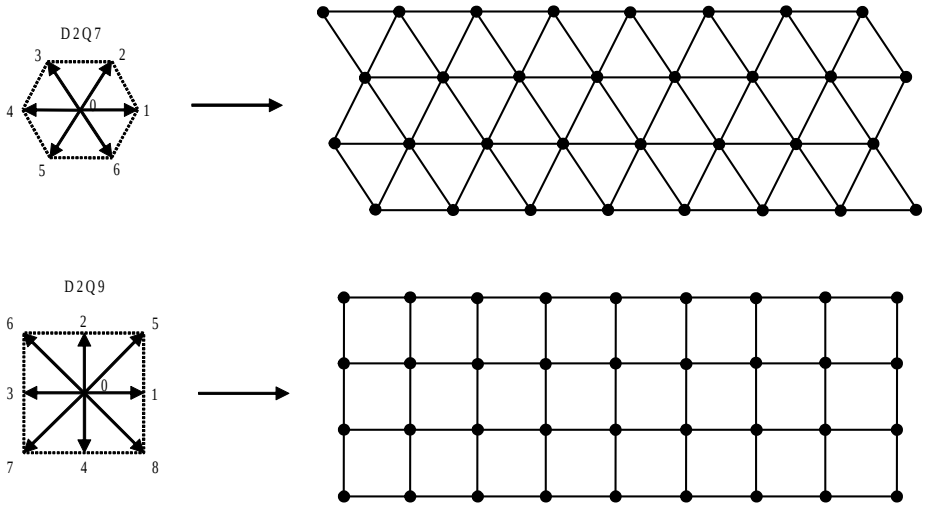


Figure 3: The meshes for different lattices in the LBM

boundary between two adjacent grid units. Such a mesh has a fixed topology among grid units, and therefore is called a structured mesh.

The challenge when dealing with complex geometries is two-fold. First, how to capture the complicated geometries or boundaries correctly with such a rigid mesh structure; and second, how to at the same time easily have local mesh refinement in the regions where it is necessary, such as the near-boundary region and high-gradient areas, in order to keep the overall numerical algorithm computationally cheap without having a global refinement. Obviously, it is difficult, if not impossible, to overcome this two-fold mesh-wise challenge when the topology and the structure of the mesh are not flexible, as the ones shown in Fig. 3. This is the reason that LBM is inferior to modern CFD codes when dealing with complex geometries, which was realized in the LBM community at a very early stage [46, 75, 99]. However, the simplicity of the LBM is simply too strong to be discarded. Many later efforts has been made to improve the situation while still sticking, completely or partially, to the LBM approach. When *completely* holding to the LBM approach, which is called the strict LBM in this work, all conditions are satisfied in the LBE column in Tab. 2; otherwise, in Tab. 2, the MSC property and the following S-C procedure will be dropped, and then the

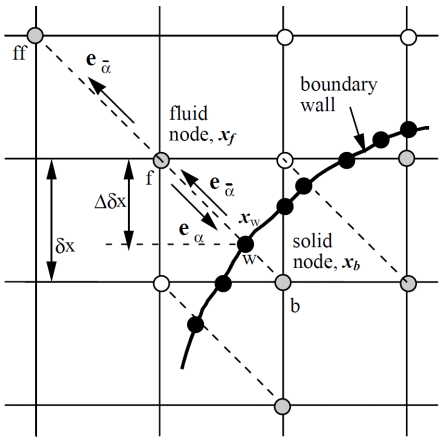


Figure 4: A typical treatment of the curved boundaries for the LBM

resulting LBM is called the loose LBM. With the following examples and analysis, it can be seen that either the strict or loose LBM are not able to fully and desirably incorporate complex boundaries and local mesh refinement.

1.2.1 The strict LBM

First, we will discuss how complex boundaries are handled with the strict LBM. Since the mesh for the LBM is predetermined by the lattice structure (Fig. 3), when a complex boundary is present, the grid nodes are generally not able to conform to the boundaries, and some nodes will even be located outside of the boundaries. Therefore, the mesh for the LBM is a non-body-fitting mesh. The opposite of the non-body-fitting mesh is the body-fitting mesh, in which the shape and location of the boundaries can be well represented by a set of boundary nodes and all nodes will be within the computational domain (either in the interior region or on the boundaries). Therefore, the LBM cannot solve the problems with complex boundaries unless two questions can be answered: how to treat the nodes outside of the boundaries and how to determine the shape and location of the boundaries? These questions were initially answered by Filippova and Hänel [31] and later improved by Mei, Luo and Shyy (hereafter referred as MLS) [71], whose scheme is illustrated in Fig. 4.

In Fig. 4, the square-shaped boxes are the mesh lines, and, at the same time, the links of the network for streaming. The nodes at the joint positions of the network are the grid nodes, which are also lattice nodes where the streaming starts and ends. The black curved line is the wall boundary. It cuts through some of the links in the network and separates all lattice nodes into two groups: the boundary nodes and fluid nodes, depending on that on which side a node is located. The filled black nodes, which are also called the wall nodes in Fig. 4, represent the locations where the wall boundary intercepts the streaming links. As a result, the shape and location of the boundary can be replaced by the ensemble of those wall nodes. The geometric information of the boundary between two adjacent wall nodes will be approximated, but will be exactly preserved at the nodes. A variable Δ is calculated to store the location information of each wall node as follows:

$$\Delta = \frac{|\boldsymbol{x}_f - \boldsymbol{x}_w|}{|\boldsymbol{x}_f - \boldsymbol{x}_b|} \tag{1.23}$$

where $\boldsymbol{x}_f$ and $\boldsymbol{x}_b$ are locations of the fluid node and boundary node whose streaming link is cut by the wall node $\boldsymbol{x}_w$. Then, in order to finish the streaming step, the PDF at $\boldsymbol{x}_b$, which will be streaming to $\boldsymbol{x}_f$, has to be calculated. Therefore, although it is physically unrealistic, the velocity at $\boldsymbol{x}_b$ needs to be evaluated. In MLS scheme, the velocity at $\boldsymbol{x}_b$ is:

$$\boldsymbol{u}_b = \frac{\Delta - 1}{\Delta}\boldsymbol{u}_f + \frac{1}{\Delta}\boldsymbol{u}_w \tag{1.24}$$

The MLS scheme is second-order accurate and can handle the curved boundaries very well. However, the scheme is intrinsically unstable when the boundaries become more complex than just being curved. When the boundary is very close to the fluid nodes, Δ becomes very small. Since Δ appears as a denominator in Eq. (1.24), the resulting velocity $\boldsymbol{u}_b$ will be very large. In some problems, such as flow in porous media, the curvature of the boundary becomes extremely large and highly irregular at different locations. Therefore, it is impossible to control every wall node to make sure that its distance to the nearest fluid node is large enough. Eventually, the solution at some singular nodes will become unstable and soon propagate throughout the entire computational domain.

Second, it can be found that the strict LBM does not allow local refinement due to the unbroken streaming network. An example based on the D2Q9 lattice is shown in Fig. 5. In Fig. 5(a), the computational domain is already meshed with nine square blocks and a

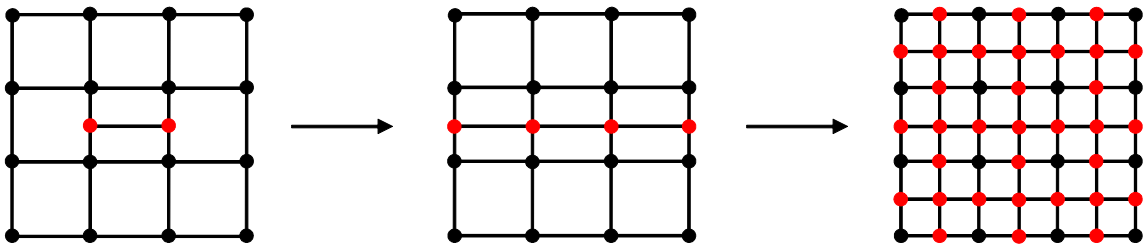


Figure 5: Mesh refinement for the LBM

total of 16 nodes (black). Next, assume that it is desired to just decrease the spacing in the y direction in the central block. Therefore, two more nodes (red) will be created, and the central block will be cut in half. So, the total computational cost will be 18 nodes. However, such a mesh cannot be solved since the two newly created nodes have no neighbors in the x direction to finish the streaming step. As a result, as shown in Fig. 5(b), the link for streaming must be extended both in $+x$ and -x directions until it reaches the boundary. This means, though, that a new issue is raised. For the three lines of nodes in the middle, the spacing in the x direction is larger than that in the y direction. However, the D2Q9 lattice requires the same velocities both in the x and y direction. Therefore, if the PDF traveling in the x direction can reach a neighbor node after the streaming step, the PDF traveling in the y direction cannot do so, and vice versa. Consequently, the mesh has to be refined in the x direction as well for the entire computational domain, in order to have the same spacings in x and y directions, as shown in Fig. 5(c). At last, there will be 49 nodes in the final mesh, which is almost three times the originally desired 18 nodes. So, the LBM is not able to include local refinement, as any local refinement will end up as an overall refinement.

1.2.2 The loose LBM

In Tab. 2 for the LBE, the S-C procedure is based upon the MSC, which is achieved by both fixing the time step size to unity and using the numerical relation shown in Eq. (1.20). Either

changing the time step size or discarding Eq. (1.20) will break the S-C procedure. When the time step size is not unity or Eq. (1.20) is no longer valid, after each streaming step, the PDF will stop at a location where there is not another lattice node to receive the incoming PDF. Such a location is usually called a non-grid point compared to the grid point, which is also the lattice node in the strict LBM. However, by using a proper interpolation, the PDFs at the non-grid points can be interpolated using the PDFs at the grid points. Therefore, by inserting a interpolation step in the original S-C procedure, the LBM can be solved by the new Streaming-Interpolation-Collision (S-I-C) procedure, which therefore becomes the loose LBM. In the loose LBM, the size of the time step is still unity; however, Eq. (1.20) is loosened, which means the coupling between the mesh and lattice structure can be abolished. Therefore, the LBM can be solved on an arbitrary mesh with the S-I-C procedure. Several authors first realized this point [9, 47–49, 75, 99], and showed that the LBM can be solved on a non-uniform mesh and even unstructured mesh. Due to the ability to incorporate an arbitrary mesh, the LBM with the S-I-C procedure can use the body-fitting mesh to handle curved or complex boundaries. Therefore, the stability issue caused by a small Δ in section 1.2.1 can be completed avoided. As shown in Fig. 6, the computational domain around a circular cylinder can be discretized with a curvilinear mesh [45, 46]. Therefore, the circular boundary can be exactly represented by a group of nodes that are precisely located on the boundary.

Also, because of the ability to incorporate an arbitrary mesh, local mesh refinement can be achieved. In Fig. 7, the mesh spacing is gradually increasing from the flow inlet on the left hand side to the downstream direction [49]. The mesh in the inlet region can be refined without increasing the refinement level in other areas.

However, such a mesh refinement is still not local, even though it is much better than the global refinement. In Fig. 7, if it is also desired to refine the spacing in the y direction in the inflow region, the mesh spacing in the y direction of the entire computational domain will be refined, since the mesh lines in the x direction can only be terminated at the boundaries. In order to overcome this, a truly local refining algorithm was first introduced by Filippova and Hänel [31] and later improved by Yu *et al.* [112, 113], in which the mesh lines can be ended internally. Fig. 8 shows a mesh for the simulation of lid-driven flow. Since the

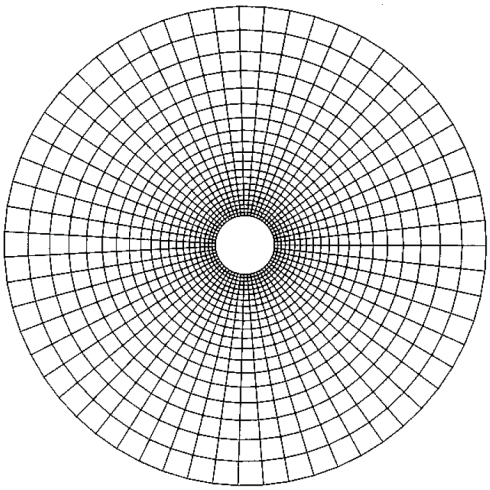


Figure 6: The LBM with a curvilinear mesh for the circular boundary

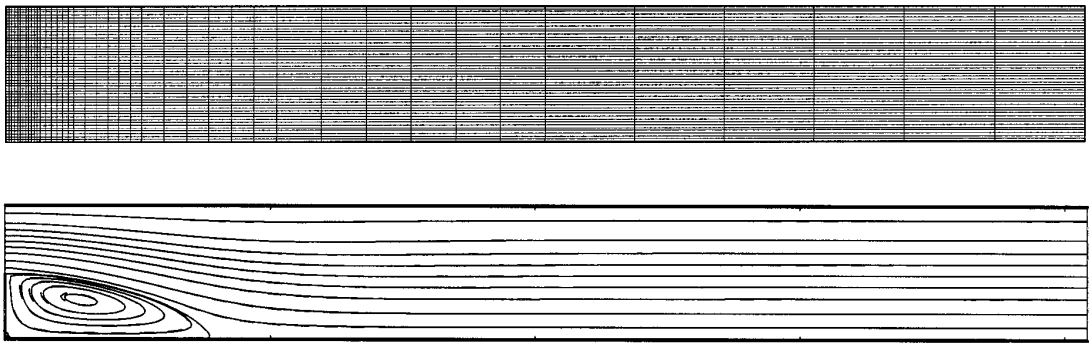


Figure 7: Channel flow simulated with the LBM on a non-uniform Cartesian mesh

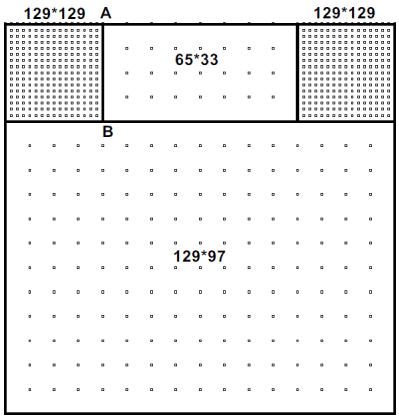


Figure 8: Multi-block method to realize local mesh refinement in the LBM

singularities appear at the two top corners, two finer mesh blocks are created to cover the areas near the corners, while, in the other region, coarse mesh blocks are utilized. However, in order to keep the solution continuous and smooth across the interfaces between different mesh blocks, special techniques must be invented to incorporate the lattices nodes on the interface, especially a node where multiple interfaces connect, such as the points A and B in Fig. 8.

Despite its better performance than the strict LBM, the loose LBM has an inherent defect, which is the direct result of the extra interpolation step. It was pointed out by Chen [11] that the conservation law on the microscopic level will be violated by the interpolation; therefore, unphysical dynamics will be added to the system. In addition, for the numerical aspect, a huge numerical viscosity will be generated by the interpolation, which will make it impossible to simulate high Reynolds number (Re) flows with a small viscosity. In order to essentially solve the mesh related issues (to handle the complex boundaries and realize the local mesh refinement), a fully unstructured mesh should be applied, instead of the structured mesh in the previous discussions. However, the utilization of the unstructured mesh will magnify the aforementioned negative effects caused by the interpolation step. By realizing this point, Chen [11] devised a volumetric approach to perform the interpolation. In this method, a control volume is created, in which the PDFs are averaged. Compared

to the early methods, in which the interpolation is point-wise, the new method can achieve exact conservation, but is still not able to decrease the numerical viscosity to a desired level.

In conclusion, in order to be able to handle a complex boundary and realize the local mesh refinement, a fully unstructured mesh is the only choice. With some effort, the LBM can incorporate the unstructured mesh, but with new problems introduced. Therefore, the loose LBM is not the ultimate solution. By revisiting Tab. 2, it can be seen that the loose LBM is devised by modifying the S-C procedure and loosening the MSC condition, which are, however, not the fundamental aspects of the LBM. The foundation feature that distinguishes the LBM is its Lagrangian nature. Therefore, in order to integrate the unstructured mesh without any major side effects, it is necessary to completely abandon its Lagrangian characteristic, which will automatically bring the LBM back to the DBM.

1.2.3 The LBM or the DBM

Continuing from the previous conclusion, the natural choice will be the DBM instead of the LBM. Even though the suggestion of using the DBM is mostly motivated by the demand of incorporating the unstructured mesh, there are also other reasons to select the DBM from the perspective of modern CFD tool characteristics, listed as follows:

- A modern CFD tool usually solves a Eulerian equation, not a Lagrangian one. As shown in Tab. 2, only the DBE is Eulerian;
- A modern CFD tool solves a partial differential equation (PDE). Then, solving each term in a PDE becomes independent of the others. Also, the numerical methods for PDEs have been developed not only by engineers but also by mathematicians for decades. Therefore, there are many different methods to choose for each term. As a result, the functionality of the whole package is modularized and can be easily extended, which is why the modern CFD tool has become so powerful. Obviously, only the DBE is a PDE.

In addition, there is a very important bonus in solving the DBE. As mentioned many times, by solving the DBE, the discretization of the velocity space and the discretization of the configuration space can be decoupled. Therefore, the choice of the lattice models and the choice of the mesh can be completely independent. The impact of this is two-sided. First,

the mesh can be arbitrary, which has already been addressed; second, the lattice model can be arbitrary as well. Another major difficulty encountered by the LBM is that its thermodynamic modeling is not satisfactory. However, it is suggested by Shan [93] that, by using higher-order lattice models, the thermodynamics can be modeled accurately. Therefore, the benefits of solving the DBE are not only the capability of using the unstructured mesh to handle the complex boundaries and realize the local mesh refinement, but also the ability of applying different lattice models to simulate different physical systems. Due to the limited length of the current work, however, the benefit of using different lattice models on the DBE will not be studied here.

It is also necessary to suggest that the purpose of the current work is not to replace the LBM with the DBM, but to develop a suitable tool for the problems that the LBM cannot tackle equally well. By solving the DBE, the simple S-C mechanism will be lost. Therefore, the computational efficiency of the DBM will be lower than the LBM. Consequently, for the problems that involve a simple geometry, have no thermal effects, and require high computational efficiency, the LBM is still chosen over the DBM.

1.3 OBJECTIVES AND THE SCOPE OF STUDY

The goal in the current work is to develop a numerical method that solves the DBE, which can be categorized into three classes: the finite difference discrete Boltzmann method (FDDBM), the finite element discrete Boltzmann method (FEDBM), and the finite volume discrete Boltzmann method (FVDBM). In each of these areas, a good deal of progress has already been made (see [39, 72, 97] for the FDDBM, [56, 57, 65, 107] for the FEDBM, and [35, 78, 79, 81–83, 86, 98, 100, 103, 109, 110, 115, 117] for the FVDBM)[2]. It can be seen, from the number of references, that the FVDBM has drawn the most attention, due to its geometric flexibility and built-in conservation (a nice feature for flow simulations). Therefore, the goal in the current work is to develop a FVDBM tool that consists of:

[2]In some publications, there is still some ambiguity in the use of the proper terminology. For example, some so-called FVLBMs are actually FVDBMs since the equation solved is the DBE, not the LBE.

- *The triangular unstructured mesh* that can be seamlessly used by the FVDBM approach;
- *The numerical algorithms and solution techniques* of the FVDBM solver that should be able to solve both the hydrodynamic and/or thermodynamic system and meet certain requirements of accuracy;
- *The boundary treatment* that should be able to realize different types of hydrodynamic and thermodynamic boundary conditions on flat and curved boundaries with desired accuracy.

Also, it is important to define the scope of current study, in which the limitations and requirements are listed as follows:

- The mesh is triangular type only, other types of unstructured mesh will not be studied;
- The collision model is BGK only, other collision models will not be discussed;
- The thermal model is passive scalar (PS) only; other thermal models including entropy and multi-speed (MS) models will not be included;
- Only the FVDBM will be developed; other DBM approach such as FDDBM and FEDBM will not be developed or discussed;
- The physical problems to be studied are restricted to simple flows that are single-phase and low Re and Kn; other complex flow will not be studied.

2.0 THE TRIANGULAR UNSTRUCTURED MESH

A mesh is very important for the simulation of any PDE because it not only can determine how accurately a PDE can be computed in the configuration space, but it also plays a vital role in the computational efficiency, memory usage and solution techniques. A mesh serves two purposes during the simulations, a front-end purpose and a back-end one. On the front end, a mesh visually discretizes the configuration space and renders the computational domain. Such a discretization is called the mesh generation. On the back end, a mesh is a database that provides all needed geometric information to the solver that solves the PDE. Therefore, the front end of a mesh can be seen as a visual representation of its back end. Once the mesh generation is done, the front end is fixed; however, the back end may have different versions depending on the nature of the PDE to be solved and the properties of the numerical algorithms and solution techniques of the solver. Therefore, a complete mesh preparation consists of two steps, the mesh generation and the construction of mesh database, as shown in Fig. 9.

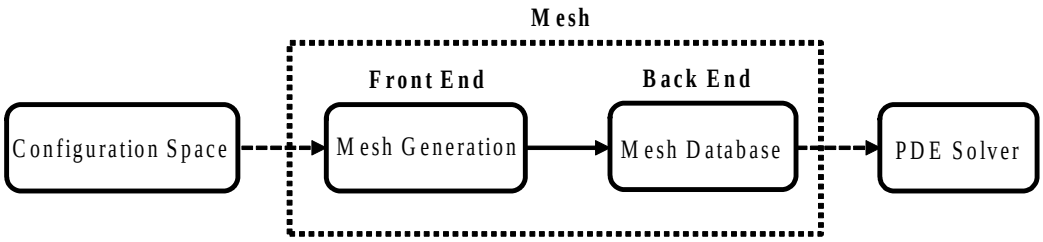


Figure 9: The mesh preparation process

2.1 THE MESH GENERATION

As mentioned in the previous chapter, in order to easily capture complex boundaries and have local mesh refinement at the same time, there is no better choice than the unstructured mesh. For 2D problems, there are many different types of unstructured meshes in terms of mesh shapes, such as triangular, quadrilateral, polygonal and even mixed type. However, the triangular mesh has become the number one choice of many successful CFD codes for three reasons. First, on the front end, it has the best adaptivity to all variety of different complex geometries; second, on the back end, it can be easily adopted by many PDE solvers and solution techniques with a good balance between computational efficiency and accuracy; third, on top of that, it has the simplest structure and topology than other types of unstructured meshes. As a result, only the triangular unstructured mesh is applied in the current study while leaving other types of meshes out of the discussion.

There are two types of triangular meshes used in the current study. The first type of mesh consists of all isosceles right triangles (IRT) with fixed topology and identical size and shape. The mesh consists of many squared blocks that have four triangles in each of them. For the one shown in Fig. 10 (a), there are nine blocks in each direction, so it is a 9×9 IRT mesh with a total of 9×9×4=324 triangular cells. The other type of mesh (Fig. 10 (b)) is more general with random topology and arbitrarily shaped and sized triangles, and therefore can achieve exact body-fitting for complex boundaries. The first difference between these two types of meshes is the size and shape of each triangular cell, which can be obviously seen in Fig. 10. The second difference is the local connectivity, or topology, at each node. For the IRT mesh, eight triangular cells share each interior node, and four are connected to each boundary node except for the corner nodes (two triangular cells). For the general mesh, the number of triangular cells connected to each node, no matter where it is located, is arbitrary. Therefore, the IRT mesh is only a special case of the general mesh. However, on the back end, the databases and data structures of these two meshes are exactly the same (see Sec. 2.2).

The IRT mesh is not able to capture a curved boundary as the general mesh; however, it is still necessary in the current study. A mandatory test for any numerical method is the

(a) IRT (b) General

Figure 10: Two types of triangular meshes

grid convergence study, which is used to characterize the order of accuracy of that numerical method. During the grid convergence study, the mesh size is refined by a fixed factor (usually two) for several times and the errors of the numerical solutions are measured after each refinement. With the general mesh, it is very difficult, if not impossible, to increase the mesh resolution with a fixed factor. Even though that is achievable, there is no guarantee that the mesh topology after each refinement will be kept the same. Therefore, both types of triangular meshes are provided in the current work, during which the IRT mesh is mostly used for the grid convergence study of a certain numerical algorithm and the general mesh is used for other numerical cases. Due to the natural difference between the IRT mesh and the general mesh, different algorithms of mesh generation are required, which are presented in the following two subsections.

2.1.1 The IRT mesh generation tool

Due to its fixed topology, it is very easy to generate the IRT mesh, since the locations of all nodes are predetermined if the node spacings on the boundaries are known. The algorithm of constructing the IRT mesh consists of two steps. The first step is to vertically and horizontally discretize the rectangular domain into square blocks by using the node

spacings on the boundaries. Taking the IRT mesh in Fig. 10 (a) as an example, there are ten equally-spaced nodes on each boundary (vertical and horizontal), which will cut each boundary into nine segments. By connecting these boundary nodes with only vertical and horizontal lines, a total of $9 \times 9 = 81$ square blocks will be generated. A interior node will be created at each location where a vertical line crosses a horizontal line. The second step is to insert a new interior node at the geometric center of each square block. Then, each square block is decomposed into four isoceles right triangles by connecting the center node with each node at the corners of the square block. Finally, the original square domain is meshed with a total of $4 \times 81 = 324$ triangles.

2.1.2 The general mesh generation tool

The algorithm for generating the general triangular mesh is much more complicated than that of the IRT meshing tool, since the mesh system has extremely large degrees of freedom. The mesh generation itself is one of the subdivisions of applied mathematics, and therefore can be a stand-alone topic. However, considering that the mesh generation technique is not the theme of the current study, it is convenient to choose one from the existing triangular meshing tools. The one that fits the purpose of the current study well is a two-dimensional triangular mesh generator called *Triangle*[1] that uses Delaunay triangulation, which is considered the best triangular meshing algorithm to maintain good mesh quality [29, 69]. However, extra components have to be developed and integrated with *Triangle* to include more functionality to improve the mesh quality. Two extra components are designed to work with *Triangle*: the Pre-Processor and Post-Processor. The structure of final mesh generation tool for the the general mesh is shown in Fig. 11, in which the arrows indicate the direction of information flow as well as the work flow.

2.1.2.1 Pre-Processor The Pre-Processor has two functions. The first one is to provide the location information of the boundary nodes to *Triangle*. During the meshing process, *Triangle* will keep the locations of given nodes fixed as a constraint and mesh the area

[1]For more information about *Triangle* and Delaunay triangulation, please refer to the publications [94,95] and the website *http://www.cs.cmu.edu/~quake/triangle.html*.

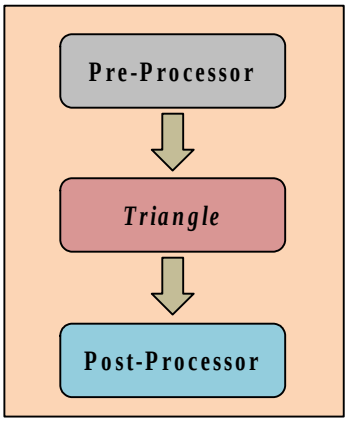
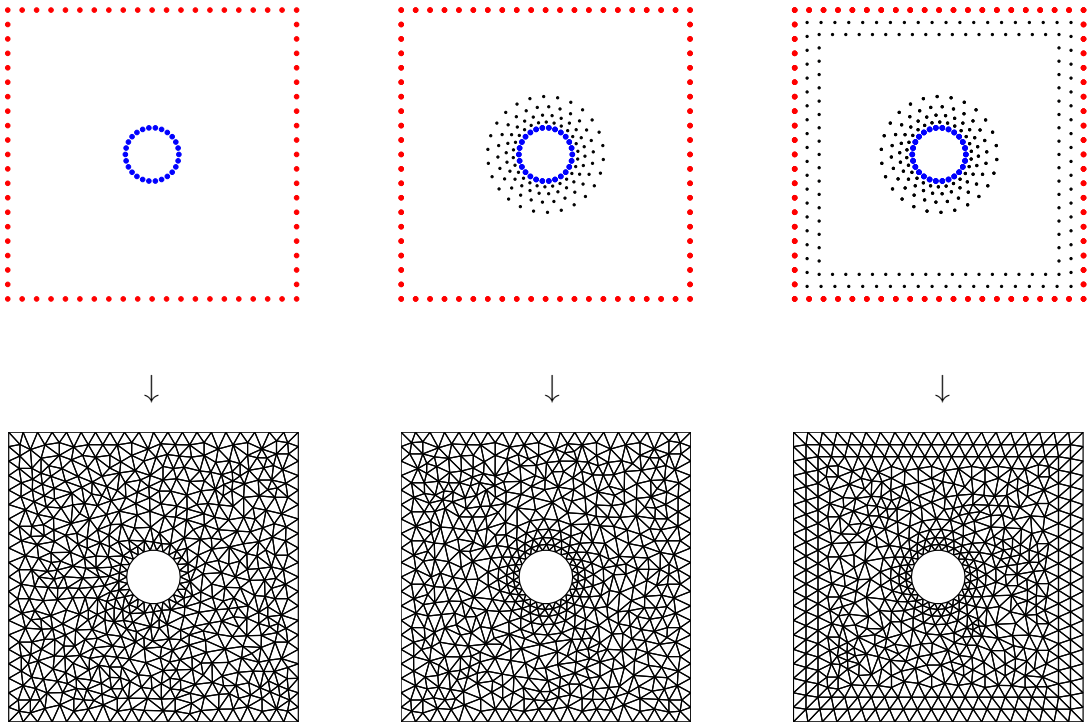


Figure 11: The structure of the *Triangle*-based mesh generation tool for the general triangular mesh

enclosed by the boundary nodes. The second function of Pre-Processor is to help *Triangle* generate a better mesh near boundaries by providing the location information of the interior nodes near the boundaries. Usually in CFD simulations, a refined mesh with good qualities is required in the near-boundary region where high gradients are often expected. A common practical approach to generate such a fine mesh, and keep the total number of grids low at the same time, is to construct high-quality grids that are coated to the boundaries layer by layer. Such layered grids act as a buffer zone that smoothly transition from the coarse-grid region to the fine-grid region. However, *Triangle* does not have such a built-in functionality to generate a layered mesh. Therefore, in addition to the boundary nodes, the ability to generate a group of layered interior nodes near the boundaries is embedded into Pre-Processor. Fig. 12 shows several node files generated by Pre-Processor and the corresponding triangular meshes yielded by *Triangle* for a rectangular domain with a circular immersed boundary, where the first row shows the node files and the second row the final meshes. From the node files, it can be seen that the node spacing on the outer rectangular boundary is looser than the immersed boundary. Therefore, a good mesh should have a smooth change in grid size between the inner and outer boundaries. However, in Fig. 12 (a), *Triangle* performs an abrupt change in

grid size right on the immersed boundary by stretching some triangles to become spike-like. In contrast, in Fig. 12 (b), by inserting four layers of nodes around the immersed boundary, the final mesh has a much smoother transition of grid size, and the triangles within the layers are all well shaped. The layered grids can also be generated for the outer boundary. In Fig. 12 (c), two layers of nodes are inserted for the outer boundary, which gives a final mesh that has two layers of good-quality grids coated to the outer boundary. The function of generating a layered mesh for the immersed or outer boundary can be switched on and off independently in Pre-Processor. In addition, Pre-Processor automatically determines how many layers of grids will be generated for the immersed and outer boundaries to achieve the optimal grid size transition, without an overlap between the two.

2.1.2.2 Post-Processor The only purpose of the Post-Processor is to improve the shape-wise quality of the mesh generated by *Triangle*. *Triangle* provides the mesh refining tool, which allows the users to divide large triangles into smaller ones. This will improve the mesh quality size-wise; however, the shape-wise quality, which affects the accuracy of numerical simulations more significantly, cannot be changed. By checking the mesh generated by *Triangle*, it can be easily seen that some triangles do not have desired shapes, which provides the opportunity to improve the mesh quality without creating or deleting any triangle. A simple algorithm can finish the task very efficiently, in which each interior node (boundary nodes have to be fixed to preserve the boundary geometry) is moved along a direction for a certain distance. Such a movement will change the shapes of all triangles connected to the moved node while preserving the local topology. The moving direction and distance are well calculated so that the node movement can ensure the quality improvement of the worst triangle connected to the moved node without deteriorating the average shape-wise quality of all connected triangles. Despite its simplicity and efficiency, such an algorithm will fail in one scenario: when the moved node has four totally connected triangles, because the movement along any direction will worsen the average shape-wise quality of all connected triangles. Therefore, the Post-Processor improves the mesh quality by sequentially executing two different algorithms. First, it surveys the entire mesh to locate all of nodes that have



(a) (b) (c)

Figure 12: The layered mesh created by Pre-Processor (red dots - outer boundary nodes; blue dots - immersed boundary nodes; black dots - interior nodes)

(a) no layered mesh; (b) layered mesh around immersed boundary; (c) layered mesh near the immersed and outer boundaries

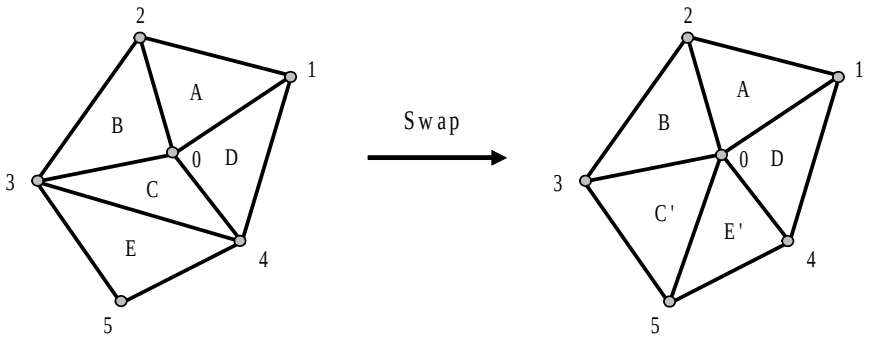


Figure 13: The swapping algorithm of changing a four-triangle structure to a five-triangle structure

four connected triangles and alter the four-triangle structure to be a five-triangle structure. Second, it moves all interior nodes to improve the shape-wise quality globally. The detailed definition of each algorithm is presented in the following parts.

First, the change from a four-triangle structure to a five-triangle structure is realized by an algorithm called swapping. The whole process is illustrated in Fig. 13, in which the node *0* is originally connected to four triangles, A, B, C and D. In order to swap, a neighbor triangle is needed. Any neighbor triangle of A, B, C or D can suffice; however, the best choice is the triangle with the largest angle at node *0*. So, in Fig. 13, the triangles to be swapped are triangle C and its neighbor E. After swapping, the face *34* is removed and the new face *05* is created. Then, the old triangle pair C and E become the new pair C' and E'. Once the swapping is done, the local topology should be updated to avoid any logic error.

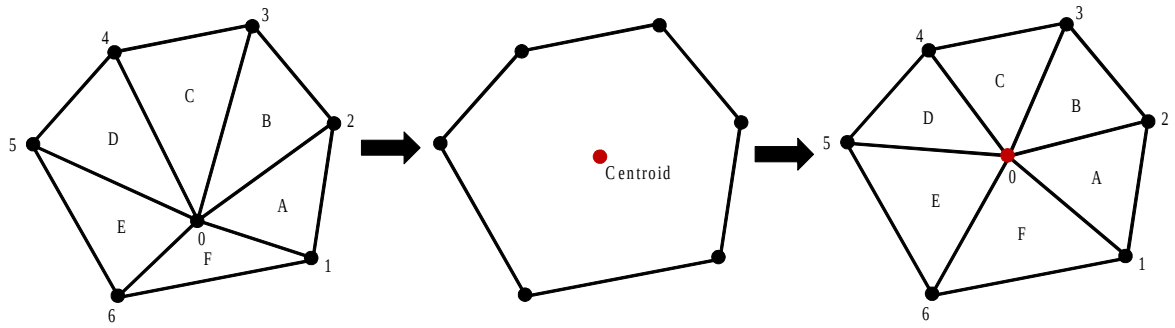
Once each interior node has at least five connected triangles, the second algorithm of Post-Processor is activated to globally improve the shape-wise quality of the mesh. As mentioned before, the idea is to move an interior node in order to, on average, improve the qualities of all triangles connected to that node. Therefore, a special algorithm is required to determine along which direction the node should be moved and how long the moving distance should be. In order to illustrate the algorithm developed in the current work, several concepts and terminologies should be built in advance. When an interior node is moved, all of, and only,

the triangles connected to the node will have their shapes affected. Such a group of triangles is called a star and each triangle is a star triangle. Fig. 14 (a) shows a star with six star triangles from A to F. The node to be moved is located inside of the star, and therefore it is called the center node (e.g. the node *0* in Fig. 14 (a)). All other nodes are satellite nodes (e.g. the nodes from *1* to *6* in Fig. 14 (a)). All star triangles form a polygon by omitting the internal faces. Such a polygon is called the star polygon, as shown in Fig. 14 (b). The algorithm works with a sequence of actions. First, with the given center node, find the star triangles and satellite nodes, as shown in Fig. 14 (a). Second, construct the star polygon and calculate the geometric centroid of the star polygon, as shown in Fig. 14 (b). Third, move the center node to the centroid of star polygon and reconstruct all star triangles, as shown in Fig. 14 (c). Such an algorithm is very efficient, since it can finalize the moving direction and distance at the same time with few simple calculations. The algorithm should be globally carried out for other interior nodes, since every center node for its own star is also the satellite node for some other stars. In addition, the algorithm is carried out in an iterative fashion, which is terminated by a exit condition that the global shape-wise mesh quality has met a self-defined criteria. Two factors are commonly used to quantitatively measure the shape-wise quality of a triangular mesh: the aspect ratio and skewness. For any triangle, the lengths of three faces are L_1, L_2 and L_3, and the three angles are θ_1, θ_2 and θ_3, measured in degrees. The aspect ratio of a triangle is the ratio of the lengths of the longest face to the shortest, as shown in Eq. (2.1). The skewness of a triangle is calculated by Eq. (2.2), where θ_r is the reference angle in the perfect case, which is 60° degrees for triangles. The perfectly shaped triangle (an equilateral triangle) has $Ar = 1$ and $Sk = 0$. When the quality of a triangle is becoming worse, both the aspect ratio and skewness will increase; however, they may not change at the same pace.

$$Ar = \frac{\max(L_1, L_2, L_3)}{\min(L_1, L_2, L_3)} \tag{2.1}$$

$$Sk = \max\left[\frac{\max(\theta_1, \theta_2, \theta_3) - \theta_r}{180 - \theta_r}, \frac{\theta_r - \min(\theta_1, \theta_2, \theta_3)}{\theta_r}\right] \tag{2.2}$$

In order to see the effectiveness of Post-Processor, the mesh in Fig. 12 (c) is improved with Post-Processor. The meshes before and after improvement are shown in Fig. 15, where Fig. 15 (a) is the same mesh in Fig. 12 (c). To see the improvement more clearly, the aspect



(a) (b) (c)

Figure 14: Three-step algorithm for improvement of the shape-wise mesh quality

ratio and skewness for each mesh are calculated with Eqs. (2.1) and (2.2), which are listed in Tab. 3. It can be seen that both the aspect ratio and skewness have been decreased with Post-Processor for the worst triangle as well as the entire mesh.

2.2 THE MESH DATABASE AND DATA STRUCTURE

Once the mesh generation is done, the needed geometric information should be created, organized and provided to the PDE solver. The ensemble of all needed geometric information is called the mesh database and the way it is organized is called the data structure.

2.2.1 The database overview

What information is stored in the database is determined by the type of PDE solver and all its computational algorithms and techniques. For example, different solvers, FE or FV, require different databases, and different numerical schemes of the solver will need different pieces of geometric information. A detailed database cannot be built unless the exact forms of the solver and all of its numerical algorithms are defined, which, however, will only be

(a) before Post-Processor (b) after Post-Processor

Figure 15: Mesh quality improvement by Post-Processor

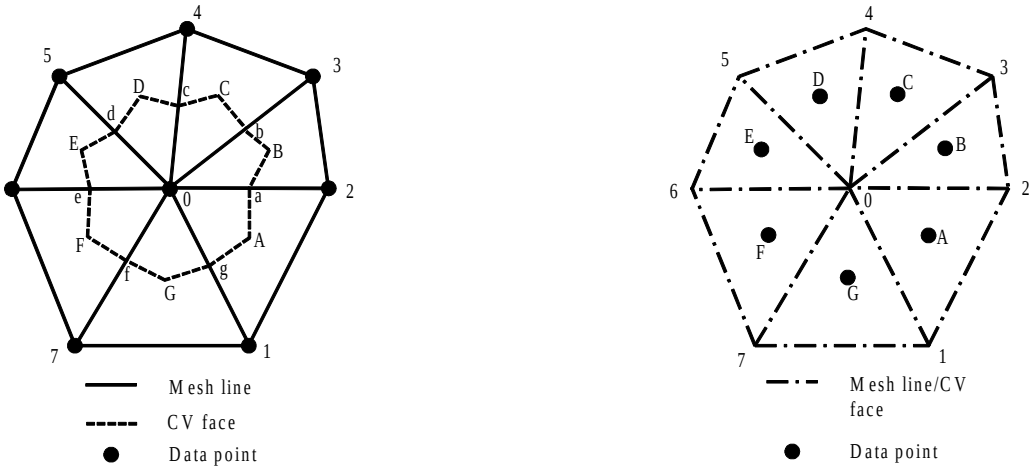
Table 3: Aspect ratio and skewness for the meshes in Fig. 15

	Aspect ratio		Skewness	
	worst triangle	global	worst triangle	global
Fig. 15 (a)	1.8316	1.2531	0.4496	0.1662
Fig. 15 (b)	1.6577	1.1869	0.4056	0.1307
% of improvement	9.5	5.3	9.8	21.4

discussed in the next chapter. Therefore, only the basics about the mesh database are introduced in this chapter.

For any FV solver, a common concept called control volume (CV) is the key to the numerical schemes. For the triangular unstructured mesh right after mesh generation, two types of CVs can be constructed subsequently. Therefore, two types of back-end meshes diverge from the base mesh: the vertex-centered (VC) mesh (also referred as a node-centered mesh) and the cell-centered (CC) mesh. Due to the difference in the CV, the locations for storing and updating solutions (also known as data points) for the two types of meshes are also different. Figure 16 shows the CVs and data points for the VC and CC meshes respectively, where the dashed lines and dot-dashed lines represent the boundaries of the CVs, which are usually called the faces, and the black dots are the data points. In Fig. 16 (a), configuration space is discretized with triangles (i.e. the base mesh), where each triangle is formed with three mesh lines (the black continuous lines), which are called edges. The triangles' vertices (i.e. nodes), which are labeled numerically, are chosen to store and update solutions. This is why such a mesh is vertex-centered. In order to complete the FV formulation, a CV has to be constructed around each data point. For the VC mesh, the CV is constructed by repeatedly connecting the center points of the edges, which are labeled with the lower-case letters, and the centroids of cells, which are labeled with the capital letters. It can be seen in Fig. 16 (a) that the CV for the VC mesh is an irregular polygon and the number of faces (e.g. the face Bb, bC, etc.) is arbitrary, since the number of cells connected to the node *0* is arbitrary in the real case. Due to superposition of the polygonal CVs on top of the base mesh, the VC mesh is often refered as a *dual mesh*. Therefore, four databases are required for the VC mesh: CELL, NODE, EDGE and FACE. By contrast, the construction of the CVs for the CC mesh is much easier. It can be seen in Fig. 16 (b) that, since the cell centroids are chosen to be the data points, the CVs are naturally the triangular cells themselves. Therefore, the extra effort for constructing the CVs can be saved. In addition, there are only three databases in the CC mesh: CELL, NODE and FACE, which is the first reason why the CC mesh is chosen in the current study.

In terms of numerical performance, the debate about which type of mesh is better for the FV method has been ongoing since the first day the two types of meshes were created



(a) the VC mesh

(b) the CC mesh

Figure 16: The vertex-centered (VC) mesh and cell-centered (CC) mesh for the FV method

[24, 27, 69]. Basically, one mesh is slightly better than the other in one aspect, and at the same time worse than the other in another aspect. As a result, the two types of meshes are generally the same in terms of performance. For example, the CC mesh can generate slightly more accurate results than the VC mesh, but it has a stronger grid dependency than the VC mesh; the CC mesh has easier bookkeeping than the VC mesh, while it requires more computational overhead than its VC counterpart. However, the CC mesh is still chosen over the VC mesh in the current work for a second reason that the CC mesh causes less ambiguity and contradiction on the boundaries than the VC mesh, which will be discussed in detail in Sec. 3.1.

2.2.2 The data structure overview

In contrast to the database, the data structure is not subject to the PDE solver at all. However, a good data structure can let the solver easily access the data it needs, and therefore keep the overall simulation computationally efficient. For the structured mesh, such as the mesh for the FD method, the mesh data can be put into a matrix, and such a matrix

and configuration space of the problem have the same dimensions. For example, a two-dimensional problem can have its mesh represented by a two-dimensional matrix. On the contrary, such dimensional correspondence for the unstructured mesh is lost. No matter how many dimensions the physical problem has (2D or 3D), the mesh data is only 1D, which is true for the cell, node and face databases simultaneously. Therefore, the same layered structure is applied to all three databases. The first layer, the top layer, is a 1D container that sequentially holds each cell, node or face. The second layer is another 1D container for the detailed information of each cell, node or face.

2.2.3 The databases and data structures for cells, nodes and faces

Prior to describing the specific database and its structure for each geometric element, it is necessary to assemble several key concepts and their symbols, which are listed in Tab. 4.

2.2.3.1 The cell database and its structure The database and data structure for cells are shown in Fig. 17. In the cell database, each cell should have a unique global address. Given that all cells are identical in terms of data structure, it is convenient to use a number from 1 to D_c as the global address, I_c, for each cell. The data entry pointed by the address of each cell, nonetheless, is not a simple and single datum. Instead, it is an assembly of miscellaneous information stored in the second layer with each of its data entry pointed by a second-layer address, II_c. This is why I_c is also called the first-layer address. The information pointed by II_c has more actual meanings than that given by I_c. So, it is better to use a string that carries an intuitive meaning, instead of an integer number, as the value of II_c. For example, the datum pointed by nd_1 is the global address of the first node of the current cell, while the data pointed by *centrd* are the coordinates of the centroid of the current cell, which are floating numbers. As a result, all of the actual information about a cell is embedded in the second layer of its database. So, both I_c and II_c are needed in the function $CELL\,(I_c, II_c)$ in order to access the specific data about a prescribed cell. For example, if the solver needs to know the area of 10th cell, the function $CELL\,(10, a)$ can deliver the desired information.

Table 4: Nomenclature of database and data structure for 2D triangular mesh

Symbol	Meaning
$CELL$	The database for all cells
D_c	The total number of cells
I_c	The first-layer address or global address of the cell database
II_c	The second-layer address of the cell database
$NODE$	The database for all nodes
D_n	The total number of nodes
I_n	The first-layer address or global address of the node database
II_n	The second-layer address of the node database
$FACE$	The database for all faces
D_f	The total number of faces
I_f	The first-layer address or global address of the face database
II_f	The second-layer address of the face database

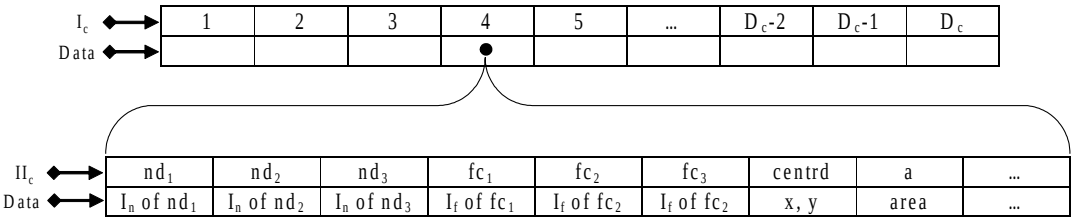


Figure 17: The cell database CELL and its structure

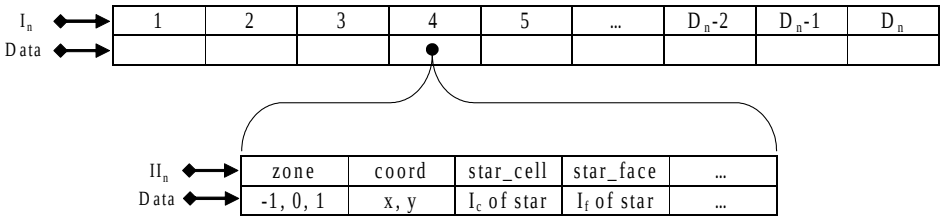


Figure 18: The node database NODE and its structure

2.2.3.2 The node database and its structure Similar to the cell database, the node data base is also constructed with a two-layer structure, as shown in Fig. 18. Each node is also uniquely pointed by the numerical first-layer address, I_n, varying from *1* to D_n. However, due to the natural difference between a cell and a node, the second-layer database of a node will have a different length, and the second-layer address II_n will bear different meanings. $NODE\,(I_n, zone)$ gives the zone identifier of a node, which can tell whether the node is on the immersed boundary, in the interior region, or on the outer boundary; $NODE\,(I_n, coord)$ provides the coordinates of the current node; $NODE\,(I_n, star\_cell)$ and $NODE\,(I_n, star\_face)$ generate the series of global addresses of all the cells and faces connected to the node in a star structure (e.g. Fig. 14), respectively.

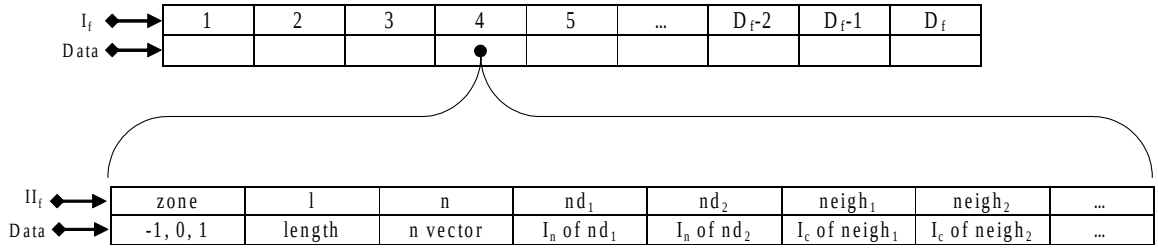


Figure 19: The face database FACE and its structure

2.2.3.3 The face database and its structure Similar to the cell and node databases, the face database copies the two-layer structure, which can be seen in Fig. 19. The global address, I_f, varying from *1* to D_f, uniquely locates each face. The second-layer address II_f points to a group of information related to a face. $FACE\left(I_f, zone\right)$ yields the zone identifier of a face, similar to a node; $FACE\left(I_f, l\right)$ and $FACE\left(I_f, n\right)$ provide the length and unit normal vector of a face; $FACE\left(I_f, nd_1\right)$ and $FACE\left(I_f, nd_2\right)$ generate the addresses of the two nodes at the ends of a face; and $FACE\left(I_f, neigh_1\right)$ and $FACE\left(I_f, neigh_2\right)$ recover the immediate neighbor cells that share the same face.

2.2.3.4 The merits of proposed databases and data structures With the developed data structures for cells, nodes and faces, the capability of the solver can be enhanced to the utmost extent with an excellent flexibility, which is twofold:

- The solver can access any data from any starting point. Since each database for cells, nodes or faces contains the global addresses of the other two, a closed access loop is formed among cells, nodes and faces. Therefore, by building an access route, which could be short or long, any cell, node or face can be accessed by another cell, node or face. For example, if we start from a face, which has two nodes at its ends, then we can access all of the cells connected to each of the two nodes. The global addresses of all those cells are the union of the star cells of each node, which can be expressed as $NODE\left(FACE\left(I_f, nd_1\right), star\_cell\right) \cup NODE\left(FACE\left(I_f, nd_2\right), star\_cell\right)$.
- The databases are flexible both horizontally and vertically. The horizontal flexibility means the length of a database in each layer is dynamic. This point is more significant for the second-layer database, since more information, if needed, can be easily stacked on the existing database with new second-layer addresses. The vertical flexibility means deeper layers of data can be added to the existing databases without affecting the data structure of the current and parent layers. It is possible that the data entry in the second-layer database is not an address, floating number or integer, but another group of assorted information with different types. Then it is necessary to create an extra layer of data for that entry. Currently, there are only two layers of data. However, if a much more advanced numerical algorithm is developed for the solver and a third or even

fourth layer of data is needed, the vertical flexibility will guarantee the easiest database expansion and minimum change to the solver. In summary, such a data structure sets almost no limitation on the capabilities of the algorithms of the solver.

3.0 OVERVIEW OF THE FVDBM SOLVER

Once the mesh, the first component of the FVDBM, is ready, the next task is to construct the FVDBM solver (called simply the solver hereafter), the second component of the proposed FVDBM, which can solve the DBE (Eq. (3.1)) on that mesh. The entire solver is the assembly of the numerical scheme for each term in Eq. (3.1), which are the temporal term, gradient term and collision term from left to right. In this chapter, the numerical scheme for each term will be briefly introduced while most of the detailed descriptions are saved for the following chapters (see Chap. 4 for the gradient term and Chap. 5 for the temporal term). However, since the collision term can be calculated very easily, its numerical scheme will be fully covered in this chapter.

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \frac{\partial f_i}{\partial \boldsymbol{x}} = \frac{1}{\lambda} \left(f_i^{eq} - f_i \right) \tag{3.1}$$

3.1 BACKGROUND AND RE-DISCUSSION OF THE MESH-RELATED ISSUES

The concept of using a FV approach in the general LBM applications was introduced by Benzi *et al.* in 1992 [7,75], and mathematically refined by Chen in 1998 [11]. However, both of their methods solve the LBE instead of the DBE. The first FVDBM was introduced by Peng *et al.* in a series of papers [81–83,109,110], which was further developed by Ubertini *et al.* [100,101,103]. Roughly at the same time, Rossi *et al.* developed the three-dimensional (3D) model [86]. However, all of these proposed FVDBM approaches were solved on the VC mesh. In addition, Stiebler [98] stated that the VC FVDBM experiences stability issues,

which will be explained later in Eq. (3.5). Patil *et al.* [79] introduced the first CC FVDBM model in 2009, followed by several other isothermal and thermal models [35, 78, 116, 117], among which the most recent work was published in 2013 [117].

As of now, the number of published CC FVDBM papers is much less than for its VC counterpart. However, the reason why the CC FVDBM is chosen in the current work is not its rarity, but two issues related to the VC mesh. The first issue is the instability caused by the VC mesh. Fig. 20 shows the CV (the shaded polygon) in a VC mesh, in which the CV is created around the vertex P. The point P is also the data point, since, within the CV, P is the most legitimate point to represent the entire CV. Therefore, the solution at P needs to be updated after every time step. During each time step, the total fluxes through the closed boundary of the CV need to be calculated, which are the sum of the flux through each face of the CV. Taking the face AB for example, in order to calculate the flux, the PDFs on AB should be evaluated. A good assumption is that the PDFs on the face can be represented by the PDF at the center point of the face, with a linear interpolation:

$$f_{AB} = \frac{f\left(\boldsymbol{x}_A\right) + f\left(\boldsymbol{x}_B\right)}{2} \tag{3.2}$$

At the same time, A is the center point between vertices P and P_1, and B is the centroid of the triangle $\triangle PP_1P_2$, so:

$$f\left(\boldsymbol{x}_A\right) = \frac{f\left(\boldsymbol{x}_P\right) + f\left(\boldsymbol{x}_{P_1}\right)}{2} \tag{3.3}$$

$$f\left(\boldsymbol{x}_B\right) = \frac{f\left(\boldsymbol{x}_P\right) + f\left(\boldsymbol{x}_{P_1}\right) + f\left(\boldsymbol{x}_{P_2}\right)}{3} \tag{3.4}$$

By substituting Eq. (3.3) and Eq. (3.4) into Eq. (3.2), we have:

$$f_{AB} = \frac{5}{12} f\left(\boldsymbol{x}_P\right) + \frac{5}{12} f\left(\boldsymbol{x}_{P_1}\right) + \frac{2}{12} f\left(\boldsymbol{x}_{P_2}\right) \tag{3.5}$$

It can be seen that the PDFs on face AB is symmetrically weighted by the PDFs on each side of the face. Therefore, Eq. (3.5) carries the nature of central difference (although may not completely), which is notorious for causing instability for the advection problems.

Of course, such an issue caused by the VC mesh can be alleviated by using some other schemes, such as the least squares linear reconstruction upwind scheme suggested by Stiebler [98]. However, another bigger issue cannot be avoided, and has not been reported in any publication so far. Since the vertices are used to store and update solutions

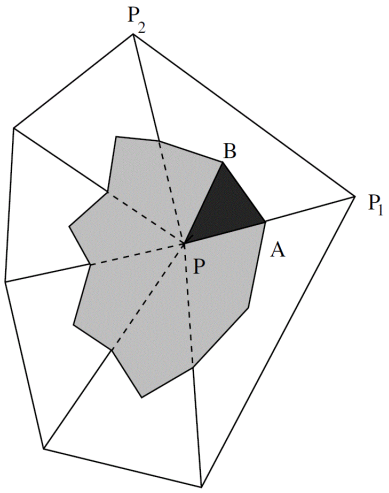
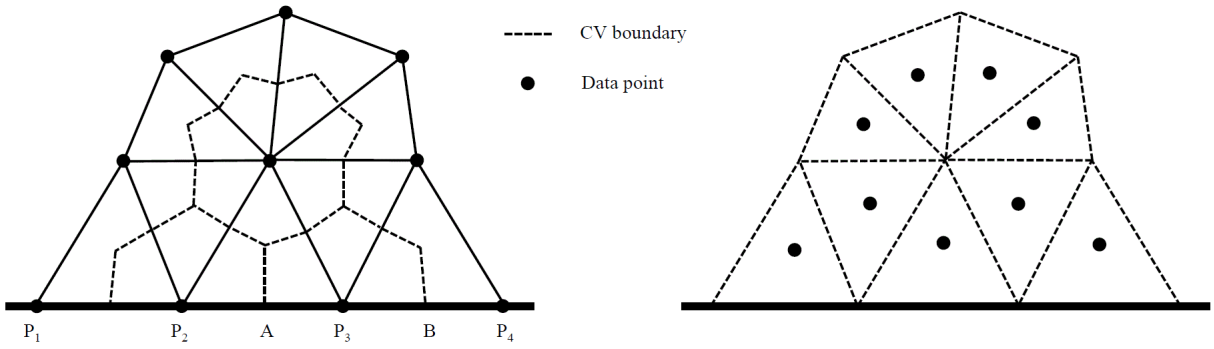


Figure 20: Flux evaluation for the control volume of vertex-centered mesh [98]



(a) vertex-centered (b) cell-centered

Figure 21: The control volumes of the VC and CC meshes

for the VC mesh, the vertices can be put into two groups: the internal vertices and boundary vertices. Due to the difference in location, the CVs constructed for different groups of vertices will be different. As shown in Fig. 21(a), for an internal vertex, the CV will form a closed segments that wrap the vertex (see Fig. 20); for the boundary vertex, the CV is not able to close in the 2π direction, and the vertex will be on the face of the half-sized CV. However, those boundary vertices should use the same governing equation to update their solutions as the internal vertices. Therefore, the same procedure of flux calculation should also be carried out for those half-sized CVs on the boundary. Taking the face P_3B in Fig. 21(a) for example, the PDFs on the face should be $f^{n+1}_{P_3B} = \left[f^n\left(\boldsymbol{x}_{P_3}\right) + f^n\left(\boldsymbol{x}_B\right)\right]/2 = \left\{f^n\left(\boldsymbol{x}_{P_3}\right) + \left[f^n\left(\boldsymbol{x}_{P_3}\right) + f^n\left(\boldsymbol{x}_{P_4}\right)\right]/2\right\}/2 = (3/4)\, f^n\left(\boldsymbol{x}_{P_3}\right) + (1/4)\, f^n\left(\boldsymbol{x}_{P_4}\right)$, in which the superscripts n and $n+1$ indicate the values at different time steps. So, as a result, the boundary solutions are updated by solely using other data points (P_3 and P_4 in this case) in the previous time step and never using the boundary conditions. Therefore, the entire system will not be subject to the boundary conditions and becomes a system freely evolving on its own. Mathematically speaking, such a system will not have a solution due to a lack of closure. On the other hand, if the boundary vertices are forced to take the boundary conditions, then they cannot be treated as data points. As a result, all of the half-sized CVs attached to the boundary will skip solving the governing equation. Consequently, the solution between the boundary vertices, whose values are anchored by the boundary conditions, and the internal vertices will witness an unexpected gradient or even discontinuity due to a layer of inactive CVs between them. In conclusion, the VC mesh will make its boundary vertices suffer solution contradiction, and therefore cause ambiguities when implementing the boundary conditions. This is probably why in the series of papers by Peng *et al.* [81–83,109,110], they never explicitly stated how to implement the boundary conditions. In contrast, for a CC mesh (see Fig. 21(b)), the CVs attached to the boundary will never experience a conflict with the boundary condition, since the data points, which are the centroids of the cells, will never be located on the boundary. Considering this, as well as the back-end numerical benefits of the CC mesh discussed in Sec. 2.2.1, the CC FVDBM is chosen for the current study.

3.2 THE FV FORMULATION OF THE DBE ON THE CELL-CENTERED MESH

The DBE with the BGK collision model is shown here again for convenience. It reads:

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i = \frac{1}{\lambda} \left(f_i^{eq} - f_i \right) \tag{3.6}$$

The left hand side of Eq. (3.6) is the temporal term and the gradient term, while the right hand side is the collision term, which is also the source term of the system. So mathematically speaking, the DBE with BGK collision is a hyperbolic PDE with a source term. However, the DBE has some unique features:

- The source term carries all non-linearity due to f^{eq}, but the non-linearity is local;
- The advection (the temporal term and the gradient term together), which is non-local, however, is linear because the advection velocity $\boldsymbol{\xi}_i$ is fixed (for the NS equation, the non-local term $\boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u}$ is also non-linear);
- The source term is usually stiff since its stiffness will increase when the relaxation time λ decreases.

The first two features make the DBE easier to solve compared to the NS equation. However, the third feature will lead to a typical stiffness effect that is similar in conventional hyperbolic equations, which will be discussed in Sec. 6.4.3 and Sec. 8.2.2.

In order to solve the DBE with the FV method, it is mandatory to perform a geometric integral over the CV, which is, for the CC mesh, the triangular cell itself. Then the integrated Eq. (3.6) over the CV with the terms slightly rearranged becomes:

$$\int_{CV} \frac{\partial f_i}{\partial t} dA = \int_{CV} \frac{1}{\lambda} \left(f_i^{eq} - f_i \right) dA - \int_{CV} \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i dA \tag{3.7}$$

The integrals of the temporal and source term can be simplified with the cell-averaged values inside of the CV. Therefore, these integrals become:

$$\int_{CV} \frac{\partial f_i}{\partial t} dA = \frac{\partial \bar{f}_i}{\partial t} A_{CV} \tag{3.8}$$

$$\int_{CV} \frac{1}{\lambda} \left(f_i^{eq} - f_i \right) dA = \frac{1}{\lambda} \left(\bar{f}_i^{eq} - \bar{f}_i \right) A_{CV} \tag{3.9}$$

where the overhead bar notation indicates the averaged value over the CV and A_{CV} is the area of the CV. Assuming point P is the centroid of the CV, the averaged values in the CV can be replaced by the values at the centroid. Therefore:

$$\bar{f}_i = f_i(P) \tag{3.10}$$

$$\bar{f}_i^{eq} = f_i^{eq}(P) \tag{3.11}$$

By using such relations, Eq. (3.8) and Eq. (3.9) then become:

$$\int_{CV} \frac{\partial f_i}{\partial t} dA = \frac{\partial \bar{f}_i}{\partial t} A_{CV} = \frac{\partial f_i(P)}{\partial t} A_{CV} \tag{3.12}$$

$$\int_{CV} \frac{1}{\lambda}\left(f_i^{eq} - f_i\right) dA = \frac{1}{\lambda}\left(\bar{f}_i^{eq} - \bar{f}_i\right) A_{CV} = \frac{1}{\lambda}\left[f_i^{eq}(P) - f_i(P)\right] A_{CV} \tag{3.13}$$

The integrated gradient term in Eq. (3.7) needs more mathematical manipulations. To begin with, for any vector $\boldsymbol{v}$ and scalar s, it holds that:

$$\boldsymbol{v} \cdot \boldsymbol{\nabla} s = \nabla \cdot (s\boldsymbol{v}) - s(\nabla \cdot \boldsymbol{v}) \tag{3.14}$$

Applying Eq. (3.14) to the integrated gradient term in Eq. (3.7), we have:

$$\int_{CV} \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i dA = \int_{CV} \nabla \cdot (f_i \boldsymbol{\xi}_i) dA - \int_{CV} f_i (\nabla \cdot \boldsymbol{\xi}_i) dA \tag{3.15}$$

Once the lattice model is selected, the lattice velocity $\boldsymbol{\xi}_i$ stays constant. Therefore, in Eq. (3.15):

$$\nabla \cdot \boldsymbol{\xi}_i = 0 \tag{3.16}$$

As a result, Eq. (3.15) reduces to:

$$\int_{CV} \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i dA = \int_{CV} \nabla \cdot (f_i \boldsymbol{\xi}_i) dA \tag{3.17}$$

According to the Gauss' divergence theorem, Eq. (3.17) is equivalent to:

$$\int_{CV} \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i dA = \int_{CV} \nabla \cdot (f_i \boldsymbol{\xi}_i) dA = \oint_{\partial CV} f_i \cdot (\boldsymbol{\xi}_i \cdot \boldsymbol{n}_{\partial CV}) ds \tag{3.18}$$

where ∂CV is the boundary of the CV and $\boldsymbol{n}_{\partial CV}$ is the outward normal of the boundary. Eq. (3.18) is, physically speaking, the total flux of the PDFs through the boundaries of the CV. Therefore, Eq. (3.18) can be further reduced to:

$$\int_{CV} \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i dA = \int_{CV} \nabla \cdot (f_i \boldsymbol{\xi}_i) dA = \oint_{\partial CV} f_i \cdot (\boldsymbol{\xi}_i \cdot \boldsymbol{n}_{\partial CV}) ds = \sum_{\alpha=1}^{S} F_{i,\alpha} \tag{3.19}$$

where $F_{i,\alpha}$ is the flux across the αth face of the CV; S, which is the total number of faces of the CV, is three for the triangular CC mesh. By replacing each term in Eq. (3.7) with Eq. (3.12), Eq. (3.13) and Eq. (3.19), the original DBE spatially discretized with the CC triangular mesh will have its finite volume formulation as simple as:

$$T_i(P) = C_i(P) - F_i(\partial P) \tag{3.20}$$

where T_i is the temporal term, C_i is the collision term and F_i is the flux term with ∂P indicating the boundary of the CV enclosing the current centroid P:

$$T_i(P) = \frac{\partial f_i(P)}{\partial t} \tag{3.21}$$

$$C_i(P) = \frac{1}{\lambda}\left[f_i^{eq}(P) - f_i(P)\right] \tag{3.22}$$

$$F_i(\partial P) = \frac{1}{A_{CV}} \sum_{\alpha=1}^{3} F_{i,\alpha} \tag{3.23}$$

In Eq. (3.20), the subscript i does not drop, which indicates that the PDF in each direction of the lattice should have its own governing equation. The numerical schemes to treat Eq. (3.21) and Eq. (3.23) are called the time marching schemes and flux schemes, respectively. In the following sections, the overviews of the time marching and flux schemes will be given, as well as the straightforward calculation of the collision term, Eq. (3.22). The detailed definitions of time marching and flux schemes will be discussed in the next two chapters.

3.3 EVALUATION OF THE COLLISION TERM

Equation (3.22) gives the value of the collision term at the centroid P of the current cell. However, as discussed in Sec. 1.1.2, the evaluation of the equilibrium PDF depends on the applied lattice model. So, generally speaking, the equilibrium PDF is a function of the lattice model $\boldsymbol{\Upsilon}$ and the macroscopic variables $\mathfrak{R}$. Therefore, Eq. (3.22) becomes:

$$C_i(P) = \frac{1}{\lambda}\left\{\Phi_i\left[\boldsymbol{\Upsilon}, \mathfrak{R}(P)\right] - f_i(P)\right\} \tag{3.24}$$

If a D2Q7, D2Q9 or D2Q13 lattice model is applied, $\boldsymbol{\Upsilon}$ and the function $\Phi_i()$ are defined according to Eqs.(1.15), Eqs.(1.16) or Eqs.(1.17). $\mathfrak{R}(P)$ is the set of macroscopic variables at the centroid P. By combining Eqs. (1.8) into one, $\mathfrak{R}(P)$ is defined as:

$$\mathfrak{R}(P) = \begin{bmatrix} \rho(P) \\ \boldsymbol{u}(P) \\ T(P) \end{bmatrix} = \sum_{i=0}^{M-1} \begin{bmatrix} f_i(P) \\ \boldsymbol{\xi}_i f_i(P)/\rho(P) \\ (\boldsymbol{\xi}_i - \boldsymbol{u}(P))^2 f_i(P) \times (2/(D_0 R))/\rho(P) \end{bmatrix} \tag{3.25}$$

For an isothermal model, this reduces to:

$$\mathfrak{R}(P) = \begin{bmatrix} \rho(P) \\ \boldsymbol{u}(P) \end{bmatrix} = \sum_{i=0}^{M-1} \begin{bmatrix} f_i(P) \\ \boldsymbol{\xi}_i f_i(P)/\rho(P) \end{bmatrix} \tag{3.26}$$

3.4 EVALUATION OF THE FLUX TERM

Since the flux calculations on all faces of the CV are identical, the subscript α in $F_{i,\alpha}$ for face index can be omitted. Therefore, the flux for any face is calculated as follows:

$$F_i = \bar{f}_i (\boldsymbol{\xi}_i \cdot \boldsymbol{n}) L \tag{3.27}$$

where $\bar{f}_i$ is the average PDF of velocity $\boldsymbol{\xi}_i$ on the current face, with L the length, and $\boldsymbol{n}$ the unit outward normal of the face, respectively.

Since L and $\boldsymbol{n}$ are immediately available once the mesh is generated, and $\boldsymbol{\xi}_i$ is predefined in the lattice model selected by the users, the only unknown in Eq. (3.27) is $\bar{f}_i$. Therefore, the problem of calculating the facial flux is transformed into the problem that how $\bar{f}_i$ is evaluated. Figure 22 shows the stencil for calculating $\bar{f}_i$ on one of the face (the face V_2V_3) of the cell P whose three vertices are V_1, V_2 and V_3, along with its neighbor on that face, cell Q (points P and Q are the centroids of two triangular cells, respectively). Point C is the geometric center point of the face. The dashed line that is passing through C and orthogonal to the face V_2V_3 is the stencil line. Points A and B are the projected points of centroids P and Q, respectively, onto the stencil line. Therefore, they are called the stencil points, and

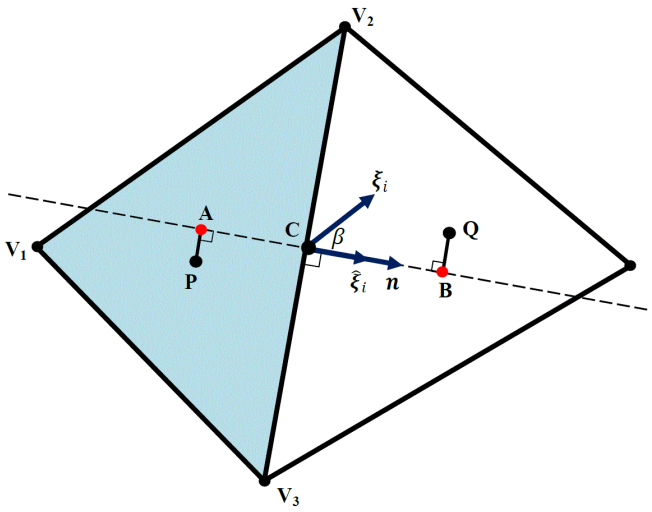


Figure 22: Two-point stencil for flux calculation

the stencil shown in Fig. 22 is a two-point stencil. Since C is the center point of the face, it can be assumed that:

$$\bar{f}_i = f_i(C) \tag{3.28}$$

Now, the problem is further transformed to how to calculate the PDFs at the face center C, with the given two stencil points A and B in Fig. 22. A seemingly good choice is the central scheme that assumes $f_i(C) = [f_i(A) + f_i(B)]/2$, which is second-order accurate in nature. However, such a scheme is unconditionally unstable. The task in the next chapter is to systematically develop several numerical algorithms that stably calculates $f_i(C)$ on a universal stencil, with desired level of accuracy. Those numerical algorithms are called flux schemes in the current work although they do not directly calculate the fluxes. The flux schemes are the utmost important ingredient of the overall FVDBM model and also a key topic in this work.

Another important task in the completion of calculating $f_i(C)$ is the evaluations of the PDFs at the stencil points; for example, points A and B in Fig. 22. Since A and B are not data points, $f_i(A)$ and $f_i(B)$ are not readily available. However, due to the fact that A and B are the feet of the data points P and Q onto the stencil line, respectively, on the stencil

line, there are no other points that are located closer to P and Q than A and B. Therefore, it can be assumed that:

$$f_i(A) = f_i(P), \quad f_i(B) = f_i(Q) \tag{3.29}$$

Equation (3.29) is equivalent to the assumption that the PDF distribution within each triangular cell is constant, which however may not be the real case. Therefore, Eq. (3.29) will introduce an error that is diffusive in nature. A more accurate treatment is to assume a linear 2D PDF distribution within each cell. However, such a treatment is very complicated to realize, and therefore will not be included in the current work. On the other hand, the error introduced by Eq. (3.29) will die down rapidly when the mesh is refined and mesh quality is improved. As a result, Eq. (3.29) is good enough. Therefore, it is the only assumption to evaluate the PDFs at the stencil points in the present work.

There are three important properties of any flux scheme: stability, accuracy, and computational efficiency, which will be studied in detail for each presented flux scheme in Chap. 4.

3.5 TIME MARCHING

Unlike the flux schemes for the flux term (Eq. (3.23)), which is calculated separately from other terms in Eq. (3.20), the time marching schemes do not calculate the temporal term itself, Eq. (3.21). Instead, the time marching schemes handle the entire governing equation (Eq. (3.20)) and evolve it in time. By combining Eq. (3.20) and Eq. (3.21), we have:

$$\frac{\partial f_i(P)}{\partial t} = C_i(P) - F_i(\partial P) \tag{3.30}$$

With a proper time marching scheme, the term on the left hand side of Eq. (3.30), which is still continuous in the temporal space, will be discretized. For example, with the forward Euler scheme, Eq. (3.30) becomes:

$$f_i^{n+1}(P) = f_i^n(P) + \Delta t \left[C_i^n(P) - F_i^n(\partial P)\right] \tag{3.31}$$

where the superscript $n+1$ and n indicates the current and previous time step, respectively. However, Eq. (3.31) is only first-order in time, which will introduce much numerical dif-

fusivity, and suffer instability. Therefore, in Chap. 5, several second-order time marching schemes will be developed and discussed in detail.

Due to the transient nature of the DBE, the time marching schemes are extremely important. Even though only a steady-state solution is required, the simulation of the DBE still has to be carried out in a time-evolving fashion since there is not a steady-state version of DBE. During such a transient simulation, three things will be affected by a time marching scheme: the stability, the accuracy and the computational efficiency, similar to the flux schemes. In Chap. 5, the accuracy and computational efficiency of each time marching scheme will compared against each other.

3.6 OVERVIEW OF THE STUDY APPROACH FOR THE FVDBM SOLVER

Due to the simplicity of the collision term, the study of the FVDBM solver is mainly the studies of the flux schemes and time marching schemes, which will be carried out in a similar style in the next two chapters. First, all presented schemes will be developed and formulated one by one. Then, the stability of each scheme will be investigated, followed by its numerical viscosity and computational efficiency. At last, each scheme will be commented based on its comparison with other schemes in the same category. One or several schemes will be recommended for different scenarios as well.

3.6.1 Stability and time step size

The ultimate question for the stability analysis of any flow simulation with any type of solver is what the maximum allowable time step size, Δt_{max}, is, to maintain a stable solution. This question is extremely important to FVDBM due to the transient nature of the DBE. A larger Δt_{max} allows the overall solver to converge faster with fewer iterations. On the contrary, if Δt_{max} is too small, the solver will need much more iterations to reach a desired state. Therefore, such a solver should not be accepted even though it is very accurate,

due to its tremendous computational overhead. To answer this question, two approaches can be adopted: an analytical one and a numerical one. The former one is based on the analytic investigation of every numerical ingredient of the overall model, with complicated mathematical tools such as Fourier Transformation. For the FVDBM, there are basically three numerical ingredients that can affect the stability property of the overall model: the flux scheme, the time marching scheme and the boundary treatment. It might be feasible to analytically study the stability property of each numerical ingredient, but it will be basically impossible to study stability analytically when all of them are intertwined together. The latter approach is very straightforward. By numerically testing different Δt to see if the solution is stable or not in a trial-and-error fashion, Δt_{max} can be obtained. The only downside about this approach is the resolution issue. The obtained Δt_{max} is not exact but within a range with selected resolution.

For the simulations using the DBE with any numerical method, Δt_{max} is determined by two stability criteria. The first one, which makes the DBE simulations unique, is the stiffness effect caused by the collision term that has been mentioned in Sec. 3.2. Physically speaking, the stiffness effect establishes a constraint that Δt should be comparable in size to the relaxation time λ in order to capture the correct collisional physics. Usually, it is given as [86, 103]:

$$\Delta t < 2\lambda \tag{3.32}$$

So, a parameter ω is devised to describe such a collision-related stability property, which is defined as:

$$\omega = \frac{\Delta t}{\lambda} \tag{3.33}$$

Another criteria, which is common in the conventional CFD simulations, is the Courant-Friedrichs-Lewy (CFL) condition. The CFL condition is directly related to the advection. Therefore, a second parameter r can be used to describe such an advection-related stability property, which is defined as:

$$r = \frac{\Delta t}{\Delta x} \tag{3.34}$$

where Δx is the representative measurement of the mesh resolution. The lower the Δx, the higher the mesh resolution. Then, in the 2D $r - \omega$ space, a stability region can be drawn by fixing λ and Δx but varying Δt.

3.6.2 Accuracy and numerical viscosity

Usually, the study of accuracy is based on the measurement of the error between the numerical and analytical solutions in velocities, pressure or temperature. Two types of conclusions can be made from the solution errors. The first conclusion is very straightforward, which is the quantitative error of a solution at a given test condition. By measuring solution errors at different test conditions (usually different spacial or temporal resolutions), the second conclusion can be made, which is the order of the accuracy, or order for short, of a certain numerical method. The order is a very important property when talking about numerical methods, which fundamentally characterizes one method from another if their orders are different. To know the order of accuracy in space, the solution errors must be measured on a steady-state solution. However, without physical boundaries, such as the wall boundary, pressure boundary and velocity boundary, a useful steady-state solution cannot be established. Therefore, the discussion of order of accuracy is saved for Chap. 6 where the boundary treatment is introduced.

Another way to study accuracy is through the measurement of numerical viscosity (from the physical point of view) or numerical dissipation (from the mathematical point of view). The numerical viscosity (or numerical dissipation) is a numerical error that can be treated as an artificial viscosity or artificial dissipation added to the flow system, which will diffuse the momentum energy faster than it should be. Numerical viscosity is an unavoidable concern for the simulations of hyperbolic equations, especially the ones with strong advection such as the DBE. On the one hand, the numerical viscosity should be as little as possible in order to have a correct advection that is not diffused too much; on the other hand, it should be large enough in order to kill off oscillations and maintain a numerical stability. For the thermal flow problems in porous media, there is a strong interplay between viscosity and thermal diffusivity. Therefore, it is necessary to know how much numerical viscosity

is generated and carefully control it. In the current work, since both the flux and time marching schemes generates numerical viscosity in the FVDBM framework, it is desired to quantify the numerical viscosity of each scheme.

For most cases, the numerical viscosity is a positive value. Therefore, the actual viscosity of a flow simulation is always larger than expected. The actual viscosity ν_a is the overall viscosity of a numerical flow system, which is the sum of the theoretical viscosity ν_t and the numerical viscosity ν_n, as shown in Eq. (3.35).

$$\nu_a = \nu_t + \nu_n \tag{3.35}$$

where the theoretical kinematic viscosity is defined as follows:

$$\nu_t = \lambda c_s^2 \tag{3.36}$$

Once the lattice model is determined, the speed of sound c_s will be known. Then, the theoretical viscosity can be obtained with the given relaxation time. Eq. (3.36) is valid for the DBE solved by any numerical method including the FV method. In order to keep ν_n low and use Eq. (3.36) to characterize the flow with a good level of confidence, it is suggested by Lee [57] that both the flux scheme and time marching scheme should be kept at least second-order.

$$\nu_n = \nu_{fs} + \nu_{tms} + \nu_{other} \tag{3.37}$$

As shown in Eq. (3.37), ν_n has several sources, but mainly the ones contributed by the flux schemes (ν_{fs}) and time marching schemes (ν_{tms}). ν_{other} includes the viscosity from all of other numerical aspects of the flow simulations, e.g., the boundary treatment and collision term, etc. In order to investigate the accuracy of a flux scheme or time marching scheme, it would be perfect to know its own numerical viscosity and separate it from all of the contributors in Eq. (3.37). However, it is not possible, actually not necessary either, to do the separation. The approach applied in the current study is to study the entire ν_n by comparison. For example, there are two flux schemes A and B. It is impossible to obtain the absolute values of ν_{fs} of their own. But, by comparing the total numerical viscosity of two numerical cases that apply A and B respectively and keep everything else the same (time marching scheme and boundary treatment, etc.), the difference between the total numerical

viscosities from two cases is solely from the difference between the two flux schemes A and B. Mathematically speaking, $\nu_{n,A} - \nu_{n,B} = (\nu_{fs,A} + \nu_{tms} + \nu_{other}) - (\nu_{fs,B} + \nu_{tms} + \nu_{other}) = \nu_{fs,A} - \nu_{fs,B}$. The same approach will be applied to study the numerical viscosity generated by the time marching scheme in Chap. 5, by keeping everything else identical in the flow simulations.

The next question is how to quantify the total numerical viscosity ν_n. The approach adopted is to use Eq. (3.35) reversely. Once the theoretical numerical viscosity ν_t is calculated with Eq. (3.36), the numerical viscosity is:

$$\nu_n = \nu_a - \nu_t \tag{3.38}$$

where υ_a can be measured directly from the simulation results.

It is worth to note that the approach to study numerical viscosity (the numerical viscosity for the flux schemes in Chap. 4, and the numerical viscosity for the time marching schemes in Chap. 5) is carried out in a bottom-up manner, in which a certain numerical scheme is selected, then its numerical viscosity is quantified. This approach is chosen in the current work, since the goal of current work is to develop a simulation tool and study it. Therefore, it is necessary to characterize each numerical scheme in the tool package. However, once the tool package has been well studied and the interest of research has transferred from developing a tool to how to use the tool to solve a real-case problem, an up-bottom approach, which is opposite to the current one, should be adopted. Basically, one should know how much numerical viscosity is allowed for a certain physical problem in the first place, and then choose the proper numerical schemes (flux scheme, time marching scheme, etc.) in the tool package according to that numerical viscosity. However, without fully characterizing the tool package, an up-bottom approach is impossible.

3.6.3 Computational efficiency and wall time

The total amount of computational time or total wall time is the number of iterations times the wall time for one iteration. Therefore:

$$t_{tw} = N_{iter} t_{iter} \tag{3.39}$$

At the same time, how many iterations are needed depends on the length of the actual physical time and the time step size Δt determined by the stability criteria. Therefore:

$$N_{iter} = \frac{t_p}{\Delta t} \tag{3.40}$$

Combining Eq. (3.39) and Eq. (3.40), the total wall time is:

$$t_{tw} = t_p \frac{t_{iter}}{\Delta t} \tag{3.41}$$

t_p is fixed, depending on the nature of a specific flow system. In other words, how much time a physical system needs to transit from the initial state to any state hereafter is determined by the system itself, instead of the type of applied numerical method. Therefore, whether a simulation is computationally efficient is measured by the ratio $t_{iter}/\Delta t$. The lower the ratio, the more computationally efficient. However, as mentioned just a short moment ago, Δt is determined by the stability criteria of a specific numerical model. For any flow simulation, there is always a maximum Δt in order to maintain stability.

Since being stable is the minimum requirement for any flow simulation, Δt has to be predetermined before discussing the computational efficiency. Therefore, the fair way to compare the efficiency among different schemes is to fix Δt. As a result, measuring $t_{iter}/\Delta t$ will be no different than measuring t_{iter}. Similar to the numerical viscosity, the wall time for each iteration has many contributors, including the flux scheme and time marching scheme. Therefore:

$$t_{iter} = t_{fs} + t_{tms} + t_{other} \tag{3.42}$$

Also similar to the numerical viscosity, it is impossible to separate t_{fs} or t_{tms} from others in Eq. (3.42). Therefore, the same approach used to study the numerical viscosity is also chosen to quantify the wall times of flux schemes and time marching schemes. For example, if two numerical cases A and B have everything the same only except for the flux schemes, the difference in the total wall time, $t_{iter,A} - t_{iter,B}$, is exclusively due to the two different flux schemes. Once t_{iter} is obtained, the computational efficiency is inversely proportional to it.

3.6.4 Taylor-Green vortex flow

In order to carry out the studies mentioned above, a proper flow problem has to be chosen. The flow chosen for the studies of flux and time marching schemes in the next two chapters is called the Taylor-Green vortex that has an analytical solution in the following form [39]:

$$u = -u_0 \cos(k_1 x) \sin(k_2 y) \exp\left[-\nu_t \left(k_1^2 + k_2^2\right) t\right] \tag{3.43a}$$

$$v = u_0 \frac{k_1}{k_2} \sin(k_1 x) \cos(k_2 y) \exp\left[-\nu_t \left(k_1^2 + k_2^2\right) t\right] \tag{3.43b}$$

$$\rho = \rho_0 - \frac{u_0^2}{4c_s^2}\left[\cos(2k_1 x) + \frac{k_1^2}{k_2^2}\cos(2k_2 y)\right] \exp\left[-2\nu_t \left(k_1^2 + k_2^2\right) t\right] \tag{3.43c}$$

where k_1 and k_2 are two parameters defined as:

$$k_1 = \frac{2\pi}{D_x} \tag{3.44}$$

$$k_2 = \frac{2\pi}{D_y} \tag{3.45}$$

where D_x and D_y are the length and height of the rectangular computational domain.

The flow has a initial vorticity and decays with time. The initial condition is also defined in Eqs. (3.43) by giving $t = 0$. In addition, the flow has only periodic boundaries. Such a flow is chosen for the studies for two reasons. First, since it has only periodic boundaries, the solution on the boundary will evolve just like the internal region. Therefore, the effect of real physical boundary on stability, accuracy, and wall time can be completely removed. Second, specifically for the study of numerical viscosity, this is the only flow problem in which the numerical viscosity can be accurately calculated. There is a critical time t_c at which the magnitude of the vorticity is half of the initial state, which can be calculated exactly as follows:

$$t_c = \frac{\ln 2}{\nu_a \left(k_1^2 + k_2^2\right)} \tag{3.46}$$

where ν_a is the actual viscosity mentioned earlier. Therefore, by timing the flow from its initial state with a virtual stopwatch and stopping it at the moment when the flow has its vorticity halved, the time shown on that stopwatch is t_c. Then, the total viscosity of the flow is:

$$\nu_a = \frac{\ln 2}{t_c \left(k_1^2 + k_2^2\right)} \tag{3.47}$$

At last, the numerical viscosity can be calculated with Eqs. (3.38) and (3.36) once ν_a is obtained by Eq. (3.47).

4.0 FLUX SCHEMES

In this chapter, all presented flux schemes, which can be put into two groups: non-Godunov type and Godunov type, are developed on a universal stencil similar to the one in Fig. 22. All flux schemes are developed in a general way that they can work with any type of lattice model including the ones in Fig. 2 and arbitrary CC triangular mesh including the IRT mesh and general mesh in Fig. 10. After the flux schemes are formulated, a stability analysis is performed. Then, all of the flux schemes are compared against each other in terms of accuracy and computational efficiency.

4.1 BACKGROUND

As summarized in Tab. 2, the LBE is solved in a Lagrangian manner by coupling the discretization of the particle velocity space and configuration space. With such a coupling mechanism, the CFL number can be chosen to be exactly one globally, which means that after each streaming step, the PDFs along all lattice velocities will stop perfectly at grid points. Such a unique feature allows the LBM to achieve a second-order accuracy in space with a first-order advection scheme. On the other side, although it allows the utilization of arbitrary meshes and lattice models, solving the DBE on irregular grids, expectedly, introduces a considerable numerical viscosity or diffusion error [100, 112]. Such an error exclusively comes from the evaluation of the advection in Eulerian space. First, the DBE (the LBE as well) has a multi-advection feature (each advection for each PDF with its own advection speed $\boldsymbol{\xi}_i$). Each $\boldsymbol{\xi}_i$ has a different value. Therefore, in order to make sure the PDF that has the largest $\boldsymbol{\xi}_i$ satisfies its local CFL condition, all other PDFs with the

smaller $\boldsymbol{\xi}_i$ must have their CFL<1, which means numerical viscosity has to be added to the system to maintain stability. The second reason, which is not unique to the DBE, is that the complex topology on an unstructured mesh will inevitably introduce some error to the advection stencil, part of which eventually becomes numerical viscosity. These explain why, in a previous study, a theoretically second-order flux scheme for advection could only deliver a result that is slightly higher than first order [12]. Almost one order of accuracy is consumed by diffusion error. As a result, there is a need for higher-order flux schemes when DBE is solved on the unstructured mesh [93], which is also true for FVDBM.

The DBE is a hyperbolic equation with a strong advection term, which has no distinguish differences from the equations encountered in the CFD simulations. Therefore, many successful techniques can be borrowed from CFD tools to solve the DBE advection. Solving a hyperbolic equation, e.g. Euler equation or Navier-Stokes (NS) equation, with minimum diffusion error while maintaining stability is a constant concern in the CFD community. After developments for decades, there are many successful techniques, among which Godunov's method has been dominating many CFD codes in the subgenre of FVM. In Godunov's method, the advected scalar is considered as a wave moving at its characteristic velocity. Then, a Riemann problem appears at the interface between two adjacent cells, which is solved by exact or approximate Riemann solvers, e.g. Roe's solver [85]. Different reconstructions of the wave structure determine the order of Godunov's method. The piecewise-constant (PC) reconstruction proposed in the original work from Godunov [87], the piecewise-linear (PL) method developed by Van Leer [58–62] that gave birth to the still popular Monotone Upstream Scheme for Conservation Laws (MUSCL), and the piecewise-parabolic (PP) reconstruction introduced by Colella and Woodward [19, 20] give the first-order, second-order and third-order Godunov's method, respectively.

Despite the high success of Godunov's method in the CFD community, its importance is not recognized widely within the LBM circle. Most FVDBM [17, 35, 81–83, 86, 98, 100, 103, 109, 110, 115–117] work treats the advected scalar as a single value in a static point of view, in contrast to Godunov's method. All of the flux schemes resulting from such a static treatment of the advection are grouped into non-Godunov methods in this work. The general form of non-Godunov flux schemes are formulated. Two non-Godunov flux

schemes that have not been utilized in the FVDBM publications are also developed in this work: the second-order upwind (SOU) scheme and Quadratic Upwind Interpolation for Convective Kinematics (QUICK) scheme. One of the tasks in this chapter is to compare the numerical viscosities associated to non-Godunov schemes with that from Godunov's method, which has not been reported in publications. The only application of Godunov's method in FVDBM so far was employed by Patil and Lakshmisha in their simulations of single-phase problems [78–80], which involves a Riemann solver and a limiter that satisfies the Total Variation Diminishing (TVD) property. However, the linear advection term in the DBE and the mutual independence among all PDFs do not require any type of Riemann solver when calculating the PDF flux on the face between two neighbor cells (this will be explained in detail later). Three Godunov flux schemes (PC, PL and PP wave reconstruction) that do not require Riemann solver are systematically developed in this chapter.

Besides the comparison of numerical viscosities between the Godunov and non-Godunov flux schemes, the order of accuracy of each flux scheme should also be measured and compared. For any flux scheme, its order has two meanings: a theoretical one and an actual one. The theoretical order is based on the formulation of the scheme and directly linked to the number of stencil points utilized in the scheme. For example, a scheme that uses one stencil point is theoretically first-order, and second-order if two stencil points are applied, etc. The actual order is measured from the numerical results, which contains all types of errors. With no exceptions, the actual order of any scheme is lower than its theoretical one. The difference between the two is called the gap of order, which is roughly one based on a previous study [12]. However, the order of each flux scheme will not be quantitatively analyzed in this chapter, until the introduction of boundary treatments in Chap. 6.

4.2 THE UNIVERSAL STENCIL AND GENERAL FORM OF FLUX SCHEMES

As introduced in Sec. 3.4, the flux scheme yields $f_i(C)$, the PDF at the face center, to represent the average PDF on that face, $\bar{f}_i$. The final flux on the face is calculated by Eq.

(3.27). Then, the same procedure is repeated for other faces of the same CV. At last, all fluxes are summed by Eq. (3.23) and plugged into the governing equation, Eq. (3.20).

From now on, the efforts will be focused on the flux schemes that calculate $f_i(C)$. In order to have a high-order flux scheme, the stencil to be constructed should be sufficiently long (longer than the one in Fig. 22). In addition, since the lattice velocity $\boldsymbol{\xi}_i$ spans a 2π direction when i changes from *0* to *M-1*, the stencil has to be symmetric with respect to the face center, point C. Consequently, a stencil consisting of four points, *P'*, *Q'*, *R'*, and *S'* (two on each side of the center point) on the arbitrary CC triangular mesh, is constructed in Fig. 23. *P'*, *Q'*, *R'*, and *S'* are the feet of the centroids P, Q, R, and S onto the stencil line. The CVs P and Q are readily available since they are immediately attached to the face V_1V_2. However, there are multiple choices for R and S, because there are many centroids whose feet have longer distances to the face center than P and Q. Two simple conditions are devised to rule out others and make sure there is only one R and S, as follows:

- Condition one: the foot of the centroid has to be located within its own CV;
- Condition two: if multiple centroids satisfy condition one, choose the centroid whose foot is closest to the face center.

For example, in Fig. 23, centroid T_1 is not selected because its foot T_1' is located outside of its own CV; S is chosen over T_2 since *S'* is closer to C than T_2'.

The goal now is to calculate $f_i(C)$ with the PDFs at the stencil points *P'*, *Q'*, *R'*, and *S'*, which, however, are not the solution points. With the same assumption adopted in Eq. (3.29), it can be safely assumed that:

$$f_i(P') = f_i(P), f_i(Q') = f_i(Q), f_i(R') = f_i(R), f_i(S') = f_i(S) \tag{4.1}$$

With the help of the stencil in Fig. 23, the original 2D problem is transformed into a local 1D problem along the stencil line. By considering the direction of $\boldsymbol{\xi}_i$, the four-point stencil in Fig. 23 is further developed to an $\boldsymbol{\xi}_i$-dependent three-point stencil, as shown in Fig. 24.

In Fig. 24, point C is the same C in Fig. 23. The positivity of the axis is defined by the direction of lattice velocity $\boldsymbol{\xi}_i$, not the face normal $\boldsymbol{n}$. As a result, *D*, *U*, and *UU* denote

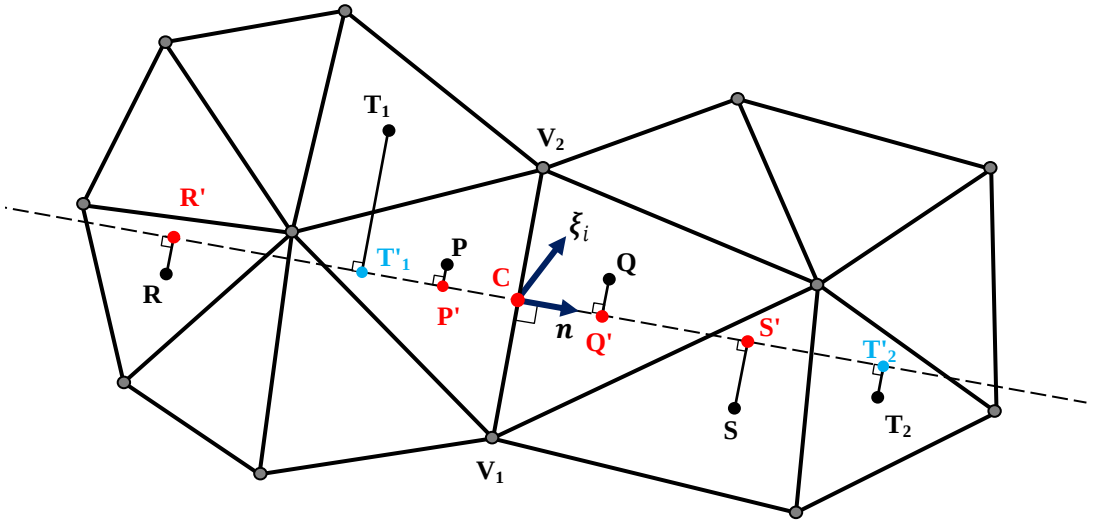


Figure 23: The universal stencil for flux schemes

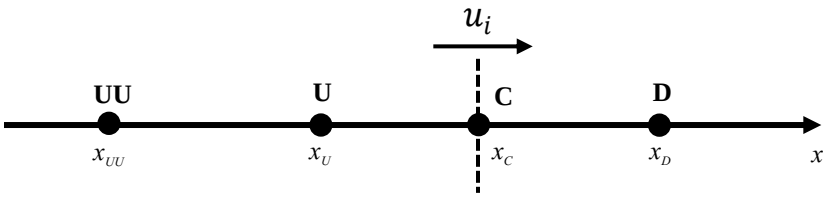


Figure 24: Lattice velocity-dependent three-point stencil across the face center

the downwind, upwind and further upwind stencil points, with respect to the wave speed u_i, which is defined as:

$$u_i = |\boldsymbol{\xi}_i \cdot \boldsymbol{n}| \tag{4.2}$$

The mapping between the three-point and four-point stencils is defined as follows, based on the direction of $\boldsymbol{\xi}_i$:

$$\begin{cases} f_i(D) = f_i(Q'), \ f_i(U) = f_i(P'), \ f_i(UU) = f_i(R') & (\boldsymbol{\xi}_i \cdot \boldsymbol{n}) > 0 \\ f_i(D) = f_i(P'), \ f_i(U) = f_i(Q'), \ f_i(UU) = f_i(S') & (\boldsymbol{\xi}_i \cdot \boldsymbol{n}) \leq 0 \end{cases} \tag{4.3}$$

Accordingly, the geometric relations are defined as:

$$\begin{cases} \left.\begin{aligned} L_D &= |x_C - x_D| = |\boldsymbol{x}_C - \boldsymbol{x}_{Q'}| \\ L_U &= |x_C - x_U| = |\boldsymbol{x}_C - \boldsymbol{x}_{P'}| \\ L_{UU} &= |x_C - x_{UU}| = |\boldsymbol{x}_C - \boldsymbol{x}_{R'}| \end{aligned}\right\} (\boldsymbol{\xi}_i \cdot \boldsymbol{n}) > 0 \\ \\ \left.\begin{aligned} L_D &= |x_C - x_D| = |\boldsymbol{x}_C - \boldsymbol{x}_{P'}| \\ L_U &= |x_C - x_U| = |\boldsymbol{x}_C - \boldsymbol{x}_{Q'}| \\ L_{UU} &= |x_C - x_{UU}| = |\boldsymbol{x}_C - \boldsymbol{x}_{S'}| \end{aligned}\right\} (\boldsymbol{\xi}_i \cdot \boldsymbol{n}) \leq 0 \end{cases} \tag{4.4}$$

where L_D, L_U, and L_{UU} are the distances from the face center C to the downwind point D, upwind point U, and further upwind point UU, respectively. Figure 24 has one less stencil point than Fig. 23 because flux schemes generally do not use the further downwind point, which will inevitably cause numerical oscillation and instability.

With the transformation from Fig. 23 to Fig. 24, the flux scheme can be put in a general formula that calculates $f_i(C)$ as a function of the subset of $f_i(D)$, $f_i(U)$ and $f_i(UU)$ and other grouped variables $\mathfrak{S}$, namely:

$$f_i(C) = \Psi\left[\Omega \in \{f_i(D), f_i(U), f_i(UU)\}, \mathfrak{S}\right] \tag{4.5}$$

As mentioned in the background of this chapter, the order of the flux scheme is determined by the number of applied stencil points. For Eq. (4.5), its theoretical order is determined by the size of the subset $\Omega \in \{f_i(D), f_i(U), f_i(UU)\}$. The theoretical order of

Eq. (4.5) is one, two and three, if the size of Ω is one, two and three, respectively. It is important to emphasize that no matter what flux scheme is applied, it has to be used on all PDFs (*i* ranges from *0* to *M-1*). For example, if the D2Q9 lattice is selected, Eq. (4.5) has to be executed nine times for each face. Therefore, the computing overhead of the overall model is largely contributed by Eq. (4.5).

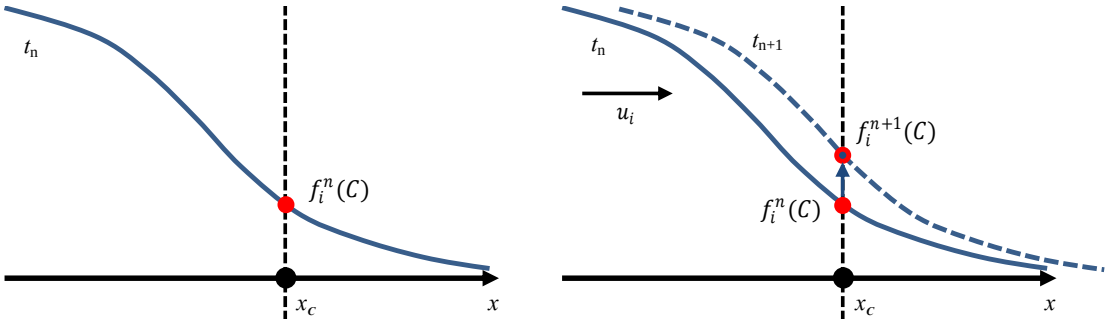
4.3 THE FLUX SCHEMES IN EXPLICIT FORMS

The next task is to substantiate Eq. (4.5) into explicit forms. Before doing that, as mentioned in the background section, the flux schemes developed in this work can be classified as Godunov type and non-Godunov type. The superscripts $\rightarrow$ and $\nrightarrow$ are used in this work to denote the Godunov flux schemes and non-Godunov flux schemes (i.e., $f_i^{\rightarrow}(C)$ is calculated with a Godunov flux scheme, $f_i^{\nrightarrow}(C)$ is calculated with a non-Godunov flux scheme).

4.3.1 Overview of Godunov and non-Godunov flux schemes

The fundamental difference between these two types of schemes is depicted in Fig. 25. For non-Godunov schemes (Fig. 25(a)), at any time step, t_n, the profile of the PDF (the solid curved line) can be assumed and constructed along the stencil. Then, the PDF at the face center can be geometrically interpolated or extrapolated using the PDF profile at t_n. For Godunov schemes (Fig. 25(b)), the same profile is constructed at t_n. However, such a profile is treated as a wave moving at speed u_i. At t_{n+1}, the wave has traveled a distance and stopped (the dashed curve line). So, between t_n and t_{n+1}, $f_i(C)$ is not a constant value but changing from $f_i^n(C)$ to $f_i^{n+1}(C)$. In essence, the non-Godunov schemes statically use the PDF profile at t_n; while the Godunov schemes take the course of movement of the PDF profile from t_n to t_{n+1} into account.

Another difference between Godunov and non-Godunov flux schemes is the continuity requirement of the PDF profile at the face center x_C. For the non-Godunov schemes in Fig. 25(a), the PDF profile has to be continuous at x_C when $x \rightarrow x_C^+$ and $x \rightarrow x_C^-$, if



(a) Non-Godunov flux schemes (b) Godunov flux schemes

Figure 25: Difference between Godunov and non-Godunov flux schemes

such schemes are based on an interpolation of the PDF profile. Otherwise, $f_i(C)$ is not obtainable. If interpreted in Fig. 23, such a continuity is equivalent to that the PDF profile is continuous all along the stencil line, even though it crosses the boundaries of the triangular CVs multiple times. However, this may not the be the real case. When the computational domain is discretized with triangular CVs, the PDFs within the CVs are continuous, but not guaranteed also continuous on the face between two CVs. In contrast, the continuity requirement at the face center is much looser for Godunov flux schemes. In Fig. 25(b), at any moment during the movement of the PDF profile from left to right, it is only required that the profile is continuous in the region where $x \leq x_C$, regardless of whether it is continuous cross the face center. It is also important to note that if a non-Godunov flux scheme calculates $f_i(C)$ with an extrapolation, instead of interpolation, the PDF profile does not need to be continuous at x_C. The extrapolation always use the left portion of the profile ($x \leq x_C$) to have a upwind nature for a better stability. Then, there is no difference between Godunov and non-Godunov flux schemes in terms of the continuity requirement at the face center.

4.3.2 Non-Godunov flux schemes

According to Eq. (4.5), different non-Godunov flux schemes with different theoretical orders can be developed by choosing a different subset of $\Omega \in \{f_i(D), f_i(U), f_i(UU)\}$.

If only one stencil point is selected, the resulting scheme will be theoretically first-order. It seems that there are three choices, $\Omega = f_i(D)$, $\Omega = f_i(U)$, or $\Omega = f_i(UU)$. However, only $\Omega = f_i(U)$ can work properly, since $\Omega = f_i(D)$ will cause unconditional instability, and $\Omega = f_i(UU)$ is too much diffusive and incorrect because the further upwind stencil point is located too far away from the face center x_C. If only the upwind stencil point is applied, the only possible way to calculate $f_i(C)$ is to constantly extrapolate $f_i(U)$ to the face center, which is the first-order upwind (FOU) scheme. The FOU is formulated in Eq. (4.6), and illustrated in Fig. 26(a).

$$f_i^{\nearrow}(C) = f_i(U) \tag{4.6}$$

If two stencil points are applied, the resulting schemes are second-order. Theoretically speaking, there are three choices as well: $\Omega \in \{f_i(D), f_i(U)\}$, $\Omega \in \{f_i(U), f_i(UU)\}$ and $\Omega \in \{f_i(D), f_i(UU)\}$. However, only the scheme that uses $\Omega \in \{f_i(U), f_i(UU)\}$ is valid. The scheme with $\Omega \in \{f_i(D), f_i(U)\}$, as discussed several times already, has the nature of the central differencing, and therefore is unconditionally unstable. The scheme with $\Omega \in \{f_i(D), f_i(UU)\}$ is too diffusive because the stencil is too long (the distance between the downwind and further upwind stencil points is the longest). The scheme with $\Omega \in \{f_i(U), f_i(UU)\}$ is called the SOU [64], since all adopted stencil points are located on the upstream side of the face center. SOU is a linear extrapolation to the face center with $f_i(U)$ and $f_i(UU)$, which is depicted in Fig. 26(b), and formulated as Eq. (4.7), where L_D, L_U and L_{UU} are defined in Eq. (4.4).

$$f_i^{\nearrow}(C) = \left(\frac{L_D}{L_D + L_U} + \frac{L_U}{L_D + L_U} \cdot \frac{L_D + L_{UU}}{L_{UU} - L_U}\right) f_i(U) - \left(\frac{L_U}{L_{UU} - L_U}\right) f_i(UU) \tag{4.7}$$

If all three stencil points are applied, the scheme is QUICK [64] that is theoretically third-order in space. QUICK scheme interpolates the PDF at the face center with all available stencil points, which is shown in Eq. (4.8), where L_D, L_U and L_{UU} are also defined in Eq. (4.4), and illustrated in Fig. 26(c).

$$\begin{aligned} f_i^{\nearrow}(C) = &\left(\tfrac{L_U}{L_D+L_U} \cdot \tfrac{L_{UU}}{L_D+L_{UU}}\right) f_i(D) + \left(\tfrac{L_D}{L_D+L_U} + \tfrac{L_U}{L_D+L_U} \cdot \tfrac{L_D}{L_{UU}-L_U}\right) f_i(U) \\ &- \left(\tfrac{L_U}{L_D+L_{UU}} \cdot \tfrac{L_D}{L_{UU}-L_U}\right) f_i(UU) \end{aligned} \tag{4.8}$$

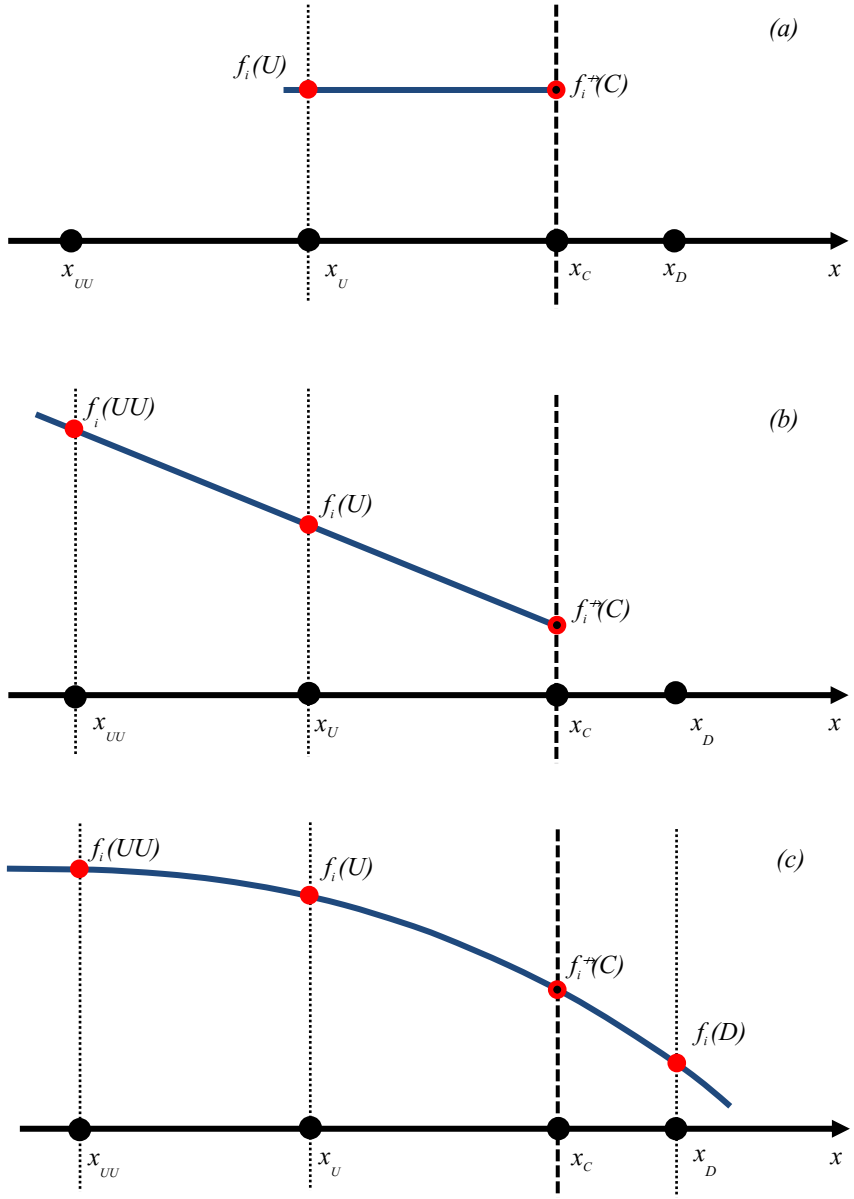


Figure 26: Non-Godunov flux schemes. (a) FOU; (b) SOU; (c) QUICK

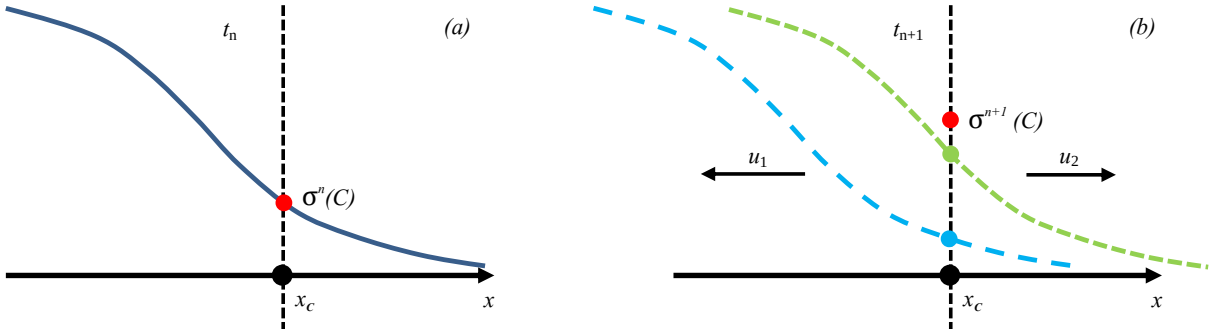


Figure 27: Wave propagation with two characteristic speeds

4.3.3 Godunov flux schemes

The typical Godunov scheme in a CFD simulation requires a Riemann solver, since the hyperbolic equation to be solved (Euler or NS) has a nonlinear advection, and the advection is coupled with acoustics. Therefore, the matrix for advection has two or more real eigenvalues, each of which corresponds to a characteristic wave speed. Such a phenomenon with two wave speeds is illustrated in Fig. 27, in which the advected scalar σ has an initial profile at t_n across the face center, as in Fig. 27(a). Such a profile will move simultaneously to the left and right with different speeds u_1 and u_2. At t_{n+1}, the value of σ at the face center is determined by overlapping two waves, as shown in Fig. 27(b). In order to solve the overlapped waves, a Riemann solver is the best choice. However, solving the gradient term in the DBE, $\boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i$, does not need a Riemann solver at all for two reasons. First, the advection is linear, since $\boldsymbol{\xi}_i$ is constant once the lattice model is determined. Second, all PDFs are mutually independent. Essentially, each PDF f_i exclusively has its own advection speed $\boldsymbol{\xi}_i$. Due to these two unique properties, when projected on the stencil, the matrix of each advected scalar, f_i, has one and only one real eigenvalue, $|\boldsymbol{\xi}_i \cdot \boldsymbol{n}|$. As a result, a Riemann solver, such as Roe's solver used in work of Patil and Lakshmisha [78–80] is a more complex technique than what is required.

Instead, a simple integral over time can suffice [63]. Assuming that the PDF at face center is a time-averaged value over the course of the wave movement from t_n to t_{n+1}, then:

$$f_i^{\rightarrow}(C) = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} f_i(x_C, t) dt \tag{4.9}$$

Since $f_i(x_C, t)$ is a simple wave with only a positive wave speed, u_i (Eq. (4.2)), and $f_i(x_C, t)$ does not change its shape during its propagation, Eq. (4.9) is equivalent to:

$$f_i^{\rightarrow}(C) = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f_i(x_C, t)dt = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f_i(x_C - u_i(t - t_n), t_n)dt \tag{4.10}$$

By introducing the following transformation between x and t:

$$x = x_C - u_i(t - t_n) \tag{4.11}$$

Then:

$$dt = -\frac{1}{u_i}dx \tag{4.12}$$

Plugging in Eqs. (4.11) and (4.12) into Eq. (4.10), then it becomes:

$$f_i^{\rightarrow}(C) = \frac{1}{u_i\Delta t}\int_{x_1}^{x_2} f_i(x, t_n)dx \tag{4.13}$$

where:

$$x_1 = x|_{t=t_{n+1}} = x_C - u_i\Delta t \tag{4.14}$$

$$x_2 = x|_{t=t_n} = x_C \tag{4.15}$$

Combining Eqs. (4.14) and (4.15), it has:

$$\Delta t = \frac{x_2 - x_1}{u_i} \tag{4.16}$$

Plugging in Eq. (4.16) into Eq. (4.13), and using the notation $f_i^n(x)$ to represent $f_i(x, t_n)$, then the Godunov flux scheme becomes:

$$f_i^{\rightarrow}(C) = \frac{1}{x_2 - x_1}\int_{x_1}^{x_2} f_i^n(x)dx \tag{4.17}$$

Equation (4.17) is the general form of Godunov flux schemes. The original temporal integral, Eq. (4.9), is transformed into a spatial integral whose physical meaning is illustrated in Fig. 28. It can be seen that the Godunov flux is the average height of the shaded area from x_1 to x_2 under the PDF profile curve at time step t_n. Since the curve is the snapshot of the moving wave at t_n for all of the possible stencil points, the superscript n in $f_i^n(x)$ in Eq. (4.17) can be dropped to maintain generality. Therefore, the final general form of Godunov flux schemes reads:

$$f_i^{\rightarrow}(C) = \frac{1}{x_2 - x_1}\int_{x_1}^{x_2} f_i(x)dx \tag{4.18}$$

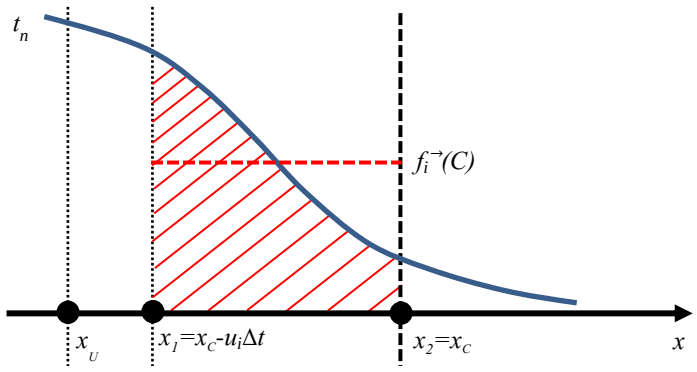


Figure 28: Figurative representation of the general form of Godunov flux schemes

The scheme developed here bears an upwind nature since the shaded region in Fig. 28 is always located to the left of the face center. The next step is to compute the integral in Eq. (4.18), which is, however, cannot be finished unless the definition of the PDF profile is known. The actual profile cannot be obtained, but can be reconstructed with piecewise-constant (PC), piecewise-linear (PL), or piecewise-parabolic (PP) assumption. Therefore, different Godunov upwind flux schemes with different orders of accuracy can be developed. Due to the upwind nature of the developed scheme, the PC, PL, and PP reconstruction only need to hold in the region to the left of the face center (i.e. $x \leq x_c$).

If the PC reconstruction is applied, as shown in Fig. 29(a), Eq. (4.18) reduces to:

$$f_i^{\rightarrow}(C) = \frac{1}{x_2 - x_1} f_i(U)\,(x_2 - x_1) = f_i(U) \tag{4.19}$$

which is exactly the FOU, Eq. (4.6), and is theoretically first-order in space.

For PL, as depicted in Fig. 29(b), a linear profile passing through $f_i(U)$ and $f_i(C)$ can be reconstructed. The general formula for a linear profile is:

$$f_i(x) = c_1 x + c_2 \tag{4.20}$$

By plugging in Eq. (4.20) back to Eq. (4.18) and finishing the integral, then:

$$f_i^{\rightarrow}(C) = \frac{1}{2} c_1\,(x_1 + x_2) + c_2 = c_1 \left(x_C - \frac{u_i \Delta t}{2} \right) + c_2 \tag{4.21}$$

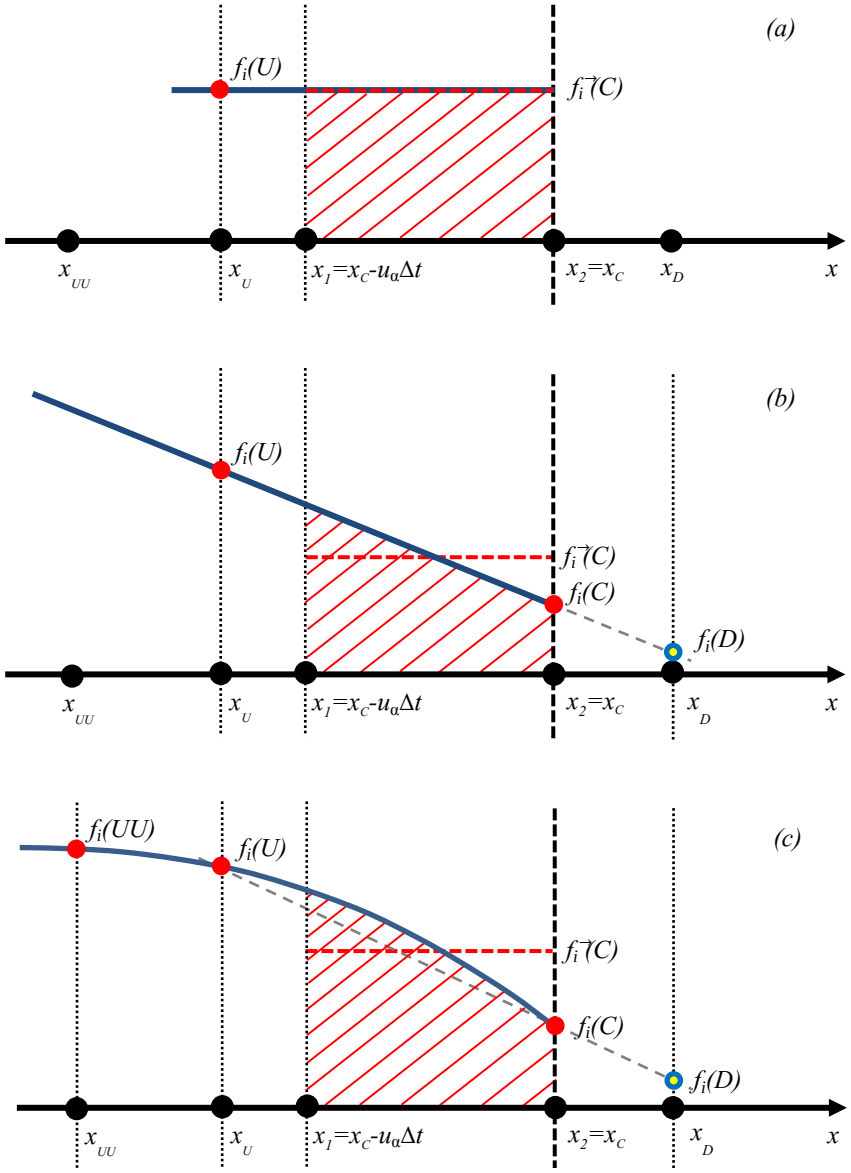


Figure 29: Godunov flux schemes with different wave reconstructions. (a) Piecewise Constant; (b) Piecewise Linear; (c) Piecewise Parabolic

$$c_1 = \frac{f_i(C) - f_i(U)}{x_C - x_U} \tag{4.22}$$

$$c_2 = f_i(U) - \frac{x_U}{x_C - x_U}\left[f_i(C) - f_i(U)\right] \tag{4.23}$$

Combining Eqs. (4.21), (4.22) and (4.23), then the final form of the Godunov flux scheme with the PL reconstruction is:

$$f_i^{\rightarrow}(C) = f_i(U) + \left(1 - \frac{u_i \Delta t}{2L_U}\right)\left[f_i(C) - f_i(U)\right] \tag{4.24}$$

where $L_U = x_C - x_U$, defined in Eq. (4.4), and $f_i(C)$ is linearly interpolated between $f_i(D)$ and $f_i(U)$, as shown in Fig. 29(b), as:

$$f_i(C) = \frac{L_U}{L_D + L_U} f_i(D) + \frac{L_D}{L_D + L_U} f_i(U) \tag{4.25}$$

The reason why the linear profile is not directly reconstructed by $f_i(D)$ and $f_i(U)$ is that, as mentioned earlier, the Godunov flux scheme allows discontinuity at x_C, and the direct reconstruction from $f_i(D)$ and $f_i(U)$ will force the PDF profile to be continuous at x_C. Therefore, the points chosen for the reconstruction have to be in the region where $x \leq x_C$. As a result, $f_i(U)$ and $f_i(C)$ are selected for the reconstruction. The former is a known value, and the latter has be to guessed, such as Eq. (4.25). However, since Eq. (4.24) is linearly reconstructed and Eq. (4.25) is also a linear interpolation, for this specific case, there is no difference between the reconstruction from $f_i(U)$ and $f_i(C)$ and the one from $f_i(D)$ and $f_i(U)$. As a result, if choosing $c_1 = \left(f_i(D) - f_i(U)\right) / \left(x_D - x_U\right)$ and $c_2 = f_i(U) - \left(f_i(D) - f_i(U)\right) x_U / \left(x_D - x_U\right)$ and using them in Eq. (4.21), Eq. (4.24) will have its alternative form as:

$$f_i^{\rightarrow}(C) = f_i(U) + \left(\frac{L_U}{L_D + L_U} - \frac{u_i \Delta t}{2\left(L_D + L_U\right)}\right)\left[f_i(D) - f_i(U)\right] \tag{4.26}$$

Eqs. (4.24) and (4.26) are exactly the same numerically. However, in order to emphasize that the PDF profile is not necessarily continuous at the face center when approaching from two directions, only Eq. (4.24) is studied in the current work.

Since $f_i(C)$ is a function of $f_i(D)$ and $f_i(U)$ (Eq. (4.25)), Eq. (4.24) is also a function of $f_i(D)$ and $f_i(U)$. Therefore, the PL Godunov flux scheme is theoretically second-order. With comparison, it can be seen that Eq. (4.24) is the modification of Eq. (4.19) with the

extra term $(1 - u_i\Delta t/(2L_U))[f_i(C) - f_i(U)]$. Therefore, Eq. (4.24) is one order higher than Eq. (4.19).

For the PP reconstruction, shown in Fig. 29(c), the general form of the PDF profile is:

$$f_i(x) = c_1x^2 + c_2x + c_3 \tag{4.27}$$

Plugging in Eq. (4.27) back to the integral Eq. (4.18), then Eq. (4.18) becomes:

$$f_i^{\rightarrow}(C) = \frac{1}{3}c_1u_i^2\Delta t^2 - \left(c_1x_C + \frac{1}{2}c_2\right)u_i\Delta t + c_1x_C^2 + c_2x_C + c_3 \tag{4.28}$$

where the coefficients c_1, c_2, and c_3 can be obtained by utilizing $f_i(C)$, $f_i(U)$, and $f_i(UU)$, with the following linear algebra:

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} x_C^2 & x_C & 1 \\ x_U^2 & x_U & 1 \\ x_{UU}^2 & x_{UU} & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} f_i(C) \\ f_i(U) \\ f_i(UU) \end{bmatrix} \tag{4.29}$$

where $f_i(C)$ is also computed with Eq. (4.25). If choosing the face center as the origin of the x axis and the positivity of the axis is pointing to the right, namely $x_C = 0$ and $x_{UU} < x_U < 0$, then Eqs. (4.28) becomes:

$$f_i^{\rightarrow}(C) = \frac{1}{3}c_1u_i^2\Delta t^2 - \frac{1}{2}c_2u_i\Delta t + c_3 \tag{4.30}$$

Also, Eq. (4.29) becomes:

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ L_U^2 & -L_U & 1 \\ L_{UU}^2 & -L_{UU} & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} f_i(C) \\ f_i(U) \\ f_i(UU) \end{bmatrix} \tag{4.31}$$

where L_U and L_{UU} are defined in Eq. (4.4). The heaviset computation for the PP Godunov flux scheme is the inverse of the 3×3 matrix. However, since such a matrix will never be singular, its inverse can be easily computed analytically, which can dramatically reduce the

computational overhead by avoiding the inverse operation of matrices. Then, Eq. (4.31) becomes:

$$\begin{aligned}\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 1 \\ L_U^2 & -L_U & 1 \\ L_{UU}^2 & -L_{UU} & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} f_i(C) \\ f_i(U) \\ f_i(UU) \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{L_U L_{UU}} & \frac{1}{L_U^2 - L_U L_{UU}} & \frac{1}{L_{UU}^2 - L_U L_{UU}} \\ \frac{1}{L_U} + \frac{1}{L_{UU}} & \frac{L_{UU}}{L_U^2 - L_U L_{UU}} & \frac{L_U}{L_{UU}^2 - L_U L_{UU}} \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_i(C) \\ f_i(U) \\ f_i(UU) \end{bmatrix}\end{aligned} \tag{4.32}$$

Equations (4.30) and (4.32) are the final form of the PP Godunov flux scheme, which is theoretically third-order since it utilizes three stencil points.

For short, The abbreviations PC, PL, and PP are used in the current work to represent the corresponding Godunov flux schemes. The FOU scheme is special, since it is both Godunov and non-Godunov. However, in order to maintain the integrity of the development of the Godunov flux scheme, FOU is considered Godunov type in the present work. Therefore, there are totally five flux schemes developed here: two non-Godunov type, SOU and QUICK, and three Godunov type, PC, PL, and PP. Theoretically, PC is first-order; SOU and PL are second-order; and QUICK and PP are third-order.

4.4 STABILITY ANALYSIS

The purpose of stability analysis is to determine the bounds (usually upper bounds) of $\Delta t/\lambda$ and $\Delta t/\Delta x$. In order to study the influences of different flux schemes on the overall FVDBM solver, every other aspects of the solver have to be kept the same, especially the time marching scheme. When changing the flux scheme, the upper bounds on $\Delta t/\Delta x$ is most likely to change, since the CFL condition, $CFL = u\Delta t/\Delta x$, is directly linked to the flux scheme that solves the advection.

In order to remove the effects of boundary conditions on the stability characteristics of the FVDBM solver, the Taylor-Green vortex flow that has only periodic boundaries is chosen to be the flow case. The D2Q9 lattice is applied. $U_0 = 0.01c_s$ in order to minimize the compressibility error. For D2Q9, $U_0 = 0.01/\sqrt{3}$. The mesh spacing Δx for the triangular mesh is not as straightforward as the Cartesian mesh. Here, it is chosen that:

$$\Delta x = \sqrt{2A_{CV}} \tag{4.33}$$

where A_{CV} is the area of the triangular CV. The IRT mesh that is shown in Fig. 10(a) with the 18×18 resolution on a 2×2 square computational domain is selected for the numerical study. Since each triangular CV is identical in the IRT mesh, Δx will be a constant value globally. Therefore, with Eq. (4.33), $\Delta x = 0.07856742$. The time marching scheme is chosen to be the IC-EC scheme (Eq. (5.13)) to allow maximum upper bound on $\Delta t/\lambda$.

With the numerical setup kept the same, the stability regions of all five flux schemes are drawn in Fig. 30, in which $\bullet$ and $\times$ represent a stable and unstable point, respectively. From the results, it can be seen that the PC scheme, shown in Fig. 30(c), has a much larger stability region than other flux scheme, in which $\Delta t/\lambda$ can reach 5, and $\Delta t/\Delta x$ can reach 0.45. However, its superiority in stability is not able to justify its application due to its extremely large numerical viscosity, which can be seen from the accuracy analysis in the next section. By comparing the Godunov and non-Godunov counterparts among the other four schemes, it can been seen that the Godunov schemes have larger stability region than the non-Godunov ones. The PL and PP schemes (Fig. 30(d) & (e)) have slightly smaller upper bounds on $\Delta t/\lambda$ than SOU and QUICK (Fig. 30(a) & (b)), but have much larger upper bounds on $\Delta t/\Delta x$, which means the Godunov flux schemes have a much better CFL condition than their non-Godunov counterparts. Therefore, for stability purpose, the Godunov flux schemes are recommended. Another observation is that the PL scheme allows larger $\Delta t/\Delta x$ than the PP scheme. In conclusion, the PL flux scheme is recommended for stability reason among all five flux schemes developed in the current work.

Nevertheless, it is necessary to note that the PL and PP schemes bizarrely have lower bounds on $\Delta t/\lambda$ and $\Delta t/\Delta x$ when Δt is decreasing. In contrast, the SOU and QUICK schemes only have the upper bounds. The explanation can be easily made by revisiting the

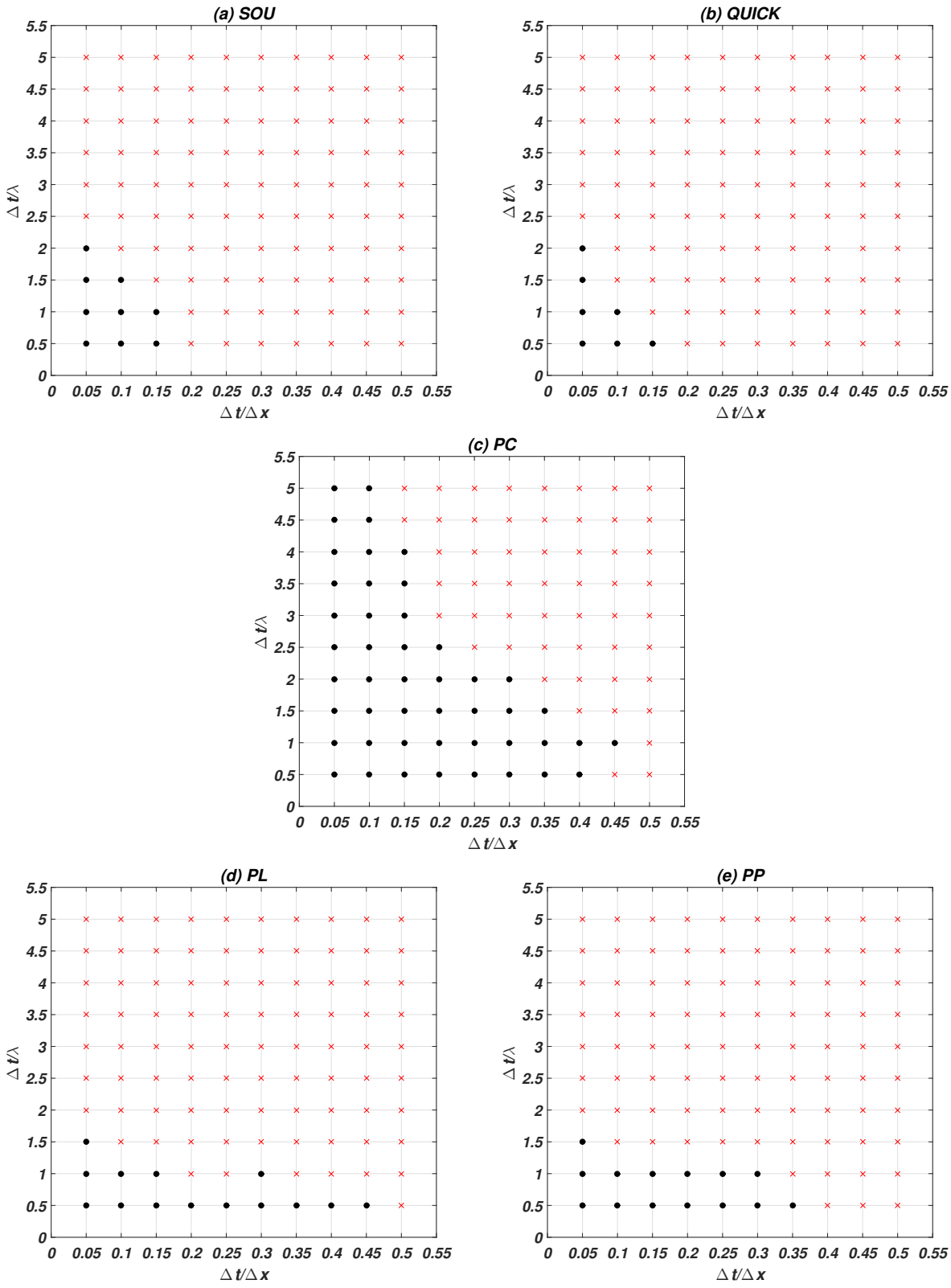


Figure 30: The stability regions of Godunov and non-Godunov flux schemes. (a) SOU; (b) QUICK; (c) PC; (d) PL; (e) PP

formulations of PL and PP schemes. For the PL scheme (Eq. (4.24)), when Δt approaches zero, the term that has Δt will be decreasing to zero too, then Eq. (4.24) is equivalent to:

$$f_i^{\rightarrow}(C) \approx f_i(U) + [f_i(C) - f_i(U)] = f_i(C) \tag{4.34}$$

Since $f_i(C)$ is obtained through central differencing, Eq. (4.25), the PL scheme will becomes unconditionally unstable. The same thing happens to the PP scheme. When Δt vanishes, the PP scheme reads:

$$f_i^{\rightarrow}(C) \approx c_3 \tag{4.35}$$

Computing c_3 with Eq. (4.32), then:

$$f_i^{\rightarrow}(C) \approx c_3 = f_i(C) \tag{4.36}$$

which is the same central differencing. However, such a feature of the PL and PP schemes does not pose any challenge for simulations, since the instability occurs only when Δt is extremely small, and one always wish to increase Δt to speed up the computation. Therefore, only the upper bound on Δt matters while its lower bound can be ignored.

4.5 ACCURACY ANALYSIS

In this section, the numerical viscosity of each flux scheme is measured. In order to do this, Taylor-Green vortex flow is selected again for the study. It is desirable to quantify the actual order of accuracy of each flux scheme as well in this section, which however cannot be completed until the boundary treatment is introduced. The reason is that the actual order of accuracy could only be measured in a meaningful steady-state solution. For a flow problem with only periodic boundaries, such as Taylor-Green vortex flow, the only steady-state solution is a died down static flow with stochastic variable variations. Therefore, in this section, the only quantitative discussion about accuracy is limited to the numerical viscosity. The actual order of accuracy of each flux scheme is only qualitatively compared, while leaving its quantitative analysis in Chap. 6.

Once the numerical viscosity ν_n is calculated with the Eqs. (3.47), (3.38) and (3.36), a variable called the relative viscosity, ν_r, can be created to measure the difference between the numerical viscosity and the theoretical viscosity. ν_r is calculated as:

$$\nu_r = \frac{\nu_n - \nu_t}{\nu_t} \tag{4.37}$$

The numerical setup is as follows. The lattice is D2Q9. $U_0 = 0.01c_s$ and $\lambda = 0.006$. The IC-EA scheme is applied again to maintain good stability, in which $\Delta t = 0.2\lambda$. Three IRT meshes with different resolutions, 18×18, 36×36 and 72×72, are prepared in order to observe the effect of mesh resolutions on ν_r of each flux scheme. The ν_r of each flux scheme with each provided mesh resolution is shown in Fig. 31, in which all of the flux schemes decrease their ν_r when the mesh is being refined. The PC scheme generates the largest numerical viscosity that is larger than the theoretical viscosity even with the fine enough 72×72 mesh. This is why the PC or FOU is never recommended for advection simulations. Besides PC, all other Godunov schemes are much less viscous than the non-Godunov counterparts, which is the direct result of the more accurate wave-like treatment of the PDF profile along the stencil than the static approach. Due to the same reason, it can be seen that the Godunov schemes are much less sensitive to the change of mesh resolution than the non-Godunov ones. Even at a very low mesh resolution, the PL and PP schemes still able to suppress the numerical viscosity. By further increasing the mesh resolution, it can be predicted that the numerical viscosity introduced by SOU and QUICK will be close to or even lower than that of PL and PP. However, due to the significant increase in computational overhead (both run time and memory), it is not able to answer the question that at what resolution the error convergence or overturn will occur. Nevertheless, what can be concluded with data support is that in moderate or low mesh resolutions, the Godunov schemes are much less diffusive than non-Godunov counterparts.

In addition, QUICK scheme is less diffusive than SOU due to its higher order of accuracy. However, the opposite phenomenon occurs between PL and PP. The supposedly higher-order PP scheme introduces more numerical viscosity than the PL scheme, which, however, does not suggest that the order of accuracy, both theoretical and actual, of PP is lower than PL. By comparing the numerical viscosity difference between PP and PL (Fig. 32), it can be

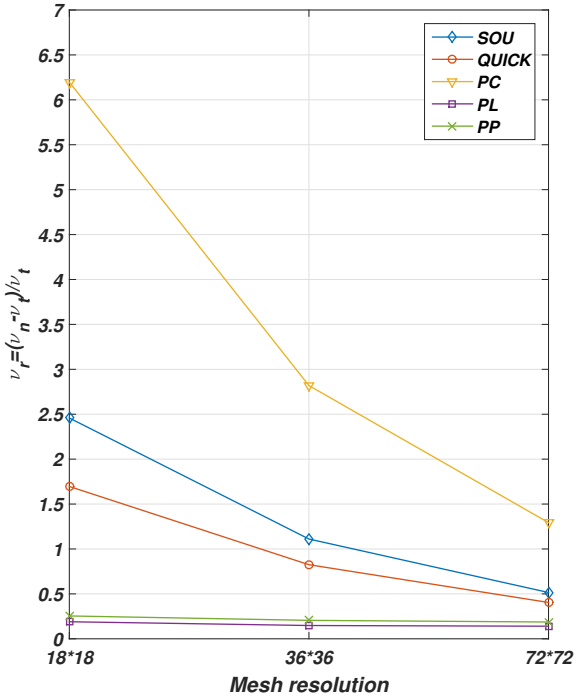


Figure 31: The relative numerical viscosity ν_r of different flux schemes with different mesh resolutions

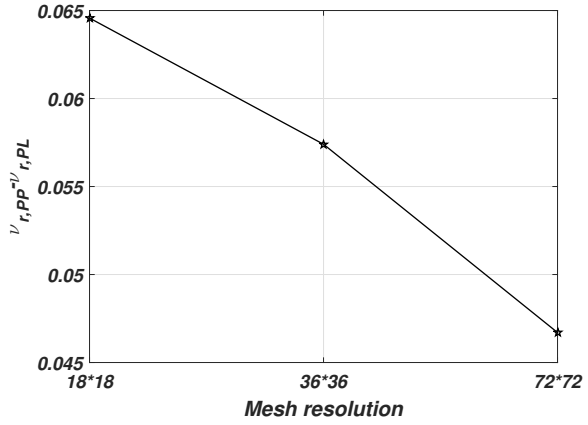


Figure 32: The difference of ν_r between the PP and PL flux schemes with different mesh resolutions

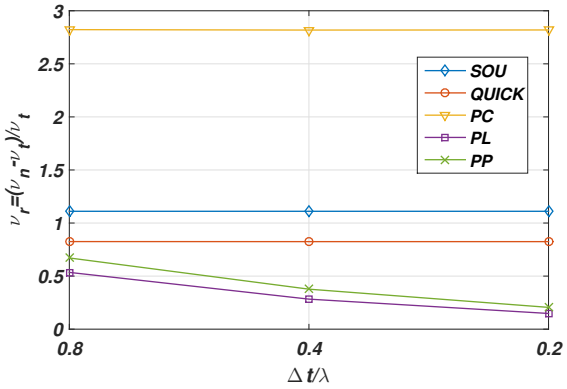


Figure 33: The relative numerical viscosity ν_r of different flux schemes with different Δt

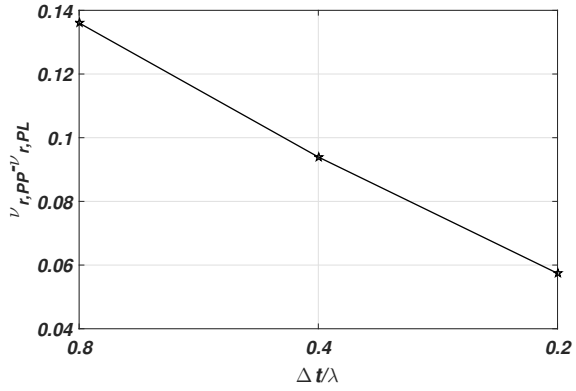


Figure 34: The difference of ν_r between the PP and PL flux schemes with different Δt

observed that, with the increasing of the mesh resolution, the numerical viscosity generated by the PP scheme is decreasing faster than the PL scheme, and more importantly, such a decreasing is accelerating, which indicates that the only reason why PP is more diffusive than PL in the current test is because the mesh is not fine enough. Therefore, there must exist a pivotal mesh resolution, below which the PL scheme is less diffusive and above which the PP scheme is less diffusive. The reason why PP is not monotonically less diffusive than PL, like that between SOU and QUICK, is also due to the difference of the wave-like and static treatments of the PDF profile between the Godunov and non-Godunov methods. As mentioned early, a three-point stencil is more accurate than a two-point stencil because more information from the extra stencil point is utilized. Another consequence caused by the extra stencil point is the dramatic change of the shape of the constructed PDF profile, which can be easily seen in Fig. 26 and Fig. 29. For non-Godunov flux schemes, the PDFs at the face center are statically interpolated or extrapolated from the stencil points. The change of the shape of the profile will only slightly affect the final PDF value at the face center. In contrast, for the Godunov flux schemes, the PDFs at the face center is based on the spatial integral under the profile curve. Therefore, the shape of the PDF profile is critical to the Godunov flux schemes. A slight change of the profile shape will result in a nontrivial change of the integral under the curve and consequently the PDF value at the face center. As a result, when the mesh is coarse, the geometric length of the stencil for PP scheme is very long, the shape of the PDF profile constructed with such a stencil is less accurate than that from a shorter stencil of the PL scheme. Then, the numerical viscosity of the PP scheme with coarse mesh is larger than the PL scheme. When the mesh is refined, the stencil will become geometrically shorter. Therefore, the improvement of the profile shape with the three-point stencil of the PP scheme will be more than that with the two-point stencil of the PL scheme, followed by less numerical viscosity.

For the PL and PP schemes, not only deceasing the spatial step size Δx but also reducing the temporal step size Δt will cause the numerical viscosity to diminish, since Δt is an input variable in their formulations (see Eqs. (4.24) and (4.30)). On the contrary, decreasing Δt will not make less or more numerical viscosity in SOU, QUICK, and PC because their formulas are not functions of Δt. Another series of tests is performed on a 72×72 IRT mesh

with three sizes of Δt: 0.2λ, 0.4λ, and 0.8λ. The relative numerical viscosity ν_r of each flux scheme when decreasing Δt is shown in Fig. 33. It can be seen that the numerical viscosity of SOU, QUICK and PC keep constant when Δt is decreasing. In contrast, PL and PP suffer gradually less numerical viscosity when reducing Δt. By calculating the ν_r difference between PP and PL at different Δt, it can be seen that the numerical viscosity of the PP scheme decrease faster than PL. This is because PP is a function of $\mathcal{O}(\Delta t^2)$ (see Eq. (4.30)) while PL is only $\mathcal{O}(\Delta t)$. Therefore, PP is one order higher than PL in both space and time.

With the numerical simulations, two major conclusions can be made. First, the non-Godunov schemes, SOU and QUICK, have more diffusion errors than the Godunov counterparts with the same orders, PL and PP. Second, PP has higher order than PL both in space and time, but more diffusive than PL when the spacial resolution is moderate or low.

4.6 ANALYSIS OF COMPUTATIONAL EFFICIENCY

The computational efficiency is inversely proportional to the wall time. By briefly comparing the complexity of the formulation of each flux scheme, it can be basically seen that the flux scheme with more computational complexity will run slower. Therefore, the PP scheme is the least efficient one since it contains more operations than other schemes, such as the matrix operation (Eq. (4.32)). However, in order to quantitatively compare the computational efficiency among all flux schemes, there is no better way than timing them in an actual numerical case. Therefore, the wall time per iteration t_{iter} of each flux scheme is measured and compared against each other. The flow problem chosen for the test is still Taylor-Green vortex flow. The test in performed on a 36×36 IRT mesh with the D2Q9 lattice, and the time marching scheme is the IC-EA (Eq. (5.13)). All of other parameters do not effect the wall time measurement, such as Δt and λ. In order to draw a general conclusion that is independent of machine hardware and software, the wall time of each flux scheme is normalized by the wall time of the PC scheme, $t_{iter,PC}$.

The test results are shown in Fig. 35. First, it can be seen that the PC scheme is the fastest. Second, QUICK is less efficient than SOU, and PP is less efficient than PL, due

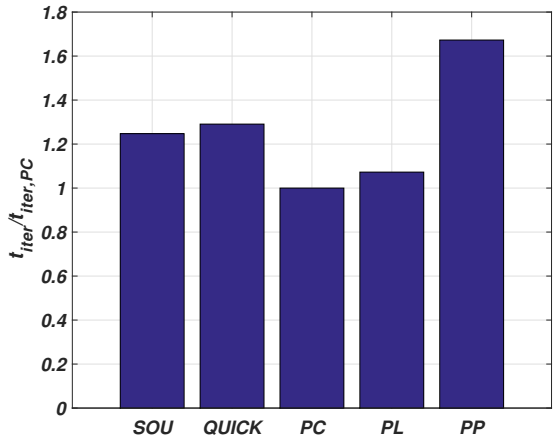


Figure 35: The normalized wall times of flux schemes per iteration

to the fact that the latter schemes contain more computational operations. Third, the PP scheme is the slowest since it contains a multiplication of a 3×3 matrix with a vector (Eq. (4.32)) that no other schemes have. The last but not the least, the PL scheme is only slightly slower than PC, and is much faster than all other schemes. Therefore, by combining the stability and accuracy analysis in the previous section, in which it can seen that PL and PP are more stable than SOU and QUICK, and PL is the least diffusive scheme with moderate mesh resolutions, a solid recommendation can be made as follows. For most numerical cases with moderate or low mesh resolutions, PL is the top one choice. If run time is not a concern and a very large mesh is required for better accuracy, PP should be selected.

5.0 TIME MARCHING SCHEMES

As mentioned in Sec. 3.6.2, in order to use Eq. (3.36) as the actual viscosity of a flow simulation, the time marching scheme has to be at least second-order. In this chapter, several second-order or higher-order time marching schemes are presented, as well as their comparisons of stability, numerical viscosity and computational efficiency. Other time marching schemes are not presented here, since they are not critical to the current work.

5.1 THE FORMULATIONS

The most standard second-order time marching scheme is the Second-Order Runge-Kutta (RK2) scheme as follows:

$$f_i^{n+1}(P) = f_i^n(P) + \Delta t\left(\frac{1}{2}K_1 + \frac{1}{2}K_2\right) \tag{5.1}$$

where:

$$K_1 = T_i^n(P) = C_i^n(P) - F_i^n(\partial P) \tag{5.2}$$

$$K_2 = T_i^*(P) = C_i^*(P) - F_i^*(\partial P) = \left[C_i(P) - F_i(\partial P)\right]\Big|_{f_i^* = f_i^n + \Delta t K_1} \tag{5.3}$$

In Eq. (5.3), the superscript $*$ indicates the corresponding value at the intermediate time step. The letter P in parenthesis is omitted in $f_i^* = f_i^n + \Delta t K_1$ in order to show that the update of the PDFs occurs not only at the current location P, but also throughout the entire computational domain. In addition, the macroscopic variables as well as the equilibrium PDFs should also be updated globally.

If a higher order is required, the Fourth-Order Runge-Kutta (RK4) scheme is another standard choice. Similar to RK2, the RK4 utilizes not one, but three intermediate time steps, which reads:

$$f_i^{n+1}(P) = f_i^n(P) + \Delta t\left(\frac{1}{6}K_1 + \frac{1}{3}K_2 + \frac{1}{3}K_3 + \frac{1}{6}K_4\right) \tag{5.4}$$

where:

$$K_1 = T_i^n(P) = C_i^n(P) - F_i^n(\partial P) \tag{5.5}$$

$$K_2 = T_i^*(P) = C_i^*(P) - F_i^*(\partial P) = \left[C_i(P) - F_i(\partial P)\right]\Big|_{f_i^* = f_i^n + \frac{\Delta t}{2}K_1} \tag{5.6}$$

$$K_3 = T_i^{**}(P) = C_i^{**}(P) - F_i^{**}(\partial P) = \left[C_i(P) - F_i(\partial P)\right]\Big|_{f_i^{**} = f_i^n + \frac{\Delta t}{2}K_2} \tag{5.7}$$

$$K_4 = T_i^{***}(P) = C_i^{***}(P) - F_i^{***}(\partial P) = \left[C_i(P) - F_i(\partial P)\right]\Big|_{f_i^{***} = f_i^n + \Delta t K_3} \tag{5.8}$$

However, all of the Runge-Kutta-based schemes, including RK2 and RK4, are quite computationally expensive due to the explicit evaluation of the collision term. Therefore, in order to maintain a good stability, Δt has to be small enough compared to the relaxation time λ. Theoretically, it should satisfy the condition $\Delta t < 2\lambda$ (Eq. (3.32)); however, such a condition becomes stricter when the unstructured mesh is adopted. Ubertini and Succi [102] showed that $\Delta t < \lambda$ should be held. Such a seemingly minor constraint is actually disastrous for the simulation of high-Re flows with the DBE solvers. For the flows simulated with the DBE, the relation between the viscosity and relaxation time is $\nu = \lambda c_s^2$ (Eq. (3.36)). Therefore, in order to simulate high-Re flows, λ has to diminish along with a deceasing Re, so Δt has to shrink as well due to $\Delta t < \lambda$. If Re is very high, Δt has to be very small, almost zero, and then the run time of the simulation will approach to infinity.

However, the conventional LBM does not suffer from such a constraint. As already discussed by Ubertini *et al.* [103], the relation between the viscosity and relaxation time for the LBM is:

$$\nu = c_s^2\left(\lambda - \frac{1}{2}\Delta t\right) \tag{5.9}$$

Such an expression implies that a very low viscosity can be achieved, while still keeping both λ and Δt relatively large. Mathematically speaking, $\lambda \sim \mathcal{O}(1)$ and $\Delta t \sim \mathcal{O}(1)$. This is

a direct consequence of a negative numerical diffusivity $-c_s^2\Delta t/2$. Therefore, just because of the difference in the $\nu - \lambda$ relation, without including all of the other computational complexities introduced by the DBE solvers (such as the flux scheme discussed previously) a general DBE solver is three to five times slower than a LBM solver. This is one of the reasons why the conventional LBM is still preferred in the community, and also why one of the state-of-art research topics about the DBM is to loosen the constraint on Δt to speed up the solver. Recent studies showed that by using an implicit approximation of the collision term, the upper bound on Δt can be increased.

A DBE formulation that includes a general implicit collision term was proposed by Bardow *et al.* [5], and later tested by Lee and Lin [57] in their FEDBM study. Such a formulation is modified for the current FVDBM approach in Eq. (5.10):

$$f_i^{n+1}(P) = f_i^n(P) + \Delta t\left[(1-\theta)\,C_i^n(P) + \theta C_i^{n+1}(P)\right] - \Delta t F_i^n(\partial P) \tag{5.10}$$

where $0 \leq \theta \leq 1$. There are usually three choices for θ: 0, 1/2, and 1. If $\theta = 0$, Eq. (5.10) reduces to the Forward Euler (F-E) scheme (Eq. (3.31)). If $\theta = 1/2$, the time marching scheme is the Crank-Nicholson (C-N) scheme. If $\theta = 1$, the resulting scheme is the Backward Euler (B-E) scheme. In the current work, only the B-E scheme will be developed and discussed. Using in $\theta = 1$ in Eq. (5.10):

$$f_i^{n+1}(P) = f_i^n(P) + \Delta t C_i^{n+1}(P) - \Delta t F_i^n(\partial P) \tag{5.11}$$

Replacing $C_i^{n+1}(P)$ with its explicit form $\frac{1}{\lambda}\left[f_i^{eq,n+1}(P) - f_i^{n+1}(P)\right]$, then Eq. (5.11) becomes:

$$f_i^{n+1}(P) = f_i^n(P) + \frac{\Delta t}{\lambda}\left[f_i^{eq,n+1}(P) - f_i^{n+1}(P)\right] - \Delta t F_i^n(\partial P) \tag{5.12}$$

Merging the terms that have $f_i^{n+1}(P)$ and rearranging the equation, then Eq. (5.12) becomes:

$$f_i^{n+1}(P) = \frac{f_i^n(P) + \frac{\Delta t}{\lambda} f_i^{eq,n+1}(P) - \Delta t F_i^n(\partial P)}{1 + \frac{\Delta t}{\lambda}} \tag{5.13}$$

Equation (5.13) evaluates the collision term implicitly and the advection term explicitly. Therefore, in this work it is called the Implicit-Collision-Explicit-Advection (IC-EA) scheme, which is second-order in time. The only obstacle to complete the calculation in Eq. (5.13) is

the evaluation of $f_i^{eq,n+1}(P)$, which, however, can be linearly extrapolated with the previous two time steps, as follows:

$$f_i^{eq,n+1}(P) = 2f_i^{eq,n}(P) - f_i^{eq,n-1}(P) \tag{5.14}$$

5.2 STABILITY ANALYSIS

Similar to the flux schemes, the stability analysis is also carried out for the time marching schemes. The same Taylor-Green vortex flow is chosen as the study problem, in order to remove the effects of boundary conditions. The numerical setup is also the same, such that the D2Q9 lattice is applied, $U_0 = 0.01c_s$, and the mesh is 18×18 IRT. With the flux scheme kept the same (SOU is chosen to maintain a second-order accuracy in space), the stability regions of all four time marching schemes are drawn in Fig. 36, in which $\bullet$ and $\times$ represent a stable and unstable point, respectively. From the results, it can be seen that the IC-EA has the largest stability region, which is slightly larger than the one of the RK4 scheme, and much larger than the identical stability regions of F-E and RK2. Therefore, for stability purposes only, the IC-EA scheme in Eq. (5.13) is the best choice.

5.3 ACCURACY ANALYSIS

As was completed for the study of the flux schemes, the numerical viscosity ν_n calculated with Eqs. (3.47), (3.38) and (3.36) for Taylor-Green vortex flow is also measured for different time marching schemes. Likewise, the same relative viscosity ν_r calculated with Eq. (4.37) is compared among different schemes.

A series of numerical tests are performed to measure ν_r for each time marching scheme formulated in Sec. 5.1. The forward Euler (F-E) scheme is also studied to serve as a base case. In order to develop a better comparison, Δt has three sizes: 0.1λ, 0.2λ, and 0.4λ. All of the other conditions are kept the same, including the D2Q9 lattice, $U_0 = 0.01c_s$, and $\lambda = 0.006$. All numerical simulations are carried out on the same 72×72 IRT mesh. In

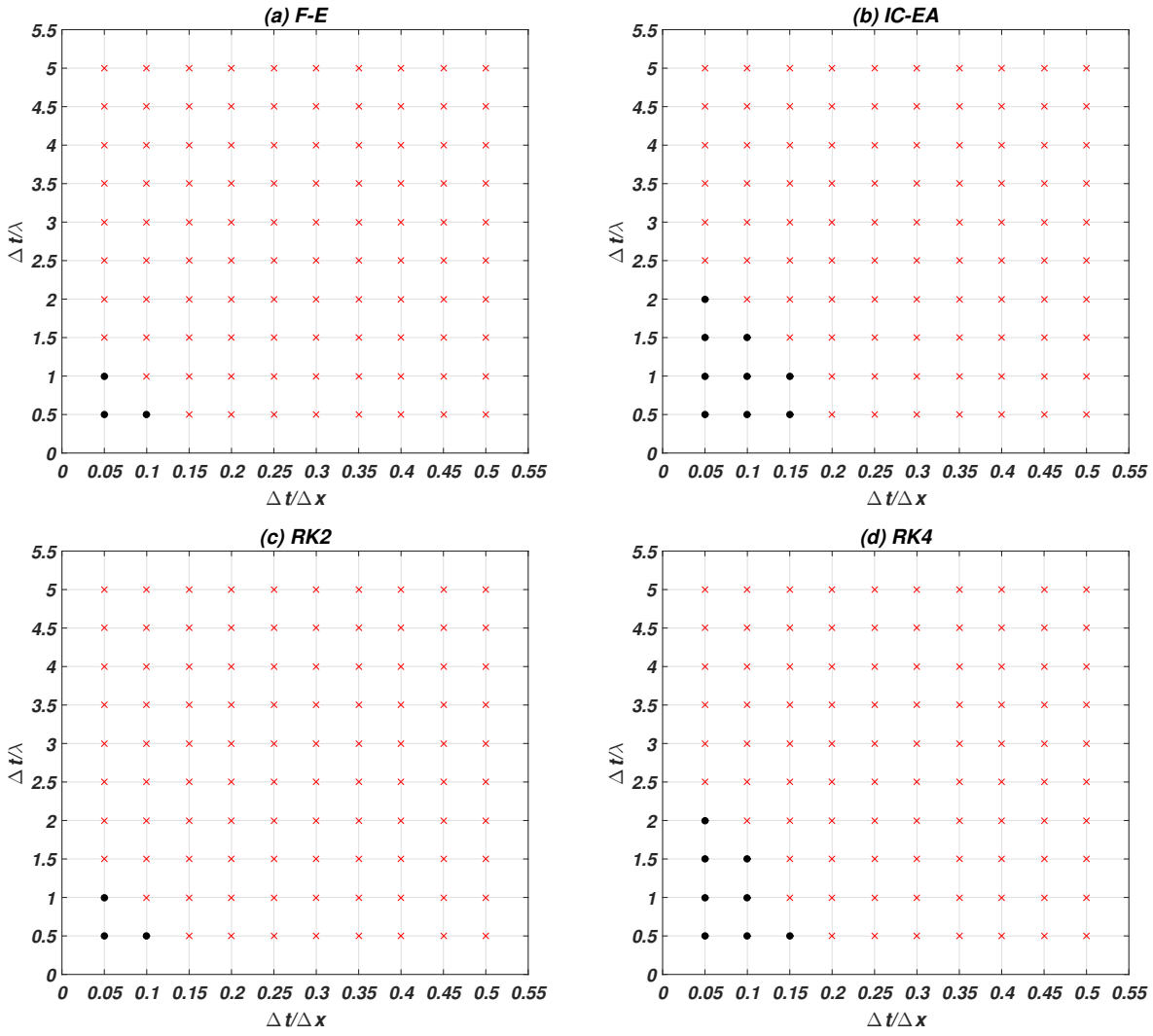


Figure 36: The stability regions of different time marching schemes. (a) F-E; (b) IC-EA; (c) RK2; (d) RK4

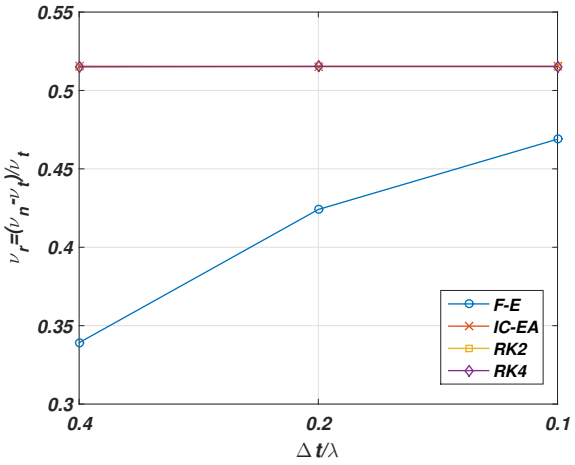


Figure 37: The relative numerical viscosity ν_r of different time marching schemes with different Δt

order to remove the effect of Δt on the numerical viscosity of the flux scheme, the SOU scheme that is independent of Δt is chosen for the testing. The ν_r of each time marching scheme with each provided time step size is shown in Fig. 37. Several conclusions can be made. First, for the F-E scheme, it can be seen that decreasing Δt will increase the numerical viscosity, which is a result of the combination of the transient nature of Taylor-Green vortex flow and the significant diffusivity of the F-E scheme. The numerical viscosity is measured after a certain length of physical time (see Eq. (3.47)). When Δt is reduced, the numerical viscosity introduced by F-E during each time step is reduced too. However, it requires more time steps to reach the same physical time. Since F-E is a lower-order method, the accumulated numerical viscosity by the increased number of time steps will eventually increase. Second, by comparison, it can be seen that the other schemes, IC-EA, RK2, and RK4, demonstrate a higher order than F-E, because their slopes are zero while the slope of F-E is positive. Third, interestingly, there is no difference in terms of accuracy among IC-EA, RK2, and RK4. The supposedly higher-order scheme, RK4, does not generate a more accurate result than RK2 and IC-EA. Therefore, RK4 is not recommended due to its unnecessary higher level of complexity.

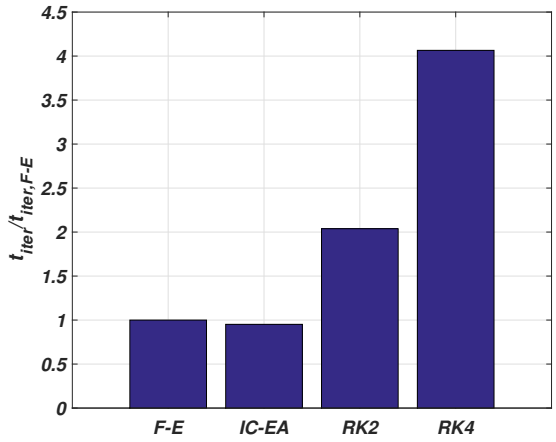


Figure 38: The normalized wall times of time marching schemes per iteration

5.4 ANALYSIS OF COMPUTATIONAL EFFICIENCY

As was done for the flux schemes, the wall time per iteration t_{iter} is utilized to compare the computational efficiency of each time marching scheme. All test conditions are kept the same as in the accuracy study, except that the mesh is a 36×36 IRT mesh. Likewise, the wall time of each time marching scheme is normalized by the wall time of the F-E scheme, $t_{iter,F-E}$.

The test results are shown in Fig. 38. It can be easily seen that IC-EA is the most efficient scheme and RK4 is the slowest. In addition, RK4 is as twice slow as RK2. Combing this with the accuracy study conclusions, it is obvious that one should choose the IC-EA scheme over other time marching schemes due to its best overall performance.

6.0 BOUNDARY TREATMENT

The boundary treatment (or boundary scheme) in the current work specifically means the methods and algorithms that numerically realize the physical boundary conditions (e.g. the specified values of macroscopic variables on the boundaries). The boundary treatment is equally as important as the FVDBM solver, since most physical problems, unlike the Taylor-Green vortex flow used for the study of the flux schemes and time marching schemes in the previous chapters, require real physical boundary conditions. In this chapter, a unified three-step boundary treatment to realize different physical boundary treatment will be developed for the FVDBM.

6.1 BACKGROUND

As discussed in Sec. 1.2, a major motivation in the current work to develop the FVDBM on an unstructured mesh is to incorporate curved boundaries easily and accurately. Therefore, the minimum requirement for the boundary treatment is the ability to handle unstructured mesh on curved boundaries, as well as flat ones. However, there are more challenges to overcome.

First, as mentioned in Chap. 1, the decoupling between the configuration space and velocity space allows the FVDBM solver to incorporate not only different types of meshes, but also different types of lattice models. It is desirable that such a flexibility is not intrinsically limited by the boundary scheme. However, none of the existing studies address this topic.

Another challenge for any boundary treatment of fluid simulations is that it should hold a non-slip condition on a wall boundary (moving or stationary) when the simulation length

scale is relatively large (small Knudsen number). However, the non-slip condition is only a specific case of a more general situation. Generally speaking, if a macroscopic variable, velocity or density, is defined on a boundary, the boundary treatment should preserve the macroscopic variable on that boundary up to a certain level of accuracy. When a macroscopic variable is prescribed on a boundary, such a physical boundary condition is called a Dirichlet boundary condition. Therefore, a desired boundary treatment should be able to preserve Dirichlet boundary conditions up to a given level of accuracy.

In addition, a desired boundary treatment should be able to work with all of the developed flux schemes in Chap. 4. Also, the boundary scheme should be at least second-order accurate, in order to achieve an overall second-order accuracy in space for the entire FVDBM model, when the chosen flux scheme is second order or higher.

However, after examining the existing boundary condition techniques in the literature (both the ones developed for the LBM and the ones for FVDBM), none of them can meet the requirements for the present study.

When the configuration space is decomposed with an unstructured mesh, some grid nodes will be exactly located on boundaries (these nodes are called boundary nodes), and then the solution scenario is very similar to what is used on the flat boundaries for the conventional LBM. However, the boundary schemes developed for the flat boundaries of the LBM are not appropriate for the FVDBM, which can basically be put into two categories. The first type was developed for open boundaries, such as an inlet and outlet, where the flow velocity or density is prescribed [50, 77, 119]. This type of scheme can be well represented by Zou-He's technique [119], which basically solves a closed linear system. The unknowns are the PDFs flowing into the fluid domain, and the constraints are usually flow velocity or density on the boundaries. Zou-He's scheme is designed for the D2Q9 lattice and could only handle three unknowns, which will break down when the boundary is encountered with an overconstrained or underconstrained scenario. The overconstrained case happens when both velocity and density are defined on the boundary or a lower-order lattice such as D2Q7 is applied. In both cases, the number of constraints is more than the number of unknown PDFs. On the other hand, when a higher-order lattice model is applied, such as D2Q13, the boundary condition will be underconstrained because the number of unknown PDFs flowing

inward is more than the number of known constraints. Therefore, such a boundary treatment will limit what type of lattice model should be selected. The second category of boundary schemes in the LBM framework were developed for a wall boundary. The most commonly used technique is the bounce-back (BB) scheme; however, BB is only first-order, and will not maintain a non-slip condition on the wall when the relaxation time is large [50, 52, 77, 118]. The halfway BB scheme developed later solved this issue. However, it is still not applicable to FVDBM, since in the unstructured Eulerian solver there is no rigorous definition of where the halfway wall should be located. Other flat boundary schemes, such as the second-order scheme for an inclined wall [37], the interpolation [18] and extrapolation [14] scheme for a flat wall, and the nonreflecting scheme for open boundaries [53], cannot be adopted by FVDBM simply because they are all developed for the LBM approach on a uniform Lagrangian grid network for streaming that does not exist in the Eulerain FVDBM on an unstructured mesh. In summary, the boundary schemes developed for the LBM do not fit the FVDBM methodology, and cannot meet the aforementioned requirements.

Other possible options are the boundary schemes developed for other FVDBM models. Most FVDBM approaches in the literature are based on a VC mesh [16, 81–83, 86, 98, 100, 103, 109, 110]. As scrutinized in Sec. 3.1, the VC mesh can cause ambiguity and numerical conflict on the boundaries, which is one of the reasons why the CC mesh is selected in the present work. Their boundary schemes, called the co-volume method, do not work for the CC mesh due to the difference in the mesh structure. There are a few FVDBM papers [78–80, 115] that use a CC mesh but still adopt the Zou-He scheme or something similar. Guo *et al.* [40, 41] developed a boundary scheme that is completely different from the Zou-He scheme and applied it in their hydro-thermal LBM simulation [42]. In that paper, the total PDF on the boundary is decomposed into two parts: equilibrium and non-equilibrium. The equilibrium part can be computed from the physical boundary conditions and the non-equilibrium part is extrapolated from an interior node. This method provides a possibility that all PDFs, no matter how many there are, can be evaluated on the boundary without considering matching the number of unknown PDFs and the number of constraints. However, such a scheme was developed for the LBM, and must be modified for the CC FVDBM. Ghasemi and Razavi [35] used Guo's scheme for their cell-centered FVDBM model on a quadrilateral

mesh. Unfortunately, they only partially applied this scheme to their thermal model, while still using the Zou-He scheme for their hydrodynamic boundary condition.

In summary, the boundary treatment developed in this chapter will use Guo's scheme as a starting point. Moreover, it will be specifically developed for the FVDBM to satisfy the following requirements:

- The boundary treatment should handle the general CC mesh on a flat or curved boundary in a unified way. Namely, the local geometric topology at each boundary node should be ignored by the boundary scheme;
- The boundary treatment should incorporate different lattice models in a unified way;
- The boundary treatment should be able to preserve a Dirichlet boundary condition of any macroscopic variable;
- The boundary scheme should seamlessly work with different flux schemes; and
- The order of accuracy of the boundary treatment should be at least second-order.

6.2 THE BOUNDARY TREATMENT BREAKDOWN

Only the flux calculation requires boundary conditions, since the collision term is localized at the centroids of the CVs and there is no centroid located on the boundaries. In order to guarantee a consistent order of accuracy throughout the entire computational domain and also a smooth transition of the solution between the boundaries and interior region, it is necessary to use the same flux scheme for the internal faces and the faces on the boundary. The flux schemes developed in the present work require a universal stencil on each face (the face is called the owner face of its stencil). However, when a face is located on the boundary, the stencil points on one side of the face are missing, due to the absence of the triangular mesh out of the boundary. Therefore, the boundary treatment should construct the missing stencil points in the first place. Then, it should evaluate the PDF at the constructed stencil points with the known physical boundary conditions. After that, the fluxes through all the faces, including the faces on the boundaries, are computed identically with the schemes

described in Chap. 4 regardless of whether a face (or which face) is on the boundaries, which can save some computational cost.

6.2.1 Construction of ghost stencil points and their PDF evaluation

One side of the stencil points will be missing when the owner face of the stencil is on the boundaries (Fig. 39(a)). Due to such a long stencil (two stencil points on each side of the face), even when the owner face is located in the interior but close enough to the boundaries, part of the stencil points may still be missing (Fig. 39(b)).

In Fig. 39(a), on face N_1N_2, its ghost stencil points Q' and S' are constructed such that they are equally spaced from the boundary:

$$|S'Q'| = |Q'C| = |P'C| \tag{6.1}$$

Therefore, the PDFs at Q' and S' can be geometrically extrapolated as:

$$f_i\left(Q'\right) = 2f_i\left(C\right) - f_i\left(P'\right) \tag{6.2}$$

$$f_i\left(S'\right) = 2f_i\left(Q'\right) - f_i\left(C\right) \tag{6.3}$$

where $f_i\left(C\right)$ can be interpolated with the PDFs at boundary nodes N_1 and N_2. Since point C is the face center, it holds that:

$$f_i\left(C\right) = \frac{1}{2}\left[f_i\left(N_1\right) + f_i\left(N_2\right)\right] \tag{6.4}$$

The problem can be closed if $f_i(N_1)$ and $f_i(N_2)$ are known, which will be introduced later in Sec. 6.2.2.

In Fig. 39(b), on the interior face N_2N_3, the stencil is extended to the exterior region and is intercepted by the boundary at point I. Only one stencil point, S', is missing. Similarly, S' is constructed in such a way that:

$$|S'I| = |Q'I| \tag{6.5}$$

With a linear extrapolation, the PDF at the ghost stencil point can be calculated as:

$$f_i\left(S'\right) = 2f_i\left(I\right) - f_i\left(Q'\right) \tag{6.6}$$

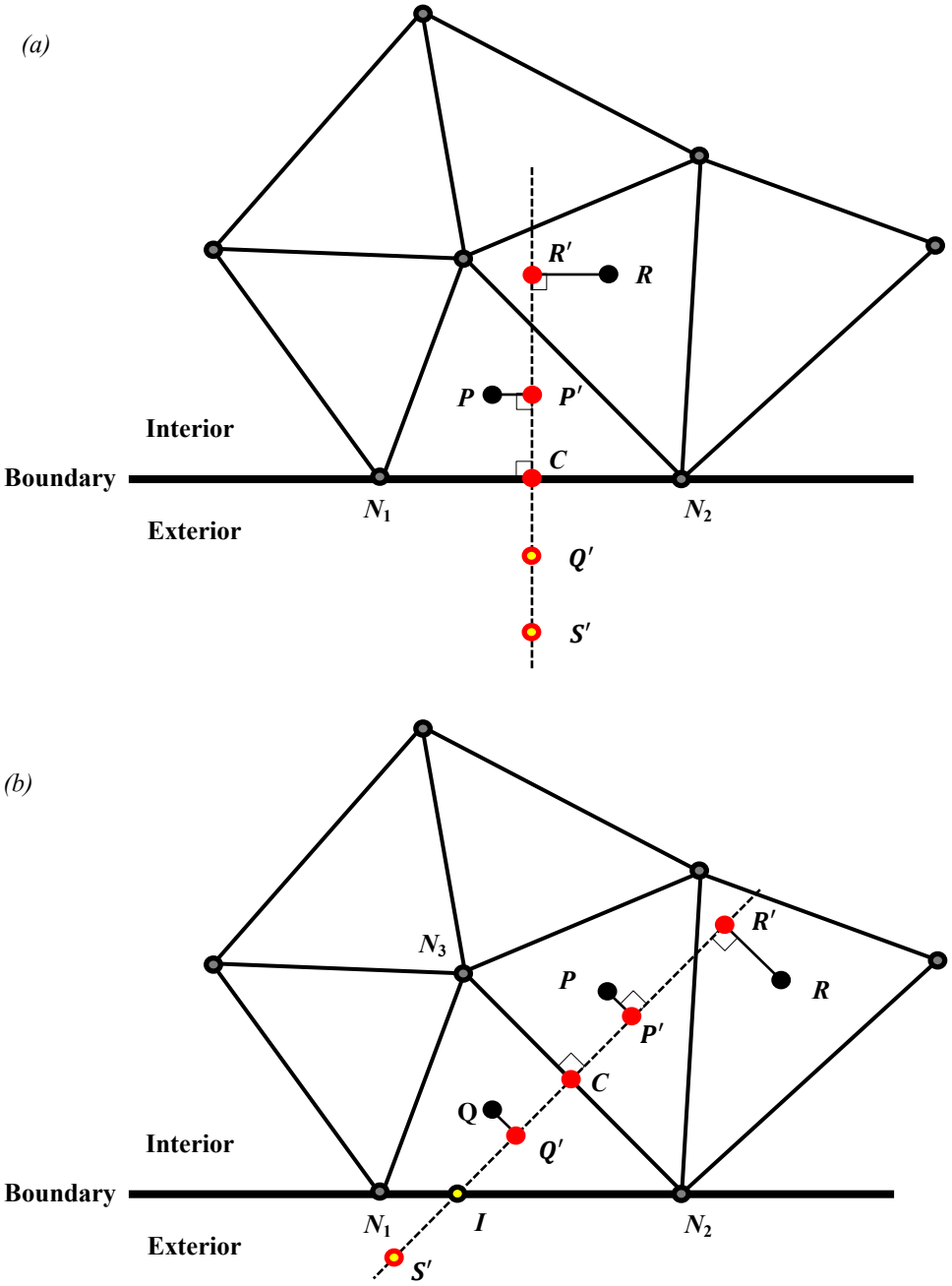


Figure 39: Construction of ghost stencil points. (a) Scenario one: the owner face is on the boundary; (b) Scenario two: the owner face is in the interior but close to the boundary

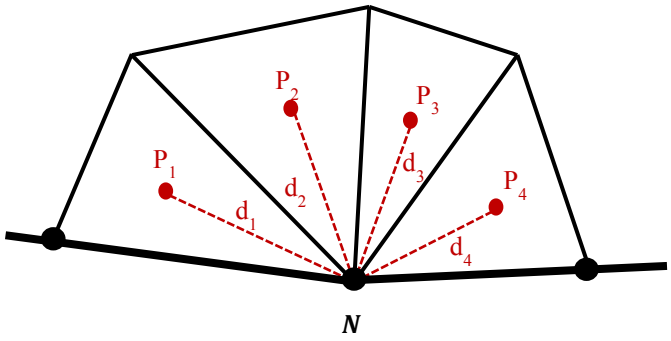


Figure 40: Evaluation of nodal PDF on the boundary

where $f_i(I)$ is interpolated with the PDFs at boundary nodes N_1 and N_2 as well, but with a more general form as:

$$f_i(I) = \frac{|N_2 I|}{|N_1 N_2|} f_i(N_1) + \frac{|N_1 I|}{|N_1 N_2|} f_i(N_2) \tag{6.7}$$

It is worth mentioning that in the real case, the ghost stencil points might be R' and P' in Fig. 39(a) and R' in Fig. 39(b), depending on the location of the face and the orientation of its stencil.

Once the stencil is complete, the flux through the owner face is calculated with the same flux schemes developed in Chap. 4. Now, the question is how to evaluate the PDFs at N_1 and N_2.

6.2.2 Evaluation of the nodal PDF on the boundary

For any boundary node, N, as shown in Fig. 40, P_1, P_2, P_3, and P_4 are the centroids of all of the triangular cells connected to N, and d_1, d_2, d_3, and d_4 the distances between the corresponding centroid and N. It should be noted that even though there are four connected cells in Fig. 40, the number of such connected cells in the actual case could be arbitrary. For any variable χ, a scalar or vector, its value at point N can be extrapolated with a function Ψ such that:

$$\chi(N) = \Psi[\chi(P)] = \frac{\sum_{j=1}^{X} \frac{\chi(P_j)}{d_j}}{\sum_{j=1}^{X} \frac{1}{d_j}} \tag{6.8}$$

where X is the total number of the connected cells. This extrapolation scheme is fully compatible with different boundary geometries and local mesh structures. It works on both flat and curved boundaries, and it treats each boundary node in the same way without considering how many cells are connected to it or how each connected cell is sized or shaped. Instead of manipulating the total PDF at the boundary nodes, here it is chosen to decompose the total PDF, according to Guo's scheme [40,41], into an equilibrium part and non-equilibrium part as:

$$f_i(N) = f_i^{eq}(N) + f_i^{neq}(N) \tag{6.9}$$

The equilibrium part is obtained by using Eqs. (1.15), (1.16), and (1.17), or by using the formulas for other lattice types, depending on the type of lattice applied, with the macroscopic variables that are defined on the boundary. Therefore, all of the components of the equilibrium PDFs can be computed at one time no matter what type of lattice model is applied. In Guo's scheme [40, 41], $f_i^{neq}(N)$ is computed using an extrapolation along the directions of characteristic velocities. Here it is modified by using Eq. (6.8), in which χ is replaced by f_i^{neq}. All of the components of the non-equilibrium PDFs are computed in the same way, and they are lattice-type-independent. Therefore, the entire scheme can be applied to any type of lattice, including the lattices in Fig. 2.

With this proposed boundary scheme, the Dirichlet boundary conditions (both density and velocities) can be preserved exactly. According to Eqs. (1.8) , at any point $\boldsymbol{x}$ in the flow domain, it holds that:

$$\begin{bmatrix} \rho(\boldsymbol{x}) \\ \rho(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}) \end{bmatrix} = \sum_{i=0}^{M-1} \begin{bmatrix} f_i(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i(\boldsymbol{x}) \end{bmatrix} = \sum_{i=0}^{M-1} \begin{bmatrix} f_i^{eq}(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i^{eq}(\boldsymbol{x}) \end{bmatrix} \tag{6.10}$$

Replacing $f_i(\boldsymbol{x})$ with $f_i^{eq}(\boldsymbol{x}) + f_i^{neq}(\boldsymbol{x})$, then Eq. (6.10) becomes:

$$\begin{aligned} \sum_{i=0}^{M-1} \begin{bmatrix} f_i(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i(\boldsymbol{x}) \end{bmatrix} &= \sum_{i=0}^{M-1} \begin{bmatrix} f_i^{eq}(\boldsymbol{x}) + f_i^{neq}(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i^{eq}(\boldsymbol{x}) + \boldsymbol{\xi}_i f_i^{neq}(\boldsymbol{x}) \end{bmatrix} \\ &= \sum_{i=0}^{M-1} \begin{bmatrix} f_i^{eq}(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i^{eq}(\boldsymbol{x}) \end{bmatrix} + \sum_{i=0}^{M-1} \begin{bmatrix} f_i^{neq}(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i^{neq}(\boldsymbol{x}) \end{bmatrix} \\ &= \sum_{i=0}^{M-1} \begin{bmatrix} f_i^{eq}(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i^{eq}(\boldsymbol{x}) \end{bmatrix} \end{aligned} \tag{6.11}$$

From Eq. (6.11), it can be deduced that:

$$\sum_{i=0}^{M-1} \begin{bmatrix} f_i^{neq}(\boldsymbol{x}) \\ \boldsymbol{\xi}_i f_i^{neq}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix} \tag{6.12}$$

Performing the recovery of macroscopic variables on Eq. (6.10) at boundary node N:

$$\begin{bmatrix} \rho(N) \\ \rho(N)\boldsymbol{u}(N) \end{bmatrix} = \sum_{i=0}^{M-1} \begin{bmatrix} f_i(N) \\ \boldsymbol{\xi}_i f_i(N) \end{bmatrix} = \sum_{i=0}^{M-1} \begin{bmatrix} f_i^{eq}(N) \\ \boldsymbol{\xi}_i f_i^{eq}(N) \end{bmatrix} + \sum_{i=0}^{M-1} \begin{bmatrix} f_i^{neq}(N) \\ \boldsymbol{\xi}_i f_i^{neq}(N) \end{bmatrix} \tag{6.13}$$

Since $f_i^{eq}(N)$ is computed with given boundary conditions, the recovery of density and velocities from $f_i^{eq}(N)$ will simply yield the boundary conditions as:

$$\sum_{i=0}^{M-1} \begin{bmatrix} f_i^{eq}(N) \\ \boldsymbol{\xi}_i f_i^{eq}(N) \end{bmatrix} = \begin{bmatrix} \rho_{BC}(N) \\ \rho_{BC}(N)\boldsymbol{u}_{BC}(N) \end{bmatrix} \tag{6.14}$$

According to Eq. (6.8), $f_i^{neq}(N)$ is linearly extrapolated using the internal centroids, at which point Eq. (6.12) is automatically satisfied. So:

$$\sum_{i=0}^{M-1} \begin{bmatrix} f_i^{neq}(N) \\ \boldsymbol{\xi}_i f_i^{neq}(N) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix} \tag{6.15}$$

Substituting Eq. (6.14) and (6.15) into (6.13):

$$\begin{bmatrix} \rho(N) \\ \rho(N)\boldsymbol{u}(N) \end{bmatrix} = \begin{bmatrix} \rho_{BC}(N) \\ \rho_{BC}(N)\boldsymbol{u}_{BC}(N) \end{bmatrix} \tag{6.16}$$

Therefore, the macroscopic variables at any boundary node are the exact Dirichlet boundary conditions at that node, which is the theoretical foundation that the proposed boundary treatment is able to preserve the Dirichlet boundary conditions up to machine accuracy.

6.3 REALIZATION OF DIFFERENT PHYSICAL BOUNDARY CONDITIONS

In the previous section, the PDF evaluation at the ghost stencil points was introduced first, followed by the boundary nodal PDF calculation. However, when using the proposed boundary treatment in the real numerical model, the execution order should be reversed, which is a three-step procedure illustrated in Fig. 41. With the developed treatment, all

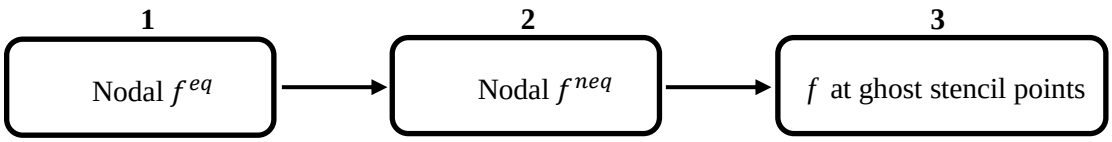


Figure 41: Execution order of the proposed boundary treatment

commonly used boundary conditions can be realized in a unified way. The evaluation of the equilibrium part of the nodal PDFs on the boundaries varies slightly depending on what type of physical boundary condition is present (density or velocity). Then the non-equilibrium part is evaluated in the same way by Eq. (6.8) for both density and velocity Dirichlet boundary conditions. Step 1 and 2 in Fig. 41 can be switched. The last step is the evaluation of the PDFs at the ghost stencil points. For the Dirichlet boundary conditions, Eqs. (6.2) and (6.3) or Eq. (6.6) are applied to linearly extrapolate the PDFs at the ghost stencil points. There are exceptions for the non-Dirichlet boundary conditions (the periodic and fully-developed boundary), which will be explained later.

6.3.1 Periodic boundary condition

The faces on the periodic boundaries are treated as internal ones, so the fluxes through those faces are computed with the same flux schemes developed in Chap. 4. Neither the PDF evaluation for the ghost stencil points nor the nodal PDF evaluation is needed. However, extra effort is required to find the periodic neighbor cell to form a complete stencil.

6.3.2 Velocity boundary condition

The velocity boundaries contain the wall boundaries, including a stationary wall and a moving wall, and open velocity boundaries, such as the velocity inlet and velocity outlet. However, when the velocity is prescribed on these boundaries, the density is usually unknown. Here, it is chosen to extrapolate the density at the boundary nodes with Eq. (6.8). The given velocity and extrapolated density can then be used to compute the equilibrium PDFs.

It should be noted that, when the velocity is given on the boundaries, the x and y velocities are not required to be defined at the same time. Sometimes, only the x or y velocity is defined. Under such a scenario, the velocity component that is not defined on the boundary is extrapolated with Eq. (6.8).

6.3.3 Density boundary condition

A density or pressure boundary usually occurs at the inlet or outlet of flows. However, the velocity there is typically unknown. Equation (6.8) is used again to extrapolate the velocity from the interior cells. Then the equilibrium PDFs are calculated with the given density and extrapolated velocity.

6.3.4 Other boundary conditions

Sometimes density and velocity both might be defined on the boundary, which will cause a physically overconstrained condition, and therefore, a numerically overconstrained condition for the conventional LBM. However, in the current numerical model, this type of boundary condition will not lead to failure. The step of density or velocity extrapolation can simply be omitted and the equilibrium PDFs are calculated by the given density and velocities. However, the extrapolations of non-equilibrium PDFs are still needed and calculated in the same way.

Finally, there is another type of boundary condition, the fully-developed or zero-gradient boundary condition, which is usually applied to the outlet of a long channel flow and probably is the only Neumann hydrodynamic boundary condition used in the majority of flow simulations. For such a boundary condition, Eq. (6.8) is used to extrapolate the total PDFs at boundary nodes without breaking it into the equilibrium and non-equilibrium parts. This is equivalent to forcing the equality of the PDFs at the boundary and the PDFs of the nearest upstream neighbors to eliminate the gradient on the boundary. Since the gradient of the PDFs is zero on the boundary, that is also true for density and velocity. Additionally, during the step of evaluating PDFs for the ghost triangle, the linear extrapolation of Eqs. (6.2) and (6.3) should be replaced by $f_i(Q') = f_i(P')$ and $f_i(S') = f_i(Q')$ in Fig. 39(a), and Eq. (6.6)

by $f_i(S') = f_i(Q')$ in Fig. 39(b), which will eliminate the gradient of PDFs between the ghost cell and its interior neighbor. As a result, the nodal PDFs on the boundary become dummies for the evaluation of the PDFs of the ghost stencil points. However, they are still kept for the sake of recovering and displaying the macroscopic variables on the boundaries.

6.4 NUMERICAL RESULTS AND DISCUSSIONS

Besides the need to prove whether the proposed boundary treatment is able to fulfill the five requirements listed at the beginning of this chapter (see the end of Sec. 6.1), another important purpose of the numerical testing in this chapter is to quantify the order of accuracy of the flux schemes in Chap. 4. Both the IRT and general meshes (Fig. 10) will be used in the numerical studies for different purposes.

6.4.1 Couette flow

This flow has a moving wall on the top and a stationary wall at the bottom, which are both treated as velocity boundaries but with different velocities (Sec. 6.3.2). The left and right boundaries are periodic (Sec. 6.3.1). The analytical solution for the x velocity is a linear profile between zero and the moving velocity of the top wall. The moving wall is given a horizontal moving velocity $u_0 = 0.1$ such that the Mach number is $Ma = 0.1732$ with a D2Q9 lattice. The relaxation time is $\lambda = 0.006$. In order to perform the grid convergence study for the order of accuracy, three IRT meshes are prepared: 18×18, 36×36, and 72×72, which double the mesh resolution in the x and y directions each time. The same parameter Δx defined in Eq. (4.33) is used to quantify the mesh resolution. The relative x velocity error in the L_2 norm is used for the study, which is defined as:

$$Err_u = \frac{\|u_a - u_n\|_2}{\|u_a\|_2} \tag{6.17}$$

where u_a and u_n are the analytical and numerical solutions, respectively. The errors for each flux scheme at each mesh resolution are plotted in log scale in Fig. 42, in which the lines to

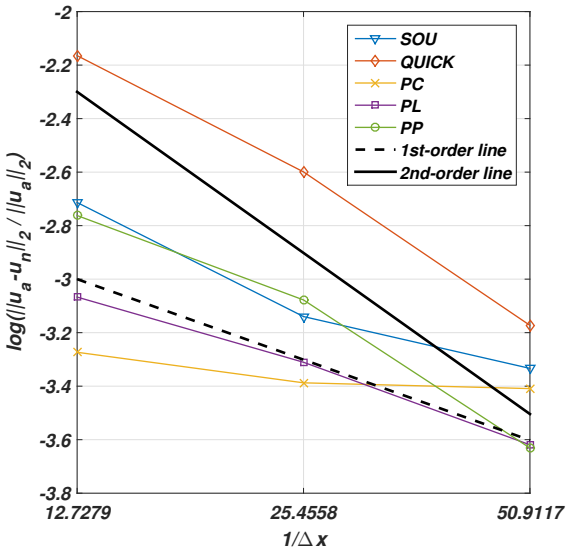


Figure 42: Grid convergence of velocity solution for Couette flow

represent the first-order and second-order slopes are also shown. Several conclusions can be made.

First, the average slope of the error from QUICK is steeper than that from SOU, which indicates that QUICK is a higher-order method than SOU. This is because QUICK utilizes one more stencil point than SOU (see Eqs. (4.7) and (4.8)). By comparing the slopes with the provided reference first-order and second-order slopes, it can be seen that QUICK, which is third-order theoretically, can basically achieve second order accuracy in actual results, and also that SOU is one order lower than QUICK both in theoretical and actual orders of accuracy.

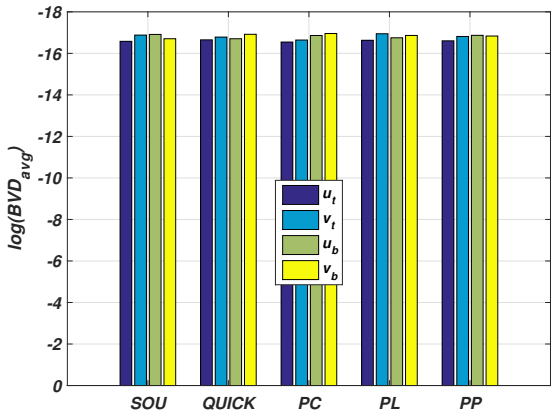
Second, for the Godunov schemes, the average slopes become steeper from PC to PL to PP. This is also because the number of utilized stencil points increases by one each time from PC to PL to PP (see Sec. 4.3.3). The slope for PC is much lower than first order and almost flat, which is evidence that the PC scheme is too erroneous and should not be chosen for the simulation of advection. The average slopes for PL and PP are one and two respectively, which indicate that the actual orders of accuracy of PL and PP are first-order and second-order (one order lower than the theoretical orders), respectively.

Third, by comparing the Godunov and non-Godunov schemes, it can be seen that the Godunov scheme is more accurate than the non-Godunov scheme with the same orders. For example, PL and SOU have the same theoretical and actual orders of accuracy, but PL is more accurate than SOU. Similarly, the lines for PP and QUICK are basically parallel in Fig. 42, which means they have the same orders of accuracy. However, PP is much more accurate than QUICK. In addition, a similar phenomenon discussed for Taylor-Green vortex flow in Sec. 4.5 can be observed here as well. In Fig. 42, it can be seen that in the low or moderate mesh resolution range, the scheme with higher order is nonetheless less accurate than the lower-order method. The higher-order scheme is more accurate than the lower-order scheme only when the mesh resolution is high enough, which is true for both the non-Godunov and Godunov schemes. Taking the Godunov schemes for the first example, at the lower mesh resolutions where $1/\Delta x$=12.7279 and $1/\Delta x$=25.4558, PC is the most accurate and PP is the least accurate. However, at the highest mesh resolution, PP is the most accurate while PC is the least accurate. Then, for the non-Godunov schemes, although SOU is always more accurate than QUICK in the provided range of mesh resolution, it will be surpassed by QUICK when the mesh resolution is high enough due to the steeper slope of QUICK. As discussed in Sec. 4.5, such a phenomenon is a direct result of using a longer stencil to evaluate the PDFs at the face center.

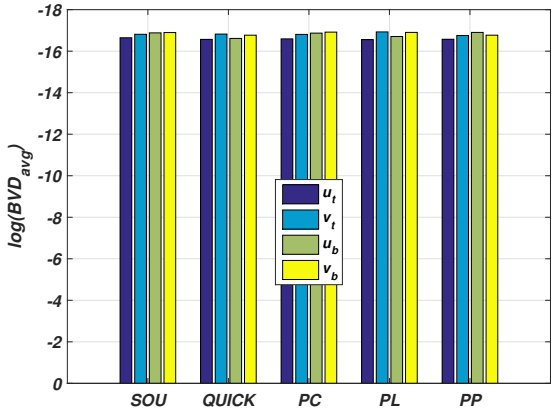
Finally, it can be concluded from the results that the developed boundary treatment can work with any flux scheme, and, more importantly, is at least second-order accurate. The reason is very simple. The order of accuracy of a solution is determined by the lowest order between the solver and boundary treatment. If the order of accuracy of the boundary treatment is lower than second order, it is impossible to achieve second order in the actual solution with any flux scheme in Fig. 42, even though that flux scheme is higher than second order.

Another task is to show the developed boundary scheme is able to preserve the physical boundary condition. In Couette flow, it is equivalent to show that the non-slip condition holds on both the moving wall on the top, and the stationary wall at the bottom. In order to expand the concept of slip velocity to any velocity boundary, a variable called boundary velocity deviation (BVD) is devised in the present work, which is the absolute difference

(a)



(b)



(c)

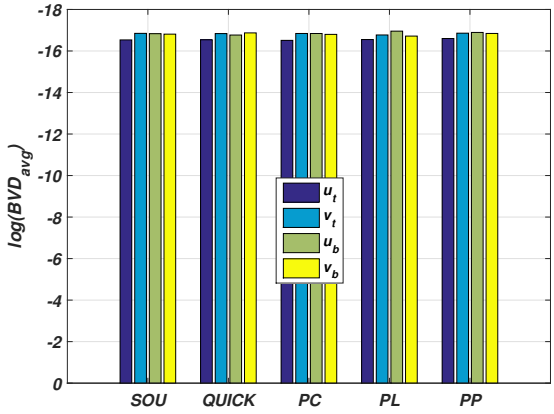


Figure 43: Boundary velocity deviation of different flux schemes for Couette flow. (a) 18×18 IRT; (b) 36×36 IRT; (c) 72×72 IRT

between the prescribed boundary velocity and real velocity on the boundary recovered from the solution. The average BVD of both the x and y velocities on the top and bottom boundaries (u_t, v_t, u_b, and v_b) are calculated for all of the flux schemes with different mesh resolutions. The results are plotted in log scale in Fig. 43. The first observation is that the BVD in the x direction on the top boundary, u_t, is larger than all other BVDs, which is expected since it is the only non-zero velocity definition among all velocity boundaries. Therefore, the larger the velocity defined on the boundary, the larger the BVD. The second observation is that the magnitudes of all BVDs are independent of mesh resolutions and flux schemes, and are all smaller than 10^{-16}, the machine epsilon. Therefore, it can be concluded that the Dirichlet velocity boundary condition can be preserved up to machine accuracy on the IRT mesh with the proposed boundary treatment.

6.4.2 Pressure-driven flow

After the previous discussions of velocity boundaries and velocity solutions in Couette flow, the density boundary and density solution are studied in this sub-section. The flow case chosen for the study is a pressure-driven flow, where a high and low pressure is defined at the flow inlet and outlet, respectively. When performing a simulation with LBE or DBE, the pressure can be directly recovered from the flow density as:

$$p = c_s^2 \rho \tag{6.18}$$

where c_s is the speed of sound that depends on the lattice models applied (Eq. (1.15d) for D2Q7, Eq. (1.16d) for D2Q9, and Eq. (1.17d) for D2Q13, etc.). Eq. (6.18) is the reason why Boltzmann-equation-based solvers are more efficient to obtain a pressure solution than NS-based solvers. The latter usually require solving a Poisson equation for pressure, which is computationally expensive.

Due to Eq. (6.18), prescribing a pressure on a boundary is equivalent to defining a density there. Therefore, the flow simulation has a high density at the inlet, ρ_{in}, and a low density at the outlet, ρ_{out}. The analytical density solution along the flow direction is a linear profile between ρ_{in} and ρ_{out}. Similar to the study for Couette flow, a grid convergence study is performed on the pressure-driven or density-driven flow, where $\rho_{in} = 2$, $\rho_{out} = 1.99$, and

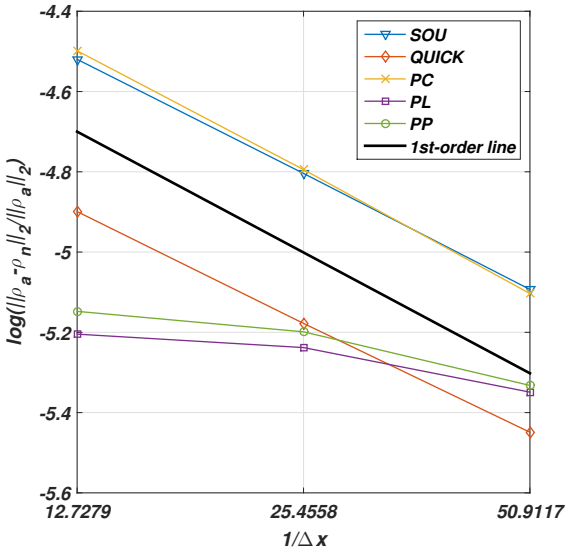


Figure 44: Grid convergence of density solution for pressure-driven flow

the other two boundaries of the rectangular domain are periodic. Also, three IRT meshes are chosen for the study: 18×18, 36×36, and 72×72. The relaxation time is defined as $\lambda = 0.006$ as well, and the same D2Q9 lattice is utilized. The error analysis is based on the relative density error in the L_2 norm, which is defined as:

$$Err_\rho = \frac{\|\rho_a - \rho_n\|_2}{\|\rho_a\|_2} \tag{6.19}$$

where ρ_a and ρ_n are the analytic and numerical solutions, respectively.

The density errors for each flux scheme at each mesh resolution are plotted in log scale in Fig. 44, in which the line representing the first-order slope is drawn as well. Several different observations can be made about the flux schemes than those developed from the velocity solutions in Couette flow.

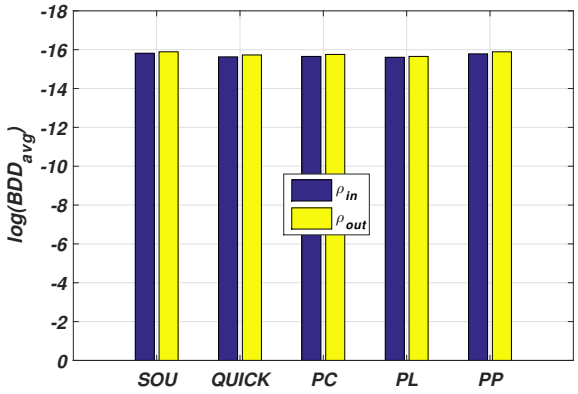
First, using a higher-order flux scheme will almost not increase the order of accuracy in solution. For example, for the SOU and QUICK schemes, at any mesh resolution, QUICK will generate a more accurate solution then SOU, but the slopes of the errors of these two schemes are parallel, which indicates that they have the same order of accuracy in the density solution. Also, for the PL and PP schemes, the average slope will become only

slightly steeper when switching from PL to PP. Such a difference in observation compared to what has been discovered in Couette flow is due to the difference between the definitions of density and velocity in the DBE. From Eqs. (1.8a) and (1.8b), it can be seen that both density and velocity are the quadratures of local PDFs; however, they have different orders of moment with respect to the lattice velocity, $\boldsymbol{\xi}_i$. On the lattice scale, density is the zeroth-order moment that is not advective at all; velocity is the first-order moment that is highly advective. Therefore, the density solution is not affected by the total viscosity, as well as the numerical viscosity introduced by the applied flux scheme. As a result, the density solution will not benefit from the decease in numerical viscosity when a higher-order flux scheme is applied, but will still become more accurate due to a longer and more accurate stencil, which is the reason why the line for QUICK is beneath that for SOU in Fig. 44, and yet they have the same slope (the same for PP and PL).

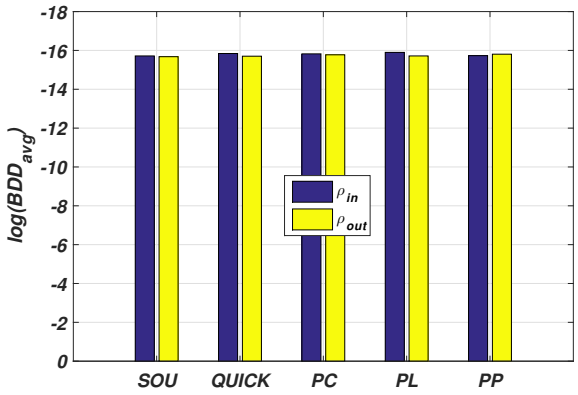
Second, the order of accuracy in the density solution for a Godunov scheme is lower than a non-Godunov scheme if they have the same theoretical order. For example, the slope for PL is flatter than for SOU although they both utilized two stencil points and have the same theoretical order of accuracy; the same fact can be observed between PP and QUICK. Such a phenomenon is also a direct consequence of the different moment orders between the density and velocity. The advantage of Godunov schemes over their non-Godunov counterparts is their advective treatment of the PDF profile across the face center. However, due to the fact that density is not advective on the lattice level, the decreased numerical viscosity of Godunov flux schemes cannot improve the accuracy in the density solution. On the other hand, Godunov flux schemes will deteriorate the density solution because they are not only spacial resolution-dependent, but also temporal resolution-dependent (see Eqs. (4.24) and (4.30)). A relatively large Δt will introduce additional error in the density solution, and therefore lower its order of accuracy.

Third, the highest order of accuracy in the density solution is one, regardless of the type and theoretical order of the flux scheme. A tentatively bold conclusion can be made by comparing this observation with the one in Fig. 42 for the velocity solution, which is that the highest order of accuracy in actual solution is one and two for density and velocity, respectively, due to the fact that density is a zeroth-order moment and velocity is a first-

(a)



(b)



(c)

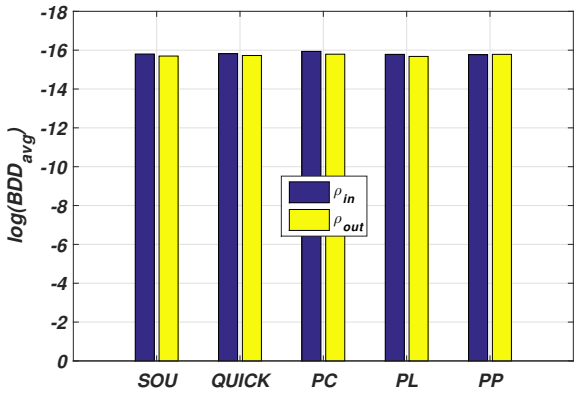


Figure 45: Boundary density deviation of different flux schemes for pressure-driven flow. (a) 18×18 IRT; (b) 36×36 IRT; (c) 72×72 IRT

order moment. In other words, the highest order of accuracy in solution for any macroscopic variable is one order more than its order of moment quadrature in a DBE solver. If this conclusion is correct, it can be predicted that the highest order of accuracy for a temperature solution could be three, since temperature is a second-order moment of the PDF (see Eq. (1.8c)). However, such a conclusion is still tentative without a solid theoretical foundation and sufficient numerical testing.

Fourth, for the density solution, the PC scheme is more like a non-Godunov type scheme, because it has the same slope as SOU and QUICK in Fig. 44.

After the analysis for the flux schemes, the boundary treatment for the density boundary conditions should studied and concluded. First, it can be concluded that the boundary treatment can also work with any flux scheme for density boundaries, and it is at least first-order accurate in the density solution. Second, it can be concluded that the Dirichlet density boundary condition can also be preserved up to machine accuracy on the IRT mesh regardless of the mesh resolution and flux scheme, which can be seen in Fig. 45 where the average boundary density deviations (BDDs), the absolute difference between prescribed boundary density and the real density on the boundary recovered from the solution, are calculated for different flux schemes under different mesh resolutions at the inlet and outlet. By comparing Fig. 43 and Fig. 45, it can be seen that the magnitude of the BDD is one order larger than that of the BVD. This is because the density boundary condition is usually one order larger than the velocity boundary condition.

6.4.3 Lid-driven square cavity flow

The third simulation case is the square cavity flow driven by a moving lid. This is also a very important benchmark problem since it is highly nonlinear, caused by the singularities at the two corners of the moving lid. The major task in this part is to demonstrate the proposed FVDBM and boundary treatment's ability to handle different lattice models, and that the Dirichlet velocity boundary condition can also be preserved no mater which lattice model is applied. Therefore, all three lattice models (D2Q7, D2Q9, and D2Q13) will be tested on the same cavity flow where the velocity of the lid is 0.1 and Re=100. The top boundary is the

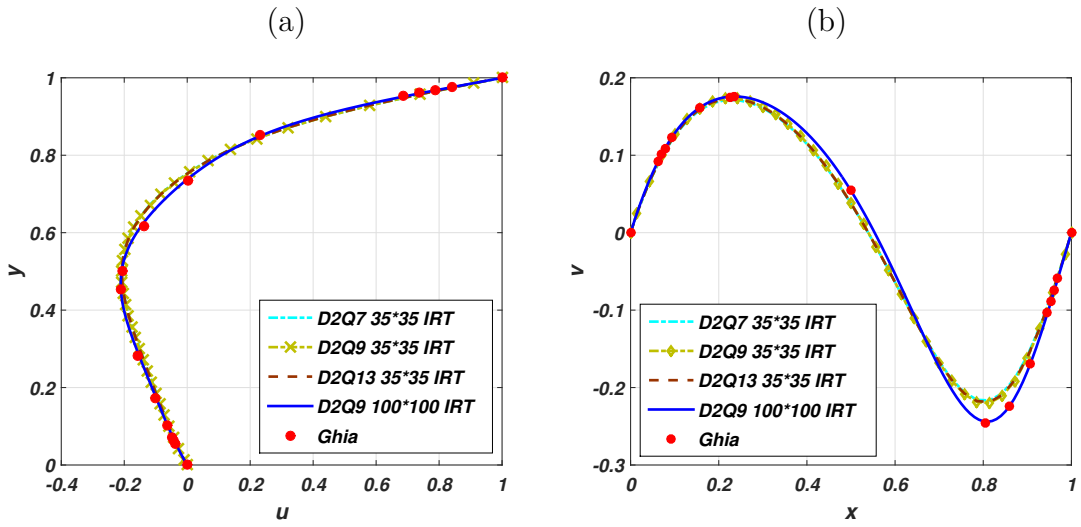


Figure 46: Velocity profiles on the vertical and horizontal mid planes of Re=100 lid-driven cavity flow. (a) u vs. y; (b) v vs. x

lid moving from left to right. All other boundaries are walls with zero velocities. The PL flux scheme is chosen for all simulations.

In order to facilitate the discussion of the following numerical results, a phenomenon that is unique to the DBE first should be presented. The DBE (Eq. (3.1)) is a hyperbolic equation with the collision term acting as a stiff source term and will become stiffer if the relaxation time decreases. In order to capture such a stiff effect correctly, the time step size must be proportional to the relaxation time, which is controlled the stability requirement due to collision (Eq. (3.32)). In addition, for any flow simulation, the time step size should be proportional to the grid size, which is due to the CFL condition. Therefore, the grid size and relaxation time are linked together by the time step size. As a result, they should also be proportional in size in order to have a stable and accurate result.

The simulations are performed on a 35×35 IRT mesh. The velocity profiles on the vertical and horizontal mid planes at steady state are shown in Fig. 46, along with the benchmark solution from Ghia [36]. The D2Q9 lattice should yield a much more accurate result than D2Q7 because the nine-velocity lattice model is able to recover the Navier-Stokes (NS)

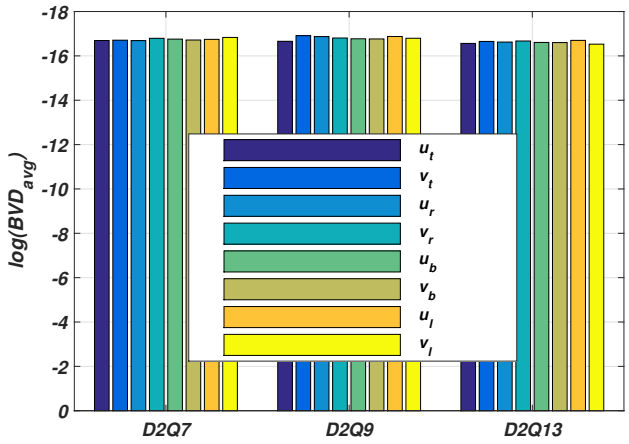


Figure 47: Boundary velocity deviation of different lattice models for lid-driven cavity flow

equation, while the seven-velocity lattice model does not have enough velocity components to do so. However, the results show that the solutions from D2Q9 are only slightly better than that from D2Q7. It should be noted that in order to achieve the same viscosity and Re, the D2Q9 model requires a smaller relaxation time than the D2Q7 model (compare Eqs. (1.15d) and (1.16d), and use (3.36)), if the size of the cavity and velocity of the moving lid are kept unchanged. Since the grid size must be proportional to the relaxation time, as previously noted, D2Q7 requires a lower mesh resolution than D2Q9 due to its larger relaxation time. Conversely, the stiffness effect of D2Q7 will be modeled with a smaller error than D2Q9 if the same mesh resolution is applied. Therefore, for the D2Q7 lattice, the error due to insufficient lattice velocity components will be mitigated by the relatively high mesh resolution. However, a D2Q9 lattice is still chosen over a D2Q7 lattice due to its better accuracy even though the D2Q7 lattice requires 20% less computation time.

Next, by comparing the results from D2Q9 and Q2Q13 in Fig. 46, it can be seen that D2Q9 and D2Q13 yield almost the same solution accuracy. The D2Q13 model should yield more accurate results than D2Q9 since it has a higher-order rotational invariance [4]. However, the results fail to prove this. The reason is the same as that for the D2Q7 and D2Q9 comparison. The D2Q13 model requires a smaller relaxation time than the D2Q9 model

in order to have the same viscosity (compare Eqs. (1.16d) and (1.17d), and use (3.36)). Therefore, D2Q13 requires a higher mesh resolution than D2Q9. Conversely, a larger error of stiffness will be introduced to D2Q13 than D2Q9 if the same mesh resolution is applied. Therefore, for the D2Q13 lattice, the extra accuracy gained by better discretization of velocity space will be offset by a relatively poor mesh resolution. In addition, the D2Q13 lattice requires almost 50% more computation time than D2Q9 since it has more PDFs to be solved. As a result, in simulations of flows with low Knudsen numbers, a D2Q9 lattice is a better choice than a D2Q7 and D2Q13 when considering both accuracy and computational efficiency. This is one of the reasons why the D2Q9 lattice dominates two-dimensional LBM simulations. Therefore, a correct way to increase accuracy is to increase the mesh resolution, not change the lattice type. A much more accurate solution with a D2Q9 lattice can be obtained with a better refined mesh, which is also shown in Fig. 46 (100×100 IRT mesh).

The effect of different lattice models on how the velocities are preserved on Dirichlet velocity boundaries was also studied. The average BVDs in the x and y directions on each boundary for different lattice models are shown in Fig. 47. It can be seen that the proposed boundary treatment is able to preserve a Dirichlet velocity boundary condition up to machine accuracy regardless of what type of lattice model is applied.

6.4.4 Flow over a circular cylinder

The last simulation case is a flow over a circular cylinder, which is included to show that the proposed FVDBM model (especially the flux schemes) and boundary treatment are able to handle curved boundaries properly by incorporating the general triangular mesh, and the boundary treatment is also able to preserve Dirichlet boundary conditions on curved boundaries with the general mesh. The geometric configuration of the computational domain and the generated mesh are shown in Fig. 48, in which D is the diameter of the cylinder.

The physical boundary conditions for each boundary are defined as follows. The left boundary, which is the inlet of the flow, is defined with a uniform x velocity and zero y velocity. Therefore, the left boundary is a Dirichlet velocity boundary. The outlet on the right is considered a fully-developed boundary, which is treated specially with the technique

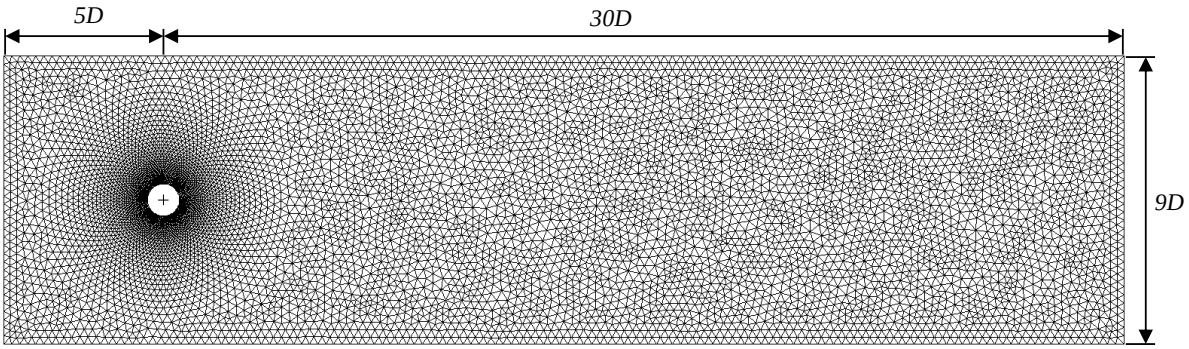


Figure 48: The configuration and mesh of the flow over a circular cylinder

in Sec. 6.3.4. The surface on the circular cylinder is an immersed boundary with the non-slip condition, in which both the x and y velocities are zero. Therefore, it is also a Dirichlet velocity boundary. The top and bottom surfaces are periodic. A D2Q9 lattice is applied for the simulation. The Reynolds number of the flow is kept at Re=20, and the Mach number at the inlet is Ma=0.1155. All of the five flux schemes are tested on the same numerical setup.

There are two velocity Dirichlet boundaries, the flat flow inlet and the curved surface of the immersed cylinder. The BVDs on these two boundaries with different flux schemes are studied first. Fig. 49 shows the average BVDs of both the x and y velocities on the inlet boundary (u_{in} and v_{in}) in log scale. It can be seen that no matter which flux scheme is adopted, the velocity Dirichlet boundary condition can be preserved up to machine accuracy on a flat boundary with a general triangular mesh. Next, the BVDs on the curved immersed boundary with different flux schemes are studied. Instead of calculating the average BVDs, the distribution of the BVD on the surface of the circular cylinder is plotted for each flux scheme. In Fig. 50, the x axis stands for the spanning angle of the cylinder surface with respect to its center, where the angles of *0* and *180* represent the front and back stagnation points, respectively. It can be seen that the velocity Dirichlet boundary condition also can be preserved up to machine accuracy on a curved boundary with a general triangular mesh, regardless of flux scheme. Assembling all of the studies about the BVD and BDD, a general

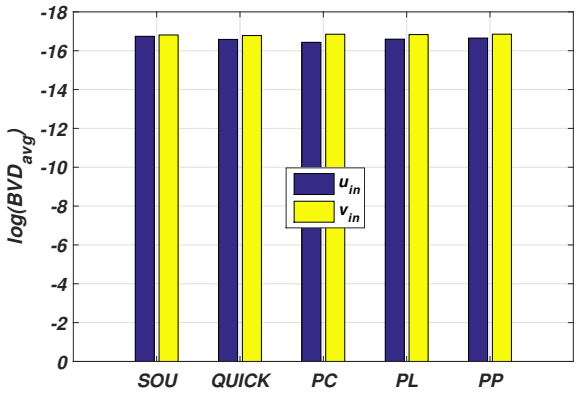


Figure 49: Boundary velocity deviation of different flux schemes with the general mesh on a flat boundary

conclusion can be made, which is that the proposed boundary treatment has the ability to preserve a Dirichlet boundary condition, density or velocity, up to machine accuracy, independent of mesh resolution, mesh topology, boundary geometry, lattice model, and flux scheme.

The velocity solutions of different flux schemes are also studied and compared with the data in other publications from various sources. At Re=20, a vortex can be formed behind the circular cylinder. Such a flow solution is very sensitive to viscosity. Therefore, a more accurate solution can be obtained with a flux scheme that is able to generate a smaller numerical viscosity. There are many ways to quantify the solution error, among which three characteristic quantities are used most commonly: L/a (the ratio of wake length to the cylinder radius), θ_s (separation angle), and C_D (drag coefficient). Usually, the larger the total viscosity (this is also true for numerical viscosity), the smaller the effective Re. Consequently, the wake length will be shorter, but the drag coefficient will increase.

The vortex structure with each flux scheme is illustrated in Fig. 51 by plotting the streamline function. It can be clearly seen that there is no vortex behind the cylinder with the PC flux scheme. This is because the PC scheme is extremely diffusive, and injects a significant amount of numerical viscosity into the flow system. Consequently, the actual Re

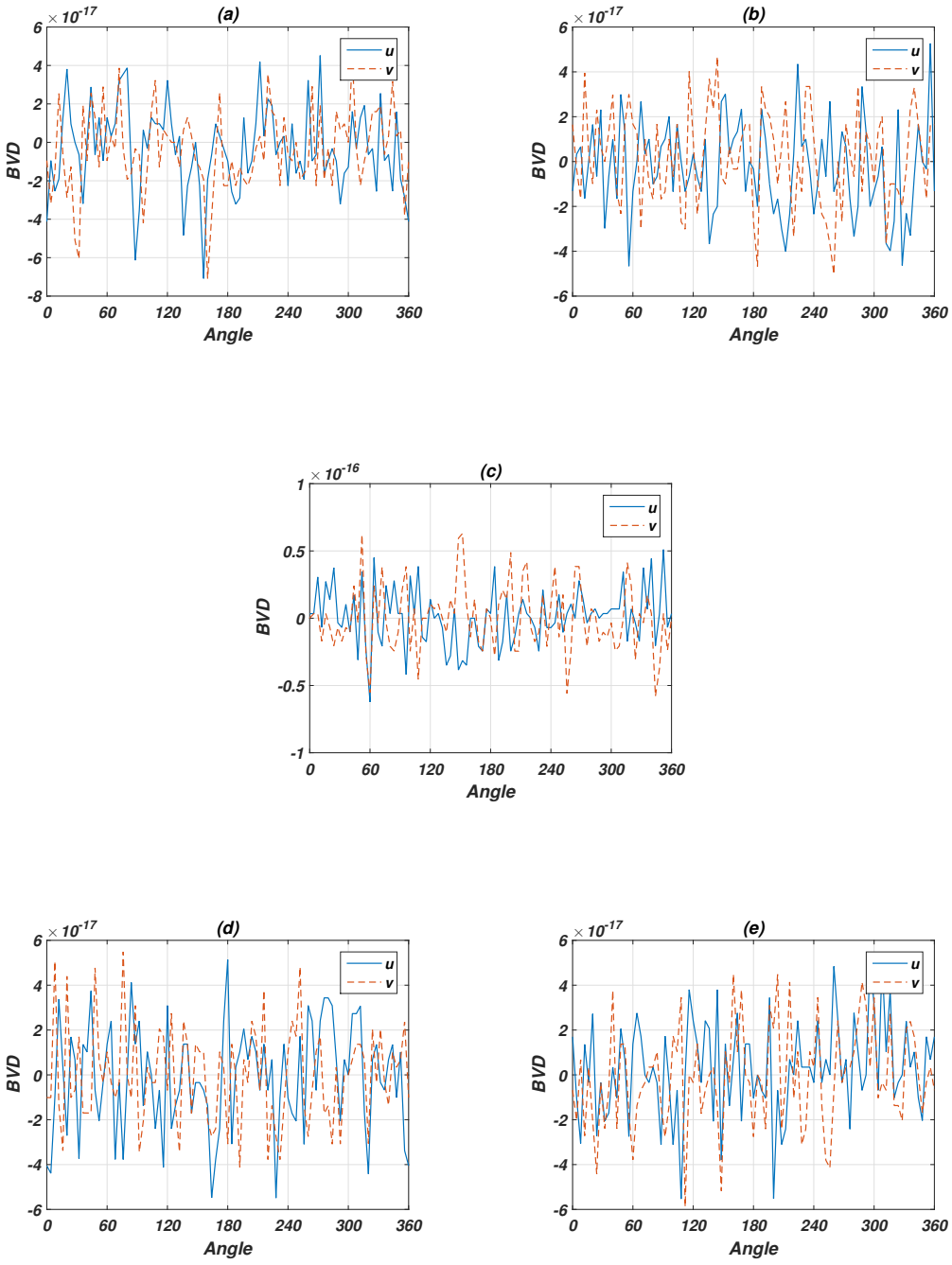


Figure 50: The distributions of boundary velocity deviation on the circular cylinder surface for different flux schemes. (a) SOU; (b) QUICK; (c) PC; (d) PL; (e) PP

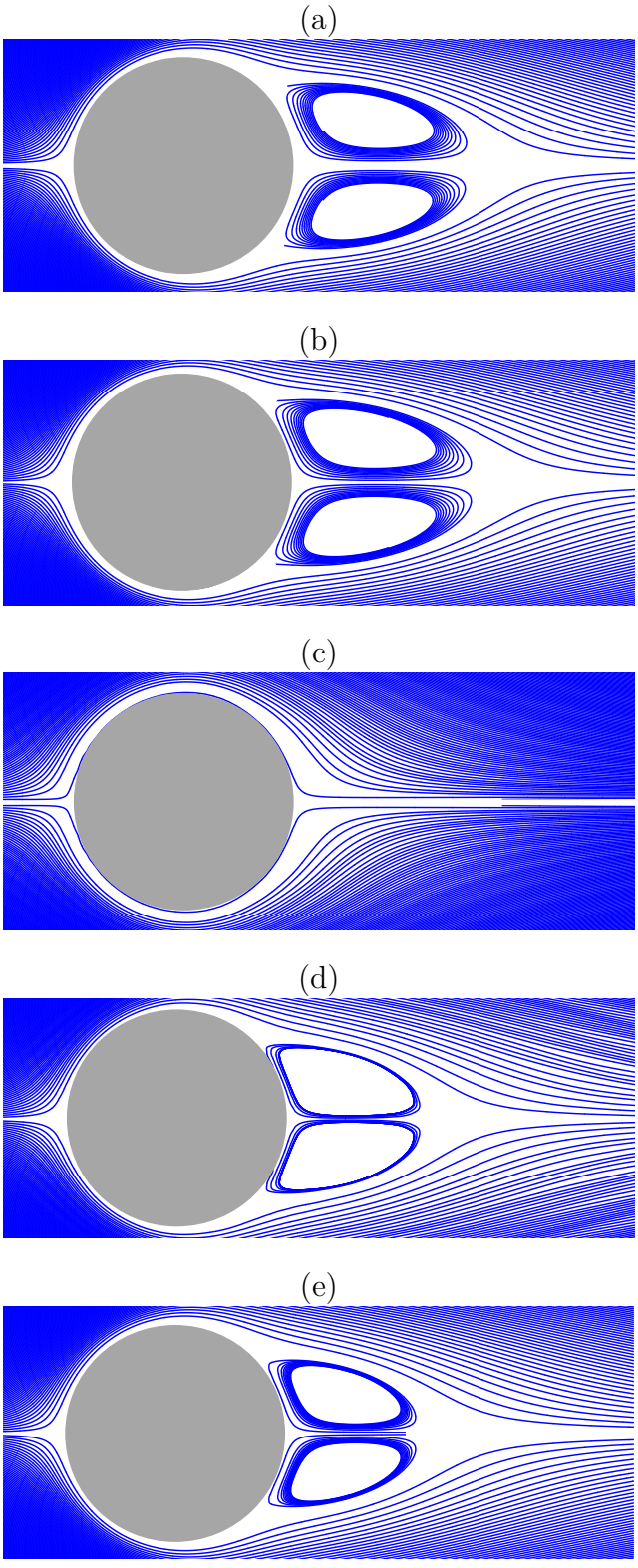


Figure 51: Vortex structures behind the circular cylinder with different flux schemes at Re=20. (a) SOU; (b) QUICK; (c) PC; (d) PL; (e) PP

of the flow is very low, which will prevent the flow separation behind the cylinder. The other four flux schemes are able to form a vortex. However, the SOU and QUICK schemes create a wake that is slightly longer than their Godunov counterparts. The quantitative measurements of the wake length and separation angle for each flux scheme are listed in Tab. 5, along with data from other numerical results with different methods. It can be seen that QUICK is more accurate than SOU, and PP is more accurate than PL, which is because the numerical viscosity will decrease when using a higher-order method, regardless of a Godunov or non-Godunov approach. However, PL is less accurate than SOU, and PP is less accurate than QUICK. This might be the result of the very refined mesh around the circular cylinder (see Fig. 48). As shown in Fig. 31 for Taylor-Green vortex flow, the numerical viscosity generated by a non-Godunov scheme will get closer to or even lower than the numerical viscosity of a Godunov scheme with the same order when the mesh resolution is high enough. Therefore, in the fine mesh region around the cylinder, the mesh might be fine enough to have a better solution with a non-Godunov scheme than a Godunov one. Another explanation lies on the difference between density and velocity from the viewpoint of moment quadrature. As discussed for Couette flow and pressure-driven flow, a non-Godunov flux scheme can generate a more accurate density solution than a Godunov scheme if they possess the same order. The vortex structure behind the cylinder heavily relies on the density solution, which can be seen from the calculation of drag coefficient in Eq. (6.25), therefore, the vortex obtained by a Godunov scheme is less accurate than the one simulated by a non-Godunov scheme.

The calculation of drag coefficient starts from the force evaluation. The total force on any boundary surface is calculated as:

$$\boldsymbol{F} = \int \boldsymbol{S} \cdot \boldsymbol{n} ds \tag{6.20}$$

where $\boldsymbol{S}$ is the stress tensor on the boundary surface and $\boldsymbol{n}$ is the unit normal vector pointing into the fluid domain. The stress tensor $\boldsymbol{S}$ is defined as:

$$\boldsymbol{S} = -p\boldsymbol{I} + \rho\nu\left(\boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^T\right) \tag{6.21}$$

where the first term is the stress tensor due to pressure p, which is calculated by Eq. (6.18) for the DBE. Since a D2Q9 lattice model is applied in this simulation, it can be obtained

with an easy relation (other lattice models have different but similar relations) from the density solution, ρ, such that:

$$p = \rho/3 \tag{6.22}$$

The second term in Eq. (6.21) is the deviatoric stress tensor caused by the flow motion. The pressure stress tensor is easy to calculate, since density can be directly obtained from the solution. However, the deviatoric stress tensor has to be numerically evaluated since the gradient of velocity $\boldsymbol{\nabla u}$ is not available in the solution. An extrapolation procedure must be applied since the solution only exists on the fluid side of the boundary. Therefore, the evaluation of the deviatoric stress tensor suffers further numerical error that depends on the order of accuracy of the applied extrapolation scheme. In order to calculate $\boldsymbol{\nabla u}$, two points at different locations have to be selected. Figure 52 shows the two points selected for the extrapolation on one segment of the boundary. The first selected point is C, which is the middle point between boundary nodes N_1 and N_2. Then the velocity at point C can be evaluated as:

$$\boldsymbol{U}_c = \frac{1}{2}\left(\boldsymbol{U}_{N_1} + \boldsymbol{U}_{N_2}\right) \tag{6.23}$$

The deviatoric stress tensor has two components, the normal stress tensor and shear stress tensor. Both components require that the direction for calculating the gradient must be perpendicular to the boundary. The dashed line that passes point C and is perpendicular to the boundary is drawn in Fig. 52. The second point must be on the dashed line as well. By using the same method for the flux scheme, the second point is chosen to be the projected point P' of centroid P onto the dashed line. Finally, the gradient of velocity can be evaluated with $\boldsymbol{U}_c$, $\boldsymbol{U}_{P'}$, and the distance between these two points. As discussed thoroughly in the previous results, the proposed boundary treatment can exactly recover the Dirichlet boundary conditions. So, $\boldsymbol{U}_c$ is the exact velocity of the boundary regardless of a moving boundary or stationary boundary. As a result, the accuracy of the extrapolation scheme is primarily determined by how accurately the velocity at point P' can be evaluated. Here, with the same assumption for the PDF evaluation at the stencil points in flux schemes, it is simply chosen that:

$$\boldsymbol{U}_{P'} = \boldsymbol{U}_P \tag{6.24}$$

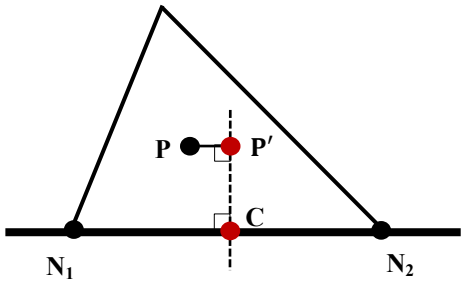


Figure 52: Stencil for the calculation of $\boldsymbol{\nabla u}$ on the boundary

Other higher-order methods could be used instead, but not discussed in the current work. After the total force is evaluated, the drag coefficient on the circular cylinder is calculated as:

$$C_D = \frac{F_D}{\rho U^2 a} \tag{6.25}$$

where F_D is the component of total force in the flow direction, ρ is the average density of the flow, U is the inlet velocity, and a is the radius of the cylinder. The drag coefficient for each flux scheme is included in Tab. 5, along with some available data from other publications. It can be seen that the PC scheme produces the largest drag, and the drags with the Godunov schemes are larger than those from the non-Godunov schemes, which accords with the wake length measurements. However, the drag coefficient calculation itself contains an error, which mainly has two sources. One is from the assumption stated in Eq. (6.24). Similar to the flux schemes, better evaluation at the projected point will improve the accuracy. Another source of error is from the calculation of the velocity gradient on the boundary. It can be seen that, with only two points, the calculated derivative will not be located on the boundary, but rather at a point in the flow domain between point P' and C in Fig. 52. The situation can be improved by using a longer stencil, similar to the flux schemes, which, however, is not addressed in the current work.

Table 5: Comparison of geometrical and dynamical parameters for flow past a circular cylinder at Re=20

Authors		L/a	θ_s	C_D
Coutanceau and Bouard [21, 67]		1.860	44.80	-
Dennis and Chang [25]		1.880	43.70	2.045
Nieuwstadt and Keller [76]		1.786	43.37	2.053
Fornberg [32]		1.820	-	2.000
Calhoun [22]		1.820	-	2.190
Ye *et al.* [111]		1.840	-	2.030
He and Doolen [45]		1.842	42.96	2.152
Mei and Shyy [72]		1.804	-	-
Lee and Lin [57]		1.834	-	2.106
Ubertini *et al.* [103]		-	-	2.090
Zarghami *et al.* [115]		1.820	42.50	2.205
Patil and Lakshmisha [79]		1.884	42.81	1.949
Present	**SOU**	**1.771**	**42.833**	**2.380**
	QUICK	**1.82**	**44.218**	**2.537**
	PC	**0**	**0**	**4.170**
	PL	**1.503**	**43.114**	**2.888**
	PP	**1.517**	**44.655**	**2.826**

7.0 FVDBM FOR THERMAL MODELS

Another important aspect of fluid problems is the thermal behavior in conjunction with the development of hydrodynamics. One of the leading research areas for the general LBM is the thermal models. Therefore, it is necessary to show that the proposed FVDBM approach (both the solver and boundary treatment) is able to solve thermal problems with existing thermal models.

7.1 BACKGROUND

Despite its significant progress in the simulations of hydrodynamics in the past decades, the general LBM suffers slow development of thermal models. The biggest challenge is the lack of microscopic-level energy conservation with desired numerical stability, especially when the flow encounters multiphase and/or large heat transfer rates.

The first type of thermal model that was developed in an early stage is the passive scalar (PS) model [30,68,88,91,105]. The PS model solves a second set of distribution functions for thermal transport, on top of the existing PDFs for hydrodynamics. The PS model is built upon the assumption that the temperature is advected by the fluid flow as a passive scalar, and does not influence the hydrodynamics of the flow, which however is not the actual case. Therefore, the PS model is only applicable for low-Re single-phase flows where the viscous heat dissipation and flow compression work can be neglected. Moreover, the PS model is not able to conserve energy fundamentally. Later, some effort was made to improve the PS model by designating the second set of PDFs for internal energy, instead of temperature. As a result, the viscous heat dissipation and flow compression work can be easily integrated

into the model. This improved version of the PS model is sometimes referred to as the double-distribution function (DDF) model [44].

Fundamentally different from the PS model, a more advanced approach uses just one set of PDFs to resolve both hydrodynamic and thermal kinetics, which is also the best method as of today to ensure energy conservation and good stability. Such a method has two distinguishing features: a lattice model that has more than nine particle velocities and a high-order equilibrium distribution function in terms of the macroscopic flow velocity that matches the multi-speed lattice. Therefore, such a model is usually called the multi-speed (MS) model [2, 15, 70, 104]. However, the MS model still suffers instability under certain scenarios, since the coefficients of the equilibrium distribution functions are acquired through matching the Chapman-Enskog expansion and the desired thermo-hydrodynamics. An amendment can be made by expanding the equilibrium distribution function on a Hermite basis. Then, the coefficients can be obtained rigorously through *a priori* techniques [92, 93].

The purpose of this chapter is to show that the proposed FVDBM is able to incorporate thermal models, instead of needing the development of new ones. Actually, it can be found that both types of thermal models (PS and MS) can be seamlessly incorporated within the proposed FVDBM approach. For the MS and DDF models, the only extra work to be done is to solve a second DBE for the thermal distribution function. The same solver and boundary treatment will be utilized for the second time. As for the MS model, none of the solvers or boundary treatments need to be modified, since both the solver and boundary treatment are able to incorporate different lattice models. Extra effort is required to obtained the desired lattice structure and corresponding high-order equilibrium distribution function, which, however, are out of the scope of current study. Therefore, only the PS model is chosen for the study in this chapter, while the generality of the conclusion will not be hampered.

7.2 FVDBM WITH PASSIVE-SCALAR THERMAL MODEL

In the passive scalar method, a second set of distribution functions, called the energy distribution function (EDF), is created to model the thermal behavior. Similar to how the

hydrodynamic microscopic variables are recovered from the PDF, the thermal macroscopic variable, temperature T, is recovered from the EDF, h_i, as follows:

$$T = \frac{1}{R\rho} \sum_{i=0}^{M-1} \boldsymbol{\xi}_i h_i \tag{7.1}$$

where R is the ideal gas constant. The evolution of the EDF is governed by the same PDE as for the PDF. So, with the same BGK collision model, the PDE for the EDF reads:

$$\frac{\partial h_i}{\partial t} + \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} h_i = \frac{1}{\vartheta} \left(h_i^{eq} - h_i \right) \tag{7.2}$$

where ϑ is the thermal relaxation time, which controls the thermal diffusivity of the flow and is not necessarily the same size as the hydrodynamic relaxation time, λ. The thermal diffusivity for the D2Q9 lattice is defined as:

$$\alpha = \frac{2}{3}\vartheta \tag{7.3}$$

Then, the last thing needed to complete the passive scalar approach is to establish the explicit form for the equilibrium EDF. Similar to the PDF, the expression of equilibrium EDF depends on the type of lattice model applied. For the same D2Q9 lattice (see Fig. 2 and Eq. (1.16a)), the equilibrium EDF [44, 84] reads:

$$h_i^{eq} = \begin{cases} \sigma_i \rho R T \left(\frac{\boldsymbol{u}\cdot\boldsymbol{u}}{3c_s^2} \right) & i = 0 \\ \sigma_i \rho R T \left[\frac{3}{2} + \frac{\boldsymbol{\xi}_i \cdot \boldsymbol{u}}{2c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2c_s^2} \right] & i = 1-4 \\ \sigma_i \rho R T \left[3 + \frac{3(\boldsymbol{\xi}_i \cdot \boldsymbol{u})}{c_s^2} + \frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2c_s^2} \right] & i = 5-8 \end{cases} \tag{7.4a}$$

$$\sigma_i = \begin{cases} -2/3 & i = 0 \\ 1/9 & i = 1-4 \\ 1/36 & i = 5-8 \end{cases} \tag{7.4b}$$

It can be seen that, by comparing Eqs. (1.16) and Eqs. (7.4), the passive scalar method may have the equilibrium PDF and equilibrium EDF constructed differently for the same lattice structure. In order to solve a thermal system with the passive scalar approach, two DBEs have to be solved, with one for the PDF and the other for the EDF. It is worth noting that the evolution of the EDF is affected by the PDF. For example, the recovery of temperature involves the density ρ (see Eq. (7.1)); and the calculation of the equilibrium EDF needs velocity $\boldsymbol{u}$ as one of its inputs (see Eqs. (7.4)). This is why the EDF is considered

a passive scalar of the PDF, which is passively affected by the hydrodynamics of the flow. It is also important to mention that the lattices for the PDF and EDF can be chosen totally independently. For example, some researchers use the D2Q5 lattice for the EDF and D2Q9 for the PDF in their passive-scalar models [73, 106].

Since the EDF is governed by exactly the same DBE as the PDF, all of the solution techniques developed for solving the PDF in the previous chapters, especially the flux schemes in Chap. 4, can be applied to the EDF without any change or modification. In addition, the three-step boundary treatment developed for hydrodynamic variables ρ and $\boldsymbol{u}$ can also be utilized to realize the temperature boundary condition T. In the first step, the equilibrium energy distribution at any boundary node $h_i^{eq}(N)$ is calculated with Eqs. (7.4), in which the nodal density $\rho(N)$, velocity $\boldsymbol{u}(N)$, and temperature $T(N)$ are obtained either by the boundary condition or the extrapolation scheme Eq. (6.8) (not for temperature); in step two, the non-equilibrium EDF $h_i^{neq}(N)$ is calculated with the the extrapolation scheme Eq. (6.8); in the last step, the EDF at the ghost stencil point(s) are evaluated with the nodal EDF obtained in the previous two steps.

Similar to density and velocity, the temperature Dirichlet boundary condition can also be preserved with the boundary treatment. For any boundary node N, it holds that:

$$T(N)\rho(N) = \frac{1}{R}\sum_{i=0}^{M-1}\boldsymbol{\xi}_i h_i(N) = \frac{1}{R}\sum_{i=0}^{M-1}\boldsymbol{\xi}_i h_i^{eq}(N) + \frac{1}{R}\sum_{i=0}^{M-1}\boldsymbol{\xi}_i h_i^{neq}(N) \tag{7.5}$$

Since $h_i^{eq}(N)$ is computed with given boundary conditions, the recovery of macroscopic variables from $h_i^{eq}(N)$ will simply yield the boundary conditions as:

$$\sum_{i=0}^{M-1}\boldsymbol{\xi}_i h_i^{eq}(N) = RT_{BC}(N)\rho_{BC}(N) \tag{7.6}$$

According to Eq. (6.8), $h_i^{neq}(N)$ is linearly extrapolated using the internal centroids, so:

$$\sum_{i=0}^{M-1}\boldsymbol{\xi}_i h_i^{neq}(N) = 0 \tag{7.7}$$

Substituting Eq. (7.6) and (7.7) into (7.5):

$$T(N)\rho(N) = T_{BC}(N)\rho_{BC}(N) \tag{7.8}$$

In conclusion, together with the hydrodynamic boundary condition, the proposed boundary treatment can preserve any component of the vector $\boldsymbol{V} = (\rho, \boldsymbol{u}, T) = (\rho, u, v, T)$ on the boundary up to machine accuracy.

7.3 NUMERICAL VALIDATION

The chosen flow case for the validation study is thermal Couette flow. Besides the existing velocity boundaries conditions, different temperatures are also defined on the top and bottom walls. When reaching a steady-state solution, a temperature profile that is both affected the viscosity (or momentum diffusivity) and thermal diffusivity can be obtained analytically as:

$$\theta^* = y^* + \frac{Br}{2} y^* \left(1 - y^*\right) \tag{7.9}$$

where θ^* and y^* are the normalized temperature and the distance in the y location with respect to the bottom wall. They are defined as:

$$\theta^* = \frac{\theta - \theta_b}{\theta_t - \theta_b} \tag{7.10}$$

$$y^* = \frac{y}{H} \tag{7.11}$$

where θ_b and θ_t are the temperatures on the bottom and top walls, respectively, and H is the height of the flow. The Brinkman number is $Br = Pr \cdot Ec$, in which the Prandtl number Pr and the Eckert number Ec are defined as:

$$Pr = \frac{\nu}{\alpha} \tag{7.12}$$

$$Ec = \frac{u_0^2}{C_p \left(\theta_t - \theta_b\right)} \tag{7.13}$$

where ν and α are defined by Eq. (3.36) and (7.3), respectively, for a D2Q9 lattice. Additionally, u_0 is the horizontal moving velocity of the top wall, and C_p is the specific heat, which equals *2* for incompressible flows in 2D DBE simulations [66, 96].

The dimensionless number Br characterizes the temperature profile. The larger the Br, the more curved the temperature profile will be towards the flow downstream; the smaller the Br, the more flattened the temperature profile will be. When $Br = 0$, the temperature profile is a straight line changing from θ_b to θ_t. A flow case where $Br = 5$ is chosen for the numerical testing. The steady-state temperature solutions with all of the five flux schemes on a 72×72 IRT mesh are plotted in Fig. 53, along with the analytical solution. If simply glancing at the data in Fig. 53, it can be seen that the PP scheme generates the most accurate data. However, with a careful analysis, it can be found that with the current numerical setup,

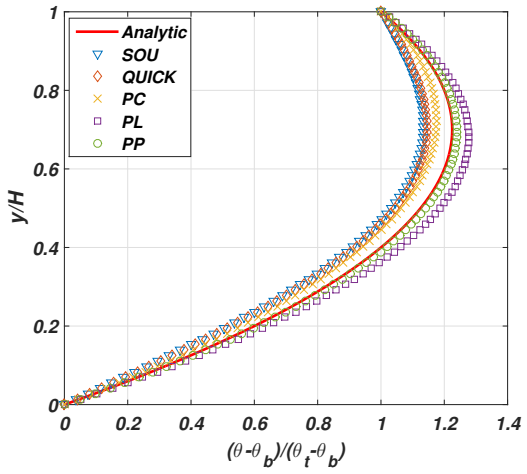


Figure 53: Temperature profiles of different flux schemes for thermal Couette flow at $Br = Pr \cdot Ec = 5$

the PL scheme generates the least numerical diffusivity (equivalent to numerical viscosity for hydrodynamic solutions), and therefore is the most accurate scheme. Similar to Eqs. (3.35) and (3.37), a thermal flow system has an actual thermal diffusivity α_a that is the sum of the theoretical thermal diffusivity α_t and numerical thermal diffusivity α_n. Mathematically speaking:

$$\alpha_a = \alpha_t + \alpha_n \tag{7.14}$$

In addition, α_n also has three contributors:

$$\alpha_n = \alpha_{fs} + \alpha_{tms} + \alpha_{other} \tag{7.15}$$

where the numerical thermal diffusivity introduced by the flux scheme α_{fs} is always positive; the α_{tms} (time marching scheme) and/or α_{other} (boundary treatment, BGK collision, etc.) may introduce negative numerical thermal diffusivity. For SOU, QUICK, and PC, even though α_{tms} and α_{other} are negative, the positive α_{fs} is large enough to make sure α_n is positive; therefore, $\alpha_a > \alpha_t$, which is why their temperature profiles are located on the left hand side of the analytical profile. For PL and PP, the magnitude of their positive α_{fs} is smaller than the magnitude of the negative α_{tms} and α_{other}; therefore, $\alpha_n < 0$ and

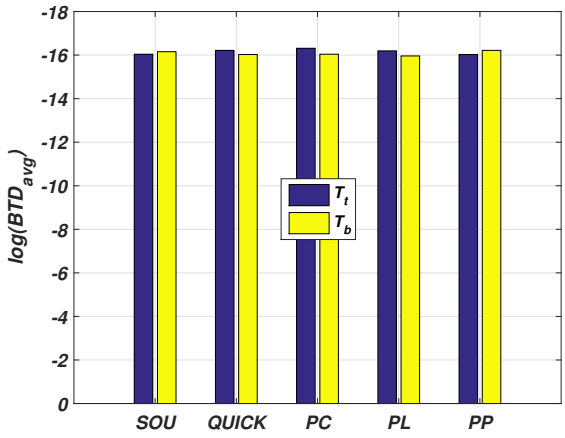


Figure 54: Boundary temperature deviation of different flux schemes for thermal Couette flow

$\alpha_a < \alpha_t$. The resulting temperature profile is on the right hand side of the analytical one. When comparing PL and PP, since the PL flux scheme itself can generate less numerical thermal diffusivity than PP ($0 < \alpha_{fs,PL} < \alpha_{fs,PP}$), $\alpha_{n,PL} < \alpha_{n,PP} < 0$, and consequently $\alpha_{a,PL} < \alpha_{a,PP} < \alpha_t$. However, the investigation of the sources of negative thermal diffusivity is out of the scope of current study.

Finally, how the temperature boundary conditions are preserved is studied. Another parameter called the boundary temperature deviation (BTD), similar to the BVD and BDD, is devised to quantify the errors on the boundary. The average BTDs on the top and bottom surfaces are calculated and plotted in log scale in Fig. 54. It can be seen that the Dirichlet temperature boundary conditions can also be preserved up to machine accuracy. Therefore, together with the studies for BVD and BDD, it can be concluded that the proposed boundary treatment can preserve the Dirichlet boundary condition for any component of the vector $\boldsymbol{V} = (\rho, \boldsymbol{u}, T) = (\rho, u, v, T)$ up to machine accuracy.

8.0 CONCLUSIONS AND FUTURE WORK

The current work provides a unique self-contained methodology to solve the DBE on an unstructured mesh for a variety of hydrodynamic and thermal flow problems with simple and complex boundary geometries. However, there is still room to improve the current model. In this chapter, the major contributions of the present work are concluded, followed by several prospective research directions for future model development and improvement.

8.1 CONTRIBUTIONS

8.1.1 A FVDBM platform for triangular unstructured mesh

Targeting flow problems with complex boundary geometries, a self-contained 2D FVDBM approach for both isothermal and thermal flows was developed from scratch in the current work. The platform consists of three components, the unstructured mesh, the FVDBM solver, and the boundary treatment.

The current work uses only the triangular mesh, due to its good adaptability to complex geometries. However, the model can be modified to incorporate other types of mesh elements, such as rectangular, pentagonal, hexagonal, or even mixed type. In terms of the mesh data structure, there are two types of triangular meshes, the Cell-Centered (CC) and the Vertex-Centered (VC). Due to its superiority in realizing the boundary conditions, the CC triangular mesh is chosen for the current study, which is not commonly used in the literature to date.

The second component of the platform is the FVDBM solver. Due to the nature of the DBE, the solver has three separate parts for each term in the DBE, the collision calculator

for the collision term, the flux scheme for the gradient term, and the time marching scheme for the temporal term. The flux scheme is the most important one among the three, and, therefore, became one of the major contributions in the current work (see Sec. 8.1.2, below, for the detailed contribution).

The last component of the platform is the boundary treatment, which is another major contribution of the current work (see Sec. 8.1.3 for the detailed contribution).

8.1.2 Flux schemes

Due to its Eulerian nature, the diffusion error of the FVDBM is intrinsically larger than that of the conventional LBM. In the FVDBM, the diffusion error is directly linked to the flux scheme. Therefore, it is necessary to study and compare as many flux schemes as possible. In the current work, for the first time, a universal stencil that allows a systematic formulation of flux schemes with different orders of accuracy was developed on the unstructured triangular mesh. In addition, this is the first time to classify the flux schemes into two categories: Godunov type and non-Godunov type. In each category, several flux schemes with different orders of accuracy were systematically developed. Furthermore, most of the resulting flux schemes (SOU, QUICK, PL, and PP) were utilized in the Boltzmann simulations for the first time. With thorough numerical validations, the Godunov schemes are recommended due to their lower diffusion error for higher-order quadrature moments (velocity and temperature).

8.1.3 Boundary treatment

The primary motivation for the current work is to incorporate complex boundaries with an unstructured mesh, which is also the minimum requirement for the boundary treatment. A unique three-step boundary treatment that has not been reported in existing publications was developed. It has many desired features beyond the minimum requirement, listed as follows:

- Different physical boundary conditions, which include the Dirichlet type (density and velocities boundaries) and the non-Dirichlet type (periodic and fully-developed boundaries), can be realized in a unified way;

- Different lattice models can be incorporated in a unified way;
- The boundary treatment can work with different flux schemes;
- The boundary treatment is at least second-order accurate for velocity boundary conditions, and at least first-order accurate for density boundary conditions; and
- The boundary treatment can preserve the Dirichlet condition for any component of the vector $\boldsymbol{V} = (\rho, \boldsymbol{u}, T) = (\rho, u, v, T)$ up to machine accuracy, regardless of changes made in a number of different factors, including the mesh structure on the boundaries, the geometry of the boundaries (flat or curved), the mesh resolution, the type of lattice model, and the type of the flux scheme.

8.2 FUTURE WORK

8.2.1 Decreasing the diffusion error

As shown by many numerical results in the previous chapters, the diffusion error of the overall model is still relatively large. Therefore, there is a need to increasingly reduce the diffusion error. One approach is to develop even higher-order flux schemes, such as the fourth-order and fifth-order. However, such an approach is not plausible. On the one hand, a higher-order scheme requires another stencil that is longer than the one in the current work. Such a prolonged stencil will be very difficult to construct on an unstructured mesh; on the other hand, even when such a stencil is available, the resulting flux scheme will be very computationally expensive. As displayed in Sec. 6.4.1, the theoretical order of accuracy of a flux scheme can be reduced by one order due to the diffusion error, no matter if it is Godunov type or non-Godunov type. So, a feasible approach is to narrow down the gap between the theoretical and actual orders for any flux scheme, which is equivalent to increasing the order of accuracy, not for one, but for all of the flux schemes.

As illustrated in Fig. 22 and Fig. 23 in Chap. 3 and Chap. 4, the stencil points are generally not the data points (the centroids of the CV). Therefore, the assumption that the PDFs at the stencil points are equal to the PDFs at the data points (Eqs. (3.29) and (4.1))

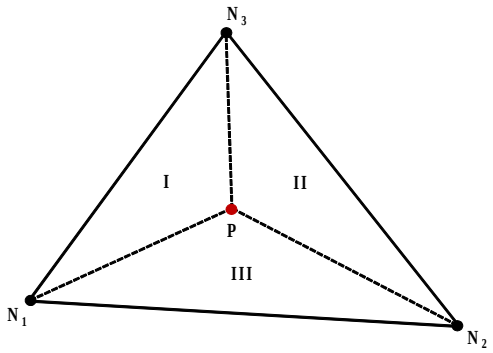


Figure 55: The nodal interpolation mapping

will introduce some diffusion error. Such an assumption can be called first-order mapping, which assumes that the PDFs within the CV are constantly distributed. An upgraded assumption is the second-order mapping that assumes a linear distribution within the CV.

There are assumably many ways to perform the second-order mapping. Here, considering the characteristics of a CC mesh and the methods already developed for the platform, a unique approach with the help of the nodal values is proposed. The approach is named nodal interpolation mapping (NIM), which is shown in Fig. 55.

For any triangular CV with the centroid P and three nodes N_1, N_2 and N_3, the CV can be geometrically divided into three zones I, II and III by connecting the centroid with each of the nodes, as shown in Fig. 55. Given that the PDF is linearly changing in the configuration space, the general formula for the PDF at any location (x, y) can be expressed as:

$$f_i = a \cdot x + b \cdot y + c \tag{8.1}$$

where a, b and c are the coefficients to be determined. For each zone in Fig. 55, the three coefficients can be calculated by solving a linear system. Taking zone I for example, we have:

$$\boldsymbol{C}_{\mathrm{I}} = \begin{bmatrix} a_{\mathrm{I}} \\ b_{\mathrm{I}} \\ c_{\mathrm{I}} \end{bmatrix} = \boldsymbol{\Lambda}_{\mathrm{I}}^{-1} \boldsymbol{F}_{\mathrm{I}} = \begin{bmatrix} x_P & y_P & 1 \\ x_{N_1} & y_{N_1} & 1 \\ x_{N_3} & x_{N_3} & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} f_i\left(P\right) \\ f_i\left(N_1\right) \\ f_i\left(N_3\right) \end{bmatrix} \tag{8.2}$$

In order to develop the coefficient vectors of the other two zones, $\boldsymbol{C}_{\mathrm{II}}$ and $\boldsymbol{C}_{\mathrm{III}}$, we just need to construct the corresponding $\boldsymbol{\Lambda}_{\mathrm{II}}$ and $\boldsymbol{F}_{\mathrm{II}}$, as well as $\boldsymbol{\Lambda}_{\mathrm{III}}$ and $\boldsymbol{F}_{\mathrm{III}}$, to calculate

$\boldsymbol{\Lambda}_{\mathrm{II}}^{-1}\boldsymbol{F}_{\mathrm{II}}$ and $\boldsymbol{\Lambda}_{\mathrm{III}}^{-1}\boldsymbol{F}_{\mathrm{III}}$. However, among the four points P and N_1, N_2, and N_3, only P is the data point, and therefore, $f_i(N_1)$, $f_i(N_2)$, and $f_i(N_3)$ need to be calculated. Then, the only thing required to complete the second-order mapping is to evaluate the PDFs at the nodes. If the node happens to be a boundary node, then no evaluation is needed since the boundary treatment will generate the PDFs at the boundary nodes; if the node is an interior node, then the interpolation is needed. However, the extrapolation scheme Eq. (6.8) for a boundary node in Sec. 6.2.2 can be easily applied for the interpolation of interior nodes by closing the 2π direction with the star cells. Therefore, the same equation can be used for both interpolation and extrapolation of the PDFs at the nodes. Nevertheless, extra effort is required to find all of the star cells for the interior nodes in order to use Eq. (6.8). It is worth noting that the NIM can map any variable, and is not limited to the PDFs. It can be imagined that the NIM will demand non-negligible computational overhead, since the mapping has to be constructed three times for each cell (due to three zones) for each iteration. Therefore, if the mesh has total X triangular cells, the inversion of the 3×3 matrix has to be performed $3X$ times. Therefore, a quantitative measurement of the computational cost needs to be carried out once the scheme is available.

8.2.2 Reduction of the collisional stiffness effect

When solving the DBE, the size of the time step Δt and relaxation time λ have to be of the same order, which is due to the physical restriction in Eq. (3.32). In addition, Δt is also restricted by the general spatial step size Δx, which is from the CFL restriction. Therefore, the sizes of Δx and λ also have to be comparable or of the same magnitude due to the indirect link connected by Δt, as shown in Fig. 56. Consequently, when the collision term becomes stiff, the mesh resolution has to be increased to counteract the stiffness. In addition, all three parameters are linked together and the decreasing of any one will result in the decreasing of the others in order to have a stable and accurate simulation. Such an increasing of the mesh resolution caused by the decreasing of relaxation time is called the collisional stiffness effect, which should appear in any DBE simulation regardless of the type of numerical methods applied (FV or FE), and will hurt the computational efficiency dramatically if an accurate

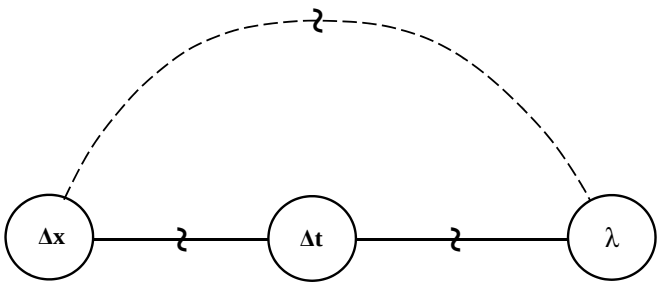


Figure 56: The size-wise relations among relaxation time, spatial step size and temporal step size

result for high-Re flows must be achieved. For example, if λ is cut by a factor of two, both Δx and Δt have to be decreased by a factor of two. As a result, the total computation time for the same steady-state solution increases four times.

The link between Δx and λ, or stiffness, can be validated through the numerical observations. It has been observed in the cavity flow in Sec. 6.4.3 that the change of λ will affect the error. Considering that such a change of λ is caused by the application of different lattice models, another series of numerical tests that use the same lattice model (D2Q9) with only λ changing has been performed. Couette flow is chosen for the tests, since the flow is laminar, and the analytical solution is not affected by the change of viscosity. The IRT mesh with fixed resolution is used for all of the tests. The Δx of the IRT mesh for the tests is 0.0786. The relaxation time λ is the only changing parameter in the tests, which has four values: 0.0003, 0.003, 0.03, and 0.3. The solid line in Fig. 57 shows the L_2 errors of the steady-state solution versus different relaxation times on a log-log scale. It can be seen that the line appears as a V-shape, which is the best evidence that there is a one-on-one relation between the relaxation time and mesh resolution. The relaxation time that generates the minimum error is 0.03, which has the same order of magnitude as Δx, 0.0786.

Next, the natural question is if it is possible to alleviate or even eliminate the coupling between relaxation time and mesh resolution in order to reduce the effect of collisional stiffness. The tentative answer to this question is yes, and the promising solution is the fractional-step (FS) method, or splitting method, which purposely solves the hyperbolic

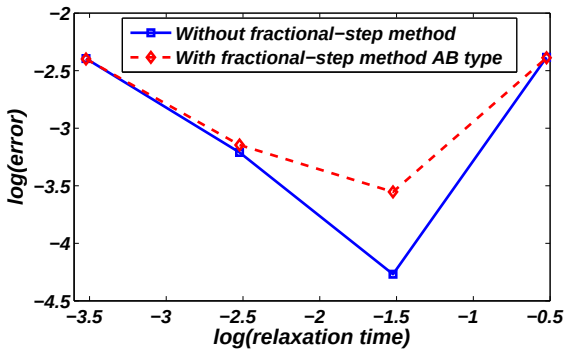


Figure 57: The effect of relaxation time on errors

equation with a stiff source term. With the FS method, the original equation is replaced by two equations. Taking the DBE, Eq. (3.1), for example, the FS method executes the following two steps one after another:

$$\begin{aligned} Step\,A: &\quad \tfrac{\partial f_i}{\partial t} = \tfrac{1}{\lambda}\left(f^{eq} - f_i\right) \\ Step\,B: &\quad \tfrac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \boldsymbol{\nabla} f_i = 0 \end{aligned} \tag{8.3}$$

In step A, the pure collision equation is solved, while in step B only the pure advection equation is solved. The FS method can have different variations. For example, one can choose to solve step B first, then step A. Therefore, it is called the BA type, which can be distinguished from the AB type, Eq. (8.3). There are also types of $A^{1/2}BA^{1/2}$ and $B^{1/2}AB^{1/2}$, in which the superscript *1/2* indicates that the corresponding step is solved on $\Delta t/2$. By splitting the original DBE into two steps, the link between Δt and λ can be loosened, and consequently the link between Δx and λ as well. The dashed line in Fig. 57 shows the results of an AB type on the same model setup. It can be seen that the V-shape profile becomes flattened, which is a good indicator that the link between the relaxation time and spacial mesh resolution can be loosened, even though the result is still not very satisfactory. However, it is not clear how other types of FS methods perform, and whether there are better methods other than the FS method. This is another area for future work.

8.2.3 Other areas for future work

In order to the help the current model become a practical tool for real engineering problems, there are also other areas that should be addressed in future work.

First, most engineering problems are 3D; and for some problems, there only exist 3D solutions, such as the Taylor-Couette flow, a free-surface flow between two concentric rotating cylinders. Therefore, it is desirable to extend the current model into the 3D domain. On the mesh side, a tetrahedral mesh is a good choice for 3D for two reasons. First, as with the triangular mesh in 2D, the tetrahedral mesh is a one-for-all solution for 3D problems due to its good adaptability to different complex 3D geometries. Second, each tetrahedral element has four faces, each one of which is a triangle. So, it is the 3D successor to the triangular mesh. Therefore, it allows minimum modifications to the solver compared to using other types of meshes. On the solver side, the challenge is the construction of the stencil for flux calculation in the 3D domain, while the time marching and collision calculator can be kept the same. However, since a tetrahedral mesh is utilized, the methodology to construct a 3D stencil is similar to that for the 2D stencil in the current work. On the boundary treatment side, the only modification is the extension of the extrapolation scheme (Eq. (6.8)) from 2D to 3D.

Another area requiring additional work is the computational aspect. Nowadays, high-performance computing (HPC) is becoming an industrial standard for almost all commercial packages. HPC is built on parallel computing that can execute the code simultaneously on a large cluster of CPUs and memory allocations, which can speed up the computation hundreds of times based on the number of CPUs. In order to do this for the current model, the code of the solver has to be rewritten. Depending on the types of memory architecture, different coding models can be selected; for example, OpenMP for shared memory and massage passing interface (MPI) for distributed memory. Another factor to consider in parallelism is how to evenly distribute the load to each CPU. Therefore, a new component, which can be called the divider or distributor, should be written to divide the computational domain prior to activating the solver. However, since an unstructured mesh is utilized, the dividing algorithm should be carefully designed to prevent slow communication among different sub-domains.

BIBLIOGRAPHY

[1] Takashi Abe. Derivation of the Lattice Boltzmann Method by Means of the Discrete Ordinate Method for the Boltzmann Equation. *Journal of Computational Physics*, 131:241–246, 1997.

[2] F. J. Alexander, S. Chen, and J. D. Sterling. Lattice Boltzmann thermohydrodynamics. *Physical Review E*, 47(4), 1993.

[3] Santosh Ansumali and Iliya V Karlin. Entropy Function Approach to the Lattice Boltzmann Method. *Journal of Statistical Physics*, 107(April):291–308, 2002.

[4] Adeline Augier, François Dubois, Benjamin Graille, and Pierre Lallemand. On rotational invariance of lattice Boltzmann schemes. *Computers and Mathematics with Applications*, 67(2):239–255, 2014.

[5] A Bardow, I V Karlin, and A A Gusev. General characteristic-based algorithm for off-lattice Boltzmann simulations. *Europhysics Letters (EPL)*, 75(3):434–440, aug 2006.

[6] O. Behrend. Solid-fluid boundaries in particle suspension simulations via the lattice Boltzmann method. *Physical Review E*, 52(1):1164–1175, 1995.

[7] R. Benzi, S. Succi, and M. Vergassola. The lattice Boltzmann equation: theory and applications. *Physics Reports*, 3:145–197, 1992.

[8] 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. *Physical Review*, 94(3):511–525, 1954.

[9] Nianzheng Cao, Shiyi Chen, Shi Jin, and Daniel Martínez. Physical symmetry and lattice symmetry in the lattice Boltzmann method. *Physical Review E*, 55(1):R21–R24, jan 1997.

[10] Feng Chen, Aiguo Xu, Guangcai Zhang, Yingjun Li, and Sauro Succi. Multiple-relaxation-time lattice Boltzmann approach to compressible flows with flexible specific-heat ratio and Prandtl number. *EPL (Europhysics Letters)*, 90(5):54003, jun 2010.

[11] Hudong Chen. Volumetric formulation of the lattice Boltzmann method for fluid dynamics: Basic concept. *Physical Review E*, 58(3):3955–3963, 1998.

[12] Leitao Chen and Laura Schaefer. A unified and preserved Dirichlet boundary treatment for the cell-centered finite volume discrete Boltzmann method. *Physics of Fluids*, 27:027104, 2015.

[13] Shiyi Chen and Gary D Doolen. Lattice Boltzmann method for Fluid Flows. *Annual Review of Fluid Mechanics*, 30:329–364, 1998.

[14] Shiyi Chen, Daniel Martínez, and Renwei Mei. On boundary conditions in lattice Boltzmann methods. *Physics of Fluids*, 8(9):2527–2536, sep 1996.

[15] Y. Chen, H. Ohashi, and M. Akiyama. Thermal lattice Bhatnagar-Gross-Krook model without nonlinear deviations in macrodynamic equations. *Physical Review E*, 50(4):2776–2783, 1994.

[16] Y T Chew, C Shu, and Y Peng. On Implementation of Boundary Conditions in the Application of Finite Volume Lattice Boltzmann Method. *Journal of Statistical Physics*, 107(April):539–556, 2002.

[17] Seok-Ki Choi and Ching-Long Lin. A Simple Finite-Volume Formulation of the Lattice Boltzmann Method for Laminar and Turbulent Flows. *Numerical Heat Transfer, Part B: Fundamentals*, 58(4):242–261, sep 2010.

[18] B. Chun and a. Ladd. Interpolated boundary condition for lattice Boltzmann simulations of flows in narrow gaps. *Physical Review E*, 75(6):066705, jun 2007.

[19] Phillip Colella. A direct Eulerian MUSCL scheme for gas dynamics. *SIAM Journal on Scientific and Statistical Computing*, 6(1):104–117, 1985.

[20] Phillip Colella and Pr Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of computational physics*, 54(1):174–201, 1984.

[21] M. Coutanceau and R. Bouard. Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation: part 2: unsteady flow. *Journal of Fluid Mechanics*, (79):257, 1977.

[22] D. Calhoun. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *Journal of Computational Physics*, 176:231–275, 2002.

[23] P J Davis and P Rabinowitz. *Methods of numerical integration*, volume 17. 1975.

[24] A. I. Delis, I. K. Nikolos, and M. Kazolea. Performance and Comparison of Cell-Centered and Node-Centered Unstructured Finite Volume Discretizations for Shallow Water Free Surface Flows. *Archives of Computational Methods in Engineering*, 18(1):57–118, feb 2011.

[25] S. C. R. Dennis and Gau-Zu Chang. Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100, 1970.

[26] Irina Ginzburg D'Humieres, D. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, pages 437–451, 2002.

[27] Boris Diskin, James L. Thomas, Eric J. Nielsen, Hiroaki Nishikawa, and Jeffery a. White. Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Viscous Fluxes. *AIAA Journal*, 48(7):1326–1338, jul 2010.

[28] Gary D. Doolen. Lattice Gas Methods: Theory, Applications, and Hardware. *Physica D*, (47):1–339, 1991.

[29] Herbert Edelsbrunner. *Geometry and topology for mesh generation*. Cambridge University Press, New York, 2001.

[30] J.G.M. Eggels and J.a. Somers. Numerical simulation of free convective flow using the lattice-Boltzmann scheme. *International Journal of Heat and Fluid Flow*, 16(5):357–364, 1995.

[31] Olga Filippova and Dieter Hänel. Grid Refinement for Lattice-BGK Models. *Journal of Computational Physics*, 147(1):219–228, nov 1998.

[32] Bengt Fornberg. A numerical study of steady viscous flow past a circular cylinder. *Journal of Fluid Mechanics*, 98:819, 1980.

[33] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-Gas Automata for the Navier-Stokes Equation. *Physical Review Letters*, 56(14):1505–1508, 1986.

[34] Uriel Frisch, Dominique D'Humières, Brosl Hasslacher, Pierre Lallemand, Yves Pomeau, and Jean-Pierre Rivet. Lattice Gas Hydrodynamics in Two and Three Dimensions. *Complex Systems*, 1:649–707, 1987.

[35] J. Ghasemi and S.E. Razavi. On the finite-volume Lattice Boltzmann modeling of thermo-hydrodynamics. *Computers & Mathematics with Applications*, 60(5):1135–1144, sep 2010.

[36] C. T Ghia, U. Ghia, K. N. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, 48:387–411, 1982.

[37] I Ginzbourg and D D'Humieres. Local Second-Order Boundary Methods for Lattice Boitzmann Models. *Journal of Statistical Physics*, 84:927–971, 1996.

[38] I. Ginzburg and K. Steiner. Lattice Boltzmann model for free-surface flow and its application to filling process in casting. *Journal of Computational Physics*, 185:61–99, 2003.

[39] Zhaoli Guo and T. S Zhao. Explicit finite-difference lattice Boltzmann method for curvilinear coordinates. *Physical Review E*, 67(6):066709, jun 2003.

[40] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. An extrapolation method for boundary conditions in lattice Boltzmann method. *Physics of Fluids*, 14(6):2007, 2002.

[41] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice Boltzmann method. *Chinese Physics*, 11(4):366–374, 2002.

[42] Zhaoli Guo, Chuguang Zheng, Baochang Shi, and T. Zhao. Thermal lattice Boltzmann equation for low Mach number flows: Decoupling model. *Physical Review E*, 75(3):036704, mar 2007.

[43] X. He, S. Chen, and R. Zhang. A Lattice Boltzmann Scheme for Incompressible Multiphase Flow and Its Application in Simulation of Rayleigh-Taylor Instability. *Journal of Computational Physics*, 152(2):642–663, jul 1999.

[44] Xiaoyi He, Shiyi Chen, and Gary D Doolen. A Novel Thermal Model for the Lattice Boltzmann Method in Incompressible Limit. *Journal of Computational Physics*, 146(146):282–300, 1998.

[45] Xiaoyi He and Gary Doolen. Lattice Boltzmann Method on Curvilinear Coordinates System: Flow around a Circular Cylinder. *Journal of Computational Physics*, 134(2):306–315, jul 1997.

[46] Xiaoyi He and Gary D Doolen. Lattice Boltzmann method on a curvilinear coordinate system: Vortex shedding behind a circular cylinder. *Physical Review E*, 56(1):434–440, 1997.

[47] Xiaoyi He and Li-shi Luo. A Priori derivation of the lattice Boltzmann equation. *Physical Review E*, 55(6):R6333–R6336, 1997.

[48] Xiaoyi He and Li-Shi Luo. Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E*, 56(6):6811–6817, dec 1997.

[49] Xiaoyi He, Li-Shi Luo, and Micah Dembo. Some Progress in Lattice Boltzmann Method. Part I. Nonuniform Mesh Grids. *Journal of Computational Physics*, 129(2):357–363, dec 1996.

[50] Xiaoyi He, Qisu Zou, Li-shi Luo, and Micah Dembo. Analytic Solutions of Simple Flows and Analysis of Nonslip Boundary Conditions for the Lattice Boltzmann BGK Model. *Journal of Statistical Physics*, 87:115–136, 1997.

[51] T. Inamuro, S. Tajima, and F. Ogino. Lattice Boltzmann simulation of droplet collision dynamics. *International Journal of Heat and Mass Transfer*, 47:4649–4657, 2004.

[52] Takaji Inamuro, Masato Yoshino, and Fumimaru Ogino. A non-slip boundary condition for lattice Boltzmann simulations. *Physics of Fluids*, 7(12):2928, 1995.

[53] Salvador Izquierdo and Norberto Fueyo. Characteristic nonreflecting boundary conditions for open boundaries in lattice Boltzmann methods. *Physical Review E*, 78(4):046707, oct 2008.

[54] A. J. C. Ladd and R. Verberg. Lattice-Boltzmann simulations of particle-fluid suspensions. *Journal of Statistical Physics*, 104:1191–1251, 2001.

[55] Pierre Lallemand and Li-Shi Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. *Physical Review E*, 61(6):6546–6562, jun 2000.

[56] Taehun Lee and Ching-Long Lin. A Characteristic Galerkin Method for Discrete Boltzmann Equation. *Journal of Computational Physics*, 171(1):336–356, jul 2001.

[57] Taehun Lee and Ching Long Lin. An Eulerian description of the streaming process in the lattice Boltzmann equation. *Journal of Computational Physics*, 185(2):445–471, mar 2003.

[58] Bram Van Leer. Towards the ultimate conservative difference scheme. I. The quest of monotonicity. *Lecture Notes in Physics*, 18:163–168, 1973.

[59] Bram Van Leer. Towards the Ultimate Conservative Difference Scheme. II. Monotonicity and Conservation Combined. *Journal of Computational Physics*, 14:361–370, 1974.

[60] Bram Van Leer. Towards the Ultimate Conservative Difference Scheme III. Upstream-Centered Finite-Difference Schemes for Ideal Compressible Flow. *Journal of Computational Physics*, 23:263–275, 1977.

[61] Bram Van Leer. Towards the Ultimate Conservative Difference Scheme. IV. A New Approach to Numerical Convection. *Journal of Computational Physics*, 23:279–299, 1977.

[62] Bram Van Leer. Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequal to Godunov's Method. *Journal of Computational Physics*, 32:101–136, 1979.

[63] R.J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*, volume 54. 2002.

[64] Yuguo Li and L Baldacchino. Implementation of some higher-order convection schemes on non-uniform grids. *International Journal for Numerical Methods in Fluids*, 2:1201–1220, 1995.

[65] Yusong Li, Eugene LeBoeuf, and P. Basu. Least-squares finite-element lattice Boltzmann method. *Physical Review E*, 69(6):065701, jun 2004.

[66] Chih-Hao Liu, Kuen-Hau Lin, Hao-Chueh Mai, and Chao-An Lin. Thermal boundary conditions for thermal lattice Boltzmann simulations. *Computers & Mathematics with Applications*, 59(7):2178–2193, apr 2010.

[67] M. Coutanceau and R. Bouard. Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation: part 1: steady flow. *Journal of Fluid Mechanics*, (79):231, 1977.

[68] F. Massaioli, R. Benzi, and S. Succi. Exponential Tails in Two-Dimensional Rayleigh-Bénard Convection. *Europhysics Letters (EPL)*, 21(3):305–310, 1993.

[69] D. J. Mavriplis. Unstructured Grid Techniques. *Annual Review of Fluid Mechanics*, 29(1):473–514, jan 1997.

[70] Guy McNamara and Berni Alder. Analysis of the lattice Boltzmann treatment of hydrodynamics. *Physica A: Statistical Mechanics and its Applications*, 194(1-4):218–228, 1993.

[71] Renwei Mei, Li-Shi Luo, and Wei Shyy. An Accurate Curved Boundary Treatment in the Lattice Boltzmann Method. *Journal of Computational Physics*, 155(2):307–330, nov 1999.

[72] Renwei Mei and Wei Shyy. On the Finite Difference-Based Lattice Boltzmann Method in Curvilinear Coordinates. *Journal of Computational Physics*, 143(2):426–448, jul 1998.

[73] Ahmed Mezrhab, Mohammed Amine Moussaoui, Mohammed Jami, Hassan Naji, and M'hamed Bouzidi. Double MRT thermal lattice Boltzmann method for simulating convective flows. *Physics Letters A*, 374(34):3499–3507, jul 2010.

[74] W. Miller, S. Succi, and D. Mansutti. Lattice Boltzmann model for anisotropic liquid-solid phase transition. *Physical Review Letters*, 86(16):3578–3581, 2001.

[75] Francesca Nannelli and Sauro Succi. The lattice Boltzmann equation on irregular lattices. *Journal of Statistical Physics*, 68(3-4):401–407, aug 1992.

[76] F. Nieuwstadt and H. B. Keller. Viscous flow past circular cylinders. *Computers & Fluids*, 1:59–71, 1973.

[77] David R. Noble, Shiyi Chen, John G. Georgiadis, and Richard O. Buckius. A consistent hydrodynamic boundary condition for the lattice Boltzmann method. *Physics of Fluids*, 7(1):203, 1995.

[78] D. V. Patil and K. N. Lakshmisha. Two-dimensional flow past circular cylinders using finite volume lattice Boltzmann formulation. *International Journal for Numerical Methods in Fluids*, 69(August 2011):1149–1164, 2012.

[79] Dhiraj V. Patil and K. N. Lakshmisha. Finite volume TVD formulation of lattice Boltzmann simulation on unstructured mesh. *Journal of Computational Physics*, 228:5262–5279, aug 2009.

[80] D.V. Patil. Chapman-Enskog analysis for finite-volume formulation of lattice Boltzmann equation. *Physica A: Statistical Mechanics and its Applications*, 392(12):2701–2712, jun 2013.

[81] Gongwen Peng and Haowen Xi. On boundary conditions in the finite volume lattice Boltzmann method on unstructured meshes. *International Journal of Modern Physics C*, 10(6):1003–1016, 1999.

[82] Gongwen Peng, Haowen Xi, Comer Duncan, and So-Hsiang Chou. Lattice Boltzmann method on irregular meshes. *Physical Review E*, 58(4):R4124–R4127, oct 1998.

[83] Gongwen Peng, Haowen Xi, Comer Duncan, and So-Hsiang Chou. Finite volume scheme for the lattice Boltzmann method on unstructured meshes. *Physical Review E*, 59(4):4675–4682, apr 1999.

[84] Y. Peng, C. Shu, and Y. Chew. Simplified thermal lattice Boltzmann model for incompressible thermal flows. *Physical Review E*, 68(2):026701, aug 2003.

[85] P. L. Roe. Approximate Riemann Solvers , Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[86] N Rossi, S Ubertini, G Bella, and S Succi. Unstructured lattice Boltzmann method in three dimensions. *International Journal for Numerical Methods in Fluids*, 49(August 2004):619–633, 2005.

[87] S. K. Godunov. a finite-difference method for the numerical computation and discontinuous solutions of the equations of fluid dynamics. *Mathematics Of The USSR-Sbornik*, 47:271–306, 1959.

[88] G. Bella S. Succi and F. Papetti. Lattice Kinetic Theory for Numerical Combustion. *J. of Scientific Computing*, 12(4):395–408, 1997.

[89] X. Shan and H. Chen. Lattice Boltzmann model for simulating flows with multi phases and components. *Physical Review E*, 47(3):1815–1819, 1993.

[90] X. Shan and H. Chen. Simulation of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation. *Physical Review E*, 49(4):2941–2948, 1994.

[91] Xiaowen Shan. Simulation of Rayleigh-Bénard convection using a lattice Boltzmann method. *Physical Review E*, 55(3):2780–2788, 1997.

[92] Xiaowen Shan and Xiaoyi He. Discretization of the Velocity Space in the Solution of the Boltzmann Equation. *Physical Review Letters*, 80(1):65–68, jan 1998.

[93] Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. Kinetic theory representation of hydrodynamics: a way beyond the Navier-Stokes equation. *Journal of Fluid Mechanics*, 550(-1):413, feb 2006.

[94] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *Applied Computational Geometry: Towards Geometric Engineering*, 1148:203–222, 1996.

[95] Jonathan Richard Shewchuk. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, 2002.

[96] Yong Shi, T. Zhao, and Z. Guo. Thermal lattice Bhatnagar-Gross-Krook model for flows with viscous heat dissipation in the incompressible limit. *Physical Review E*, 70(6):066310, dec 2004.

[97] Victor Sofonea and Robert F. Sekerka. Viscosity of finite difference lattice Boltzmann models. *Journal of Computational Physics*, 184(2):422–434, jan 2003.

[98] Maik Stiebler, Jonas Tölke, and Manfred Krafczyk. An upwind discretization scheme for the finite volume lattice Boltzmann method. *Computers & Fluids*, 35(8-9):814–819, sep 2006.

[99] Sauro Succi, Giorgio Amati, and Roberto Benzi. Challenges in lattice Boltzmann computing. *Journal of Statistical Physics*, 81(1-2):5–16, oct 1995.

[100] S. Ubertini, G. Bella, and S. Succi. Lattice Boltzmann method on unstructured grids: Further developments. *Physical Review E*, 68(1):016701, jul 2003.

[101] S. Ubertini, G. Bella, and S. Succi. Unstructured lattice Boltzmann equation with memory. *Mathematics and Computers in Simulation*, 72(2-6):237–241, sep 2006.

[102] S Ubertini and S Succi. A generalised Lattice Boltzmann equation on unstructured grids. *Communications in Computational Physics*, 3(2):342–356, 2008.

[103] S Ubertini, S Succi, and G Bella. Lattice Boltzmann schemes without coordinates. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 362(1821):1763–71, aug 2004.

[104] G Vahala, P Pavlo, L Vahala, and N S Martys. Thermal lattice-Boltzmann models (TLBM) for compressible flows. *International Journal of Modern Physics C-Physics and Computer*, 9(8):1247–1262, 1998.

[105] R. G M Van Der Sman, M. H. Ernst, and A. C. Berkenbosch. Lattice Boltzmann scheme for cooling of packed cut flowers. *International Journal of Heat and Mass Transfer*, 43(4):577–587, 2000.

[106] Jia Wang, Donghai Wang, Pierre Lallemand, and Li-Shi Luo. Lattice Boltzmann simulations of thermal convective flows in two dimensions. *Computers & Mathematics with Applications*, 65(2):262–286, jan 2013.

[107] Kent E. Wardle and Taehun Lee. Finite element lattice Boltzmann simulations of free surface flow in a concentric cylinder. *Computers & Mathematics with Applications*, 65(2):230–238, jan 2013.

[108] Stephen Wolfram. Cellular automaton fluids 1: Basic theory. *Journal of Statistical Physics*, 45:471–526, 1986.

[109] H Xi, G Peng, and S H Chou. Finite-volume lattice Boltzmann method. *Physical Review E*, 59(5):6202–6205, may 1999.

[110] Haowen Xi, Gongwen Peng, and So-Hsiang Chou. Finite-volume lattice Boltzmann schemes in two and three dimensions. *Physical Review Eeview E*, 60(3):3380–3388, sep 1999.

[111] T. Ye, R. Mittal, H.S. Udaykumar, and W. Shyy. An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries. *Journal of Computational Physics*, 156:209–240, 1999.

[112] Dazhi Yu, Renwei Mei, Li-Shi Luo, and Wei Shyy. Viscous flow computations with the method of lattice Boltzmann equation. *Progress in Aerospace Sciences*, 39(5):329–367, jul 2003.

[113] Dazhi Yu, Renwei Mei, and Wei Shyy. A multi-block lattice Boltzmann method for viscous fluid ows. *International Journal for Numerical Methods in Fluids*, 39(October 2001):99–120, 2002.

[114] Peng Yuan and Laura Schaefer. A Thermal Lattice Boltzmann Two-Phase Flow Model and Its Application to Heat Transfer Problems-Part 1. Theoretical Foundation. *Journal of Fluids Engineering*, 128(1):142, 2006.

[115] A Zarghami, M J Maghrebi, J Ghasemi, and S Ubertini. Lattice Boltzmann Finite Volume Formulation with Improved Stability. *Communications in Computational Physics*, 12(1):42–64, 2012.

[116] A. Zarghami, S. Ubertini, and S. Succi. Finite-volume lattice Boltzmann modeling of thermal transport in nanofluids. *Computers & Fluids*, 77:56–65, apr 2013.

[117] Ahad Zarghami, Chiara Biscarini, Sauro Succi, and Stefano Ubertini. Hydrodynamics in Porous Media: A Finite Volume Lattice Boltzmann Study. *Journal of Scientific Computing*, 59(1):80–103, 2013.

[118] Donald P. Ziegler. Boundary conditions for lattice Boltzmann simulations. *Journal of Statistical Physics*, 71(5-6):1171–1177, jun 1993.

[119] Qisu Zou and Xiaoyi He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591, 1997.