

Fluid Simulation with the Lattice Boltzmann Method

WEI LI, Shanghai Jiao Tong University, China

CHAOYANG LYU, Shanghai Artificial Intelligence Lab, China

MENGYUN LIU, ShanghaiTech University, China

YIXIN CHEN, University of Toronto, Canada

MATHIEU DESBRUN, Inria – Ecole polytechnique, France

KUI WU, Lightspeed, USA

XIAOPEI LIU, ShanghaiTech University, China



Macroscopic fluid flow solvers based on the incompressible Navier–Stokes equations are widely used in graphics, but achieving high visual fidelity in complex scenarios—such as free-surface motion, high–Reynolds-number flows, multiphase flow effects, and interactions with moving boundaries—often requires significant additional modeling and engineering effort. The kinetic *Lattice Boltzmann Method* (LBM) has recently emerged as an attractive alternative for fluid flow simulation in this setting. Rather than directly discretizing macroscopic flow equations, LBM evolves mesoscopic particle distribution functions on a regular Cartesian lattice through simple, local collision and streaming operations. This local computational structure provides excellent data locality, maps naturally to modern GPU architectures, and enables stable, efficient time integration in practice. In these course notes, we introduce the fundamentals of the Lattice Boltzmann Method along with various recent advances that have fostered its use across a range of applications relevant to graphics and simulation, including single- and multiphase flows, fluid–solid interaction, and learning-based control for aerial and underwater robotics. We further discuss how LBM is used beyond academic research, highlighting its adoption in industrial simulation tools and game production pipelines, as well as high-fidelity visual effects.



This work is licensed under a Creative Commons Attribution 4.0 International License.

Foreword

Over the past decade, the Lattice Boltzmann Method (LBM) has matured into a leading framework for high-performance fluid simulation across research and industry alike. Its appeal rests on a set of properties that align unusually well with the demands of modern computing: collision and streaming operators that are strictly local, mapping naturally onto massively parallel architectures; an intrinsic ability to handle complex geometries through simple boundary rules; and a level of physical fidelity that has proven sufficient across a widening range of applications — from real-time simulation and visual effects to large-scale industrial tools such as virtual wind tunnels. The growing body of LBM-based work at SIGGRAPH and SIGGRAPH Asia reflects this trajectory directly: the method has moved from occasional appearance to recurring presence as a core computational framework.

Despite this prominence, LBM has never been the subject of a dedicated SIGGRAPH course. To date, it has only been mentioned peripherally, most notably as a small component of the Real-Time Physics course at SIGGRAPH 2004. As a result, there remains a significant gap between the method's practical importance and its formal coverage within the SIGGRAPH educational program. This lack of structured exposure has made it difficult for practitioners and researchers to fully understand the method's foundations, strengths, limitations, and best practices for deployment on contemporary hardware.

These course notes are an attempt to close that gap. Written by authors with direct experience across multiple LBM publications at SIGGRAPH over the past six years, they aim to provide something the existing literature rarely offers in a single place: a coherent path from first principles to production-oriented practice. The intent is not to survey recent work, but to equip attendees with the concepts, implementation details, and the critical perspective needed to start working with LBM solvers.

About the Lecturers

The authors of these course notes combine leading senior researchers, one of whom has organized six highly successful SIGGRAPH courses, with emerging researchers from academia and industry who are actively advancing the state of the art. We hope this balance ensures both depth of experience and relevance to current research and production challenges. The authors are:

Wei Li

Shanghai Jiao Tong University

1104720604wei@gmail.com

Wei Li is a tenure-track Associate Professor at the John Hopcroft Center for Computer Science, School of Computer Science and Engineering, Shanghai Jiao Tong University. Before joining SJTU, he served as a Senior Research Scientist at Tencent's Lightspeed Studio (Shanghai). Before that, he conducted postdoctoral research at Inria Saclay (France) after earning his Ph.D. from the School of Information Science and Technology at ShanghaiTech University. Wei Li's research focuses on computer graphics, physical simulation, and physics-based deep learning. His specific research areas include high-performance Lattice Boltzmann Methods (LBM), modeling and simulation of complex physical phenomena, visualization, and high-performance physics simulation engines.

Chaoyang Lyu

Shanghai Artificial Intelligence Lab

lyuchaoyang@outlook.com

Chaoyang Lyu is a Postdoctoral Researcher at Shanghai Artificial Intelligence Lab. His research interests are physics-based simulation, computer graphics, and embodied artificial intelligence. He received his Ph.D. in Computer Science from ShanghaiTech University in 2024, working on high-performance mesoscopic methods for fluid simulation with complex fluid-solid coupling.

Mengyun Liu

ShanghaiTech University

liumy.melia@gmail.com

Mengyun Liu is a Ph.D. candidate in Computer Science at ShanghaiTech University, where she also earned her B.E. degree in 2022. Advised by Professor Xiaopei Liu, her research focuses on high-performance and high-precision fluid dynamics simulations using mesoscopic formulations, with applications in industrial design, robotics, and computer animation.

Yixin Chen

University of Toronto

yixinc.chen@mail.utoronto.ca

Yixin Chen is a PhD candidate in Computer Science at the University of Toronto, advised by Professor David I.W. Levin. Her research focuses on high-performance and differentiable fluid simulation and control, combining insights from numerical methods, physics, and computer graphics. Prior to her PhD, Yixin received a B.Eng. in Computer Science and

Technology from ShanghaiTech University.

Mathieu Desbrun

Inria / Ecole polytechnique

mathieu.desbrun@inria.fr

Mathieu Desbrun is an advanced researcher at Inria Saclay and a Professor at Ecole polytechnique, where he focuses on geometry-driven numerics, covering data analysis, machine learning, and simulation. Earlier, he was a faculty member at Caltech (USA) in the Computing and Mathematical Sciences department for over twenty years. He was named an ACM Fellow in 2020, and became a SIGGRAPH Academy member in 2021. He has organized a number of courses at SIGGRAPH over the years, including *Discrete Differential Geometry*, *Geometry Processing with Discrete Exterior Calculus*, and *Vector Field Processing on Triangle Meshes*.

Kui Wu

LIGHTSPEED

kwwu@lightspeed-studios.com

Kui Wu is a principal research scientist at LIGHTSPEED studios. Previously, he was a postdoctoral associate in the Computational Design and Fabrication Group of Prof. Wojciech Matusik at MIT CSAIL. He received his PhD degree in Computer Science from the University of Utah in 2019. His research interests are computer graphics, including modeling, rendering, simulation, and fabrication.

Xiaopei Liu

ShanghaiTech University

liuxp@shanghaitech.edu.cn

Xiaopei Liu is a tenured Associate Professor at the School of Information Science and Technology (SIST) at ShanghaiTech University. He obtained his Ph.D. in Computer Science and Engineering from the Chinese University of Hong Kong (CUHK) and conducted postdoctoral research at Nanyang Technological University (NTU), Singapore. His work focuses on high-performance simulation and visualization for classical and quantum fluids using mesoscopic approaches, and extending to fluid-interactive robotic simulation, control, and learning. These contributions have been applied to the verification and optimization of aircraft and automotive design, sim-to-real robotic systems, medical diagnosis, visual effects, and fundamental scientific research.

Targeted audience, objectives, and syllabus

These course notes are intended for academic researchers and industry professionals interested in the Lattice Boltzmann Method (LBM) for fluid simulation and its recent advances, covering its foundations and practical use in single-phase, two-phase, and multiphase flows, as well as fluid–solid coupling for complex interacting boundaries. This course assumes basic knowledge of fluid dynamics and requires familiarity with multivariable calculus, linear algebra, and common numerical algorithms. No prior experience with LBM is required. Some background in statistical mechanics and kinetic theory could be useful, and prior knowledge of GPU parallel computing is a plus. Attendees will gain an overall understanding of recent advances in the Lattice Boltzmann Method (LBM) for fluid simulation. They will learn the methodological evolution and implementation essentials of LBM for single-phase, two-phase, and multiphase flows, and understand modern techniques for fluid–solid coupling with complex, interacting boundaries. Participants will

also be provided with reference source code and executable examples, enabling them to explore, adapt, and reuse the material as a foundation for their own simulation projects. The structure of these course notes is as follows:

1. **Introduction and Welcome** (*All speakers*) (5 min)
2. **Kinetic Theory** (*Mathieu Desbrun*) (15 min)
3. **Collision Model, Boundary Condition and Grid Refinement** (*Chaoyang Lyu*) (25 min)
4. **Moment Representation & Multiphase LBM** (*Wei Li*) (20 min)
5. **GPU Optimization** (*Yixin Chen*) (10 min)
6. **Applications** (*Kui Wu*) (10 min)
7. **Conclusion and Q&A** (*All speakers*) (5 min)

1 Introduction

Fluid dynamics is not merely visually mesmerizing for its intricate, convoluted motions; it is also one of the most intellectually rich and practically important fields in physics and engineering. It forms the foundation of a wide range of scientific and industrial applications involving fluid motion. Its inherent difficulty arises from its *multiscale behavior*: a single flow problem may simultaneously involve phenomena spanning many orders of magnitude in space and time, from nanometer-scale interactions between droplets and solid surfaces to the meter-scale design and operation of cooling systems in power-generation facilities.

For viscous flows, the governing equations are a set of nonlinear partial differential equations known as the *Navier–Stokes equations*, which conserve mass and momentum. Although these equations have been studied for well over a century, closed-form analytical solutions exist only in idealized settings. Even the fundamental question of whether smooth solutions exist for all time in three dimensions remains unresolved and is recognized as one of the Clay Mathematics Institute’s Millennium Prize Problems. For general flows, exact solutions are simply not available.

To simulate fluids in practice and integrate these models into real-world industrial pipelines and entertainment production, a wide range of numerical methods has been developed across computational fluid dynamics (CFD) and computer graphics (CG). In these course notes, we primarily adopt the CG perspective and present a focused review of representative kinetic simulation techniques that have proven effective in visual effects and interactive applications.

From macroscopic simulation... To date, fluid simulation techniques primarily rely on the Euler and Navier–Stokes equations to model the evolution of *macroscopic variables* over time. This framework underpins many widely used algorithms [11], including Eulerian methods [21, 87], Fluid Implicit Particle (FLIP) [33, 101], Smoothed Particle Hydrodynamics (SPH) [2, 22], and the Material Point Method (MPM) [51, 56]. These approaches are popular in both research and production due to their ability to reproduce a broad range of complex fluid behaviors with visually convincing results, particularly for viscous flows. However, these methods face significant limitations that can make highly scalable solvers more challenging. In particular, divergence-free projections and the inherent numerical dissipation of these schemes tend to artificially damp high-frequency flow structures and turbulent fluctuations, reducing visual richness and suppressing physically meaningful fine-scale dynamics. As a result, visually capturing detailed turbulence remains a persistent challenge.

... To kinetic simulation. Against this backdrop, a growing body of works [63, 64, 67, 68, 92] has recently highlighted the Lattice Boltzmann Method (LBM) as a compelling alternative to traditional macroscopic discretization approaches. Rather than solving the Euler or Navier–Stokes equations directly, *LBM evolves a mesoscopic particle distribution function governed by a kinetic equation*, from which macroscopic flow quantities are recovered as low-order moments. Owing to

Symbol	Meaning
ρ	fluid density
T	fluid temperature
$\mathbf{u}$	fluid velocity
$\mathbf{F}$	external force
p	fluid pressure
$\boldsymbol{\sigma}$	fluid viscous stress tensor
R	gas constant
$\dot{\mathbf{S}}$	fluid symmetric rate-of-strain tensor
f	continuous particle distribution function
ν	kinematic shear viscosity
μ	dynamic viscosity
f^{eq}	particle equilibrium distribution function
f^{neq}	particle non-equilibrium distribution function
$\mathbf{c}$	mesoscopic velocity
Ω	collision operator
Π	momentum flux tensor
Δt	lattice time step
f_i	discrete distribution function in lattice direction i
f_i^*	discrete post-collision distribution function in lattice direction i
f_i^{eq}	discrete equilibrium distribution function in lattice direction i
$\mathbf{c}_i$	discrete lattice velocity in direction i
w_i	lattice quadrature weight in direction i
c_s	lattice speed of sound
τ	relaxation time
$\mathbf{H}^{(n)}$	n -th order Hermite polynomial tensor

its regular lattice structure and strictly local collision and streaming operations, LBM exhibits excellent data locality and a highly structured computational pattern, making it particularly well suited for massively parallel architectures. In addition, LBM often demonstrates reduced numerical dissipation across a wide range of flow regimes, thus better capturing fine-scale features in turbulence-dominated problems [64, 74]. Beyond computational efficiency and numerical fidelity, LBM readily accommodates complex and moving boundaries through flexible boundary treatments [71, 72, 75], and it extends naturally to two-phase [44, 65–67, 98] as well as more general n -phase flow simulations [25, 70] within a unified modeling framework.

Despite its growing relevance to graphics, robotics, and learning-based simulation, there is currently no comprehensive tutorial for these audiences that balances theoretical foundations with the practical considerations necessary for implementation. This course is designed to address that gap. The remainder of these notes is thus organized into the following sections:

- (1) In Chapter 2, we introduce the kinetic-theory perspective underlying LBM and establish the connection between mesoscopic distribution functions and macroscopic flow quantities.
- (2) In Chapter 3, we review collision modeling, evaluating common relaxation operators and their impact on numerical stability and accuracy.

- (3) Chapter 4 focuses on boundary conditions, with an emphasis on robust and accurate treatments of complex geometries.
- (4) Chapter 5 then discusses grid-refinement techniques for LBM to use a fine lattice only where accuracy is required, while employing a coarser lattice elsewhere to reduce computational cost.
- (5) We then present the high-order moment (HOME) representation in Chapter 6, showing how a compact set of three velocity moments can be used in place of the full distribution function while preserving simulation accuracy.
- (6) In Chapter 7, we turn to GPU-oriented optimization strategies, highlighting implementation choices that maximize throughput on modern parallel hardware.
- (7) Chapter 8 surveys representative applications, including single-phase and multiphase flows as well as fluid–solid coupling scenarios relevant to industrial and graphics pipelines.
- (8) Finally, Chapter 9 concludes with a summary of key takeaways and an outlook on promising future directions.

2 Kinetic Theory

In this chapter, we introduce a kinetic-theory perspective on fluid dynamics, following the view that hydrodynamic descriptions can be systematically derived from a continuum Boltzmann equation [86]. We begin with a focused review of the continuum Boltzmann (BGK) equation, illustrating how macroscopic variables — specifically density, momentum, and energy — are derived as velocity moments of the distribution function. We then highlight the role of the collision operator in driving relaxation toward equilibrium and use its key properties to derive the Maxwell–Boltzmann equilibrium distribution, together with a brief discussion of Boltzmann’s H-theorem. Finally, we apply a Hermite expansion and Gauss-Hermite quadrature to discretize the continuum Boltzmann equation in velocity space, yielding a discrete-velocity lattice Boltzmann method.

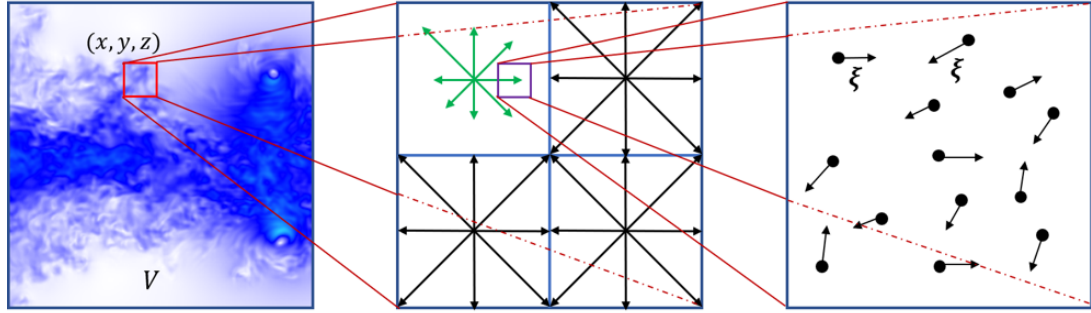


Fig. 1. **Descriptions of fluid motion across varying scales:** From left to right: macroscopic, mesoscopic, and microscopic scales. The macroscopic approach utilizes a control volume (V) to average the physical effects of numerous particles. The mesoscopic approach, relying on a localized control volume around $x = (x, y, z)$, uses the Boltzmann equation to trace mesoscopic particles (a statistical representation of microscopic particles), equivalently recovering macroscopic dynamics. The microscopic approach utilizes molecular dynamics to describe fluid behavior by tracking individual particle motions.

2.1 Navier-Stokes equation

Compressible Navier-Stokes Framework. The dynamics of isothermal fluid flows are mathematically described by the standard compressible Navier-Stokes (N-S) equations. The governing laws for the conservation of mass and momentum

are expressed as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \quad \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot \boldsymbol{\sigma} + \mathbf{F}. \quad (1)$$

In this formulation, t denotes time, while ρ , $\mathbf{u}$, and p represent the macroscopic density, velocity vector, and pressure fields within the fluid domain, respectively. The symbol $\otimes$ denotes the tensor dyadic product, and $\mathbf{F}$ accounts for any volumetric body forces, e.g., $\mathbf{F} = \rho \mathbf{g}$ for gravity. To achieve thermodynamic closure, an Equation of State (EoS) is introduced to link the pressure to the density at a prescribed isothermal temperature T , formulated generally as $p = p(\rho, T)$. For instance, assuming an ideal gas yields the relation $p = \rho RT$, with R acting as the specific gas constant. Within the momentum equation, the viscous stress tensor is modeled as

$$\boldsymbol{\sigma} = 2\rho\nu \left[\mathbf{S} - \frac{1}{D} (\nabla \cdot \mathbf{u}) \mathbf{I} \right] + \rho\nu_b (\nabla \cdot \mathbf{u}) \mathbf{I}, \quad (2)$$

where $\mathbf{I}$ denotes the second-order identity tensor, D is the spatial dimension, and ν and ν_b denote the kinematic shear and bulk viscosities, respectively. The symmetric rate-of-strain tensor is defined as

$$\mathbf{S} = \frac{1}{2} [\nabla \mathbf{u} + (\nabla \mathbf{u})^T]. \quad (3)$$

Under Stokes' hypothesis, the bulk viscosity is assumed to vanish, i.e., $\nu_b = 0$. Consequently, in three spatial dimensions, the viscous stress tensor reduces to

$$\hat{\mathbf{S}} = 2\rho\nu \hat{\mathbf{S}} - \frac{2}{3} \rho\nu (\nabla \cdot \mathbf{u}) \mathbf{I}. \quad (4)$$

where $\hat{\mathbf{S}}$ characterizes the symmetric rate-of-strain tensor, which is evaluated via:

$$\hat{\mathbf{S}} = \frac{1}{2} (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \quad (5)$$

Numerically integrating the compressible N-S equations presents numerous computational bottlenecks. The primary challenge stems from the finite propagation speed of acoustic waves inherent in compressible media. Resolving these high-frequency acoustic phenomena using explicit schemes imposes a severe restriction on the time-step size to ensure stability. While implicit time-marching strategies can circumvent this constraint and allow for larger time increments, they inevitably introduce the computationally expensive burden of assembling and inverting massive, globally coupled linear systems at every discrete temporal step.

Incompressible Navier-Stokes Formulation. A common simplification assumes a spatially and temporally uniform fluid density, ρ , bypassing the severe timestep constraints imposed by acoustic waves in the compressible regime. This idealized *incompressibility* postulate effectively presumes an infinite speed of sound, thereby reducing the governing dynamics to the standard incompressible Navier-Stokes (N-S) framework:

$$\nabla \cdot \mathbf{u} = 0, \quad \rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \rho\nu \nabla^2 \mathbf{u} + \mathbf{F}. \quad (6)$$

This incompressible formulation is the cornerstone of fluid dynamics simulation within visual computing and physically based animation. Considerable research has focused on accelerating the associated numerical solvers as we reviewed in Section 1. A fundamental computational phase in these implicit algorithms involves determining the scalar pressure field, p , by resolving the pressure Poisson equation (PPE):

$$\nabla^2 p = \rho (\nu \nabla \cdot \nabla^2 \mathbf{u} - \nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u})) + \nabla \cdot \mathbf{F}. \quad (7)$$

Discretizing the PPE inherently generates a massive, globally coupled linear system encompassing the entire computational domain — and solving this linear system is computationally intensive and presents significant hurdles for massive parallelization, frequently emerging as the primary performance bottleneck in high-fidelity fluid simulations. Furthermore, because the divergence-free constraint ($\nabla \cdot \mathbf{u} = 0$) acts as a mathematical abstraction rather than a strict physical reality, deriving rigorously consistent pressure boundary conditions for the PPE remains non-trivial. Although Dirichlet or Neumann boundary conditions corresponding to no-slip or no-penetration solid boundaries are ubiquitous in computer graphics pipelines, the exact mathematical formulations required to guarantee optimal numerical accuracy continue to be a subject of active debate within the theoretical fluid dynamics community [43].



Fig. 2. **Profile of Ludwig Eduard Boltzmann:** An Austrian physicist and philosopher, Boltzmann made his most significant contributions through the development of statistical mechanics — a framework that predicts macroscopic physical properties from atomic behavior — and his statistical formulation of the Second Law of Thermodynamics. The right velocity field is reproduced from medium.com.

2.2 The Boltzmann transport equation

Formulated in 1872 by Ludwig Eduard Boltzmann (Fig. 2, left), the Boltzmann transport equation (BTE) governs the temporal evolution and statistical behavior of particles in a dilute gas [32].

By capturing these dynamics, the BTE serves as a foundational tool for deriving macroscopic conservation laws for mass, momentum, and energy. Furthermore, it enables the theoretical extraction of fundamental fluid transport coefficients, including thermal conductivity, kinematic viscosity, and even the electrical conductivity of ionized particles. Mathematically, this formulation is expressed as an integro-differential relation dictating the continuous trajectory of the single-particle distribution function, $f(\mathbf{x}, \mathbf{c}, t)$. This function quantifies the distribution density of locating a particle at spatial coordinates $\mathbf{x}$, possessing a velocity vector $\mathbf{c}$, at a specific time t . The governing expression takes the following structure using Einstein notation for repeated Greek indices:

$$\partial_t f + c_\alpha \partial_\alpha f + G_\alpha \partial_{c_\alpha} f = \Omega(f, f) \quad (8)$$

where G_α denotes the body force exerted per unit mass, whereas Ω defines the *collision* term. Examining the left side of the relation, the advection (or free-flight) phase is represented by the second term, while the third term accounts for

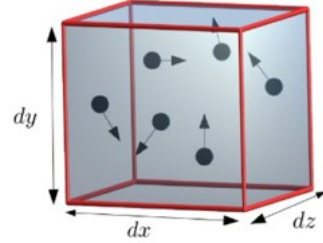


Fig. 3. Particle distribution function f

any external forces acting on the domain. The right-hand side is solely dedicated to modeling binary intermolecular interactions. To construct the collision mechanism inherent in this model, Boltzmann invoked the "Stosszahlansatz" (the molecular chaos hypothesis). This critical assumption dictates that pre-collision particles exhibit no statistical correlation prior to undergoing a binary interaction. Adhering to this postulate allows the collision operator to be expanded as a multi-dimensional integral spanning the velocity space:

$$\Omega = \iiint W(\mathbf{c}', \mathbf{c}'_1 | \mathbf{c}, \mathbf{c}_1) [f(\mathbf{c}')f(\mathbf{c}'_1) - f(\mathbf{c})f(\mathbf{c}_1)] d\mathbf{c}_1 d\mathbf{c}' d\mathbf{c}'_1 \quad (9)$$

Here, the transition probability kernel, $W(\mathbf{c}', \mathbf{c}'_1 | \mathbf{c}, \mathbf{c}_1)$, characterizes the likelihood that a colliding pair with initial velocities $\mathbf{c}$ and $\mathbf{c}_1$ will scatter into the post-collision velocity states $\mathbf{c}'$ and $\mathbf{c}'_1$. The specific formulation of this probability metric is intrinsically tied to the microscopic physical interactions and specific intermolecular potentials of the gas.

2.2.1 The collision operator. The collision term Ω in the Boltzmann equation encapsulates both the macroscopic irreversibility of a fluid system and its microscopic molecular details. Fundamentally, it dictates the rate of change of free particles based entirely on binary interactions, capturing the dynamic balance between the "loss" of particles as they enter each other's sphere of influence and the "gain" of particles as they separate after a collision [7]. The subsequent discussion delineates the defining mathematical and physical properties of this collision integral.

Spatial Locality and Velocity Non-Locality. Because the BTE relies on the idealized point-particle assumption, scattering interactions are strictly localized within the physical coordinate space, yet they remain inherently non-local across the velocity spectrum. From a computational perspective, this spatial localization is incredibly advantageous; calculating the interaction operator at any specific node requires only data immediately present at that exact location, streamlining numerical schemes.

Conservation of Additive Invariants. Within the standard Boltzmann framework, molecular scattering is treated as a perfectly elastic process. Consequently, fundamental macroscopic metrics—specifically particle mass (density), linear momentum, and total kinetic energy—must remain perfectly conserved across pre- and post-collision states. Mathematically, the integration of these invariant quantities against the collision kernel evaluates to zero:

$$\int (\chi + \boldsymbol{\zeta} \cdot \mathbf{c} + \lambda c^2) \Omega(f, f) d\mathbf{c} = 0 \quad (10)$$

In this expression, the parameters χ , $\boldsymbol{\zeta}$, and λ act as arbitrary scalar and vector constants. This conservation law is crucial: it dictates that any temporal evolution of localized hydrodynamic fields is driven entirely by the spatial advection (free flight) of the probability distribution, rather than being generated or destroyed by the collisions themselves.

Null Space and Equilibrium Formulations. When a system reaches statistical equilibrium, the collision operator evaluates to zero, a state governed by the principle of detailed balance:

$$\Omega(f, f) = 0 \quad (11)$$

This null condition mathematically translates to the identity $f(\mathbf{c})f(\mathbf{c}_1) = f(\mathbf{c}')f(\mathbf{c}'_1)$, which holds true across virtually the entire velocity domain. Identifying the roots of Eq. (11) is a foundational step in asymptotic analysis techniques, such as the Chapman-Enskog expansion, where these null solutions serve as the zeroth-order baseline for approximating the distribution function. Solutions satisfying this zero-collision criterion generally belong to a specific

exponential family:

$$f = \exp(\chi + \zeta \cdot \mathbf{c} + \lambda \mathbf{c}^2) \quad (12)$$

The most physically significant member of this functional class is the local Maxwellian equilibrium distribution, f^{eq} . This specific distribution is entirely dictated by locally conserved macroscopic variables—namely mass density ρ , momentum density $\rho \mathbf{u}$, and total energy—and is formulated as (Appendix A):

$$f^{\text{eq}}(\mathbf{c}, \rho, \mathbf{u}, T) = \rho \left(\frac{m}{2\pi k_B T} \right)^{D/2} \exp \left(-\frac{m(\mathbf{c} - \mathbf{u})^2}{2k_B T} \right) \quad (13)$$

where D indicates the spatial dimension, m represents the individual molecular mass, T denotes the localized thermodynamic temperature, and k_B stands for the Boltzmann constant. This perspective is linked to classical thermodynamics via the specific gas constant, defined as $R = k_B/m$. By substituting $\frac{m}{k_B T}$ with $\frac{1}{RT}$, the normalization coefficient naturally transforms to $(2\pi RT)^{-D/2}$, and the exponential argument simplifies to $-(\mathbf{c} - \mathbf{u})^2/(2RT)$. This yields the macroscopic thermodynamic representation of the equilibrium distribution:

$$f^{\text{eq}}(\mathbf{c}, \rho, \mathbf{u}, T) = \frac{\rho}{(2\pi RT)^{D/2}} \exp \left(-\frac{(\mathbf{c} - \mathbf{u})^2}{2RT} \right) \quad (14)$$

Furthermore, in isothermal fluid formulations, it is standard practice to substitute the specific gas constant with the isothermal speed of sound, $c_s = \sqrt{RT}$. This substitution elegantly distills the distribution into its fundamental hydrodynamic form, which serves as the continuous foundation for lattice Boltzmann velocity space discretizations:

$$f^{\text{eq}}(\mathbf{c}, \rho, \mathbf{u}, T) = \frac{\rho}{(2\pi c_s^2)^{D/2}} \exp \left(-\frac{(\mathbf{c} - \mathbf{u})^2}{2c_s^2} \right) \quad (15)$$

Thermodynamic Entropy Inequality. Finally, the collision integral intrinsically respects the second law of thermodynamics, as demonstrated by Boltzmann's theorem for local entropy production:

$$\dot{s}(\mathbf{x}, t) = -k_B \int \Omega(f, f) \ln f \, d\mathbf{c} \geq 0 \quad (16)$$

This inequality is valid for any definable distribution function f . The relation reaches strict equality (zero entropy production) if and only if the distribution resides within the null space of the collision operator—in particular, when it matches the Maxwellian state defined above. Consequently, the integral acts as a mathematical driver of relaxation, continuously forcing any non-equilibrium distribution f to decay toward its localized equilibrium counterpart, f^{eq} .

2.2.2 *H-theorem.* Proposed by Ludwig Boltzmann in 1872 alongside his eponymous equation, the H-theorem remains a cornerstone of statistical mechanics. Crucially, it provides a quantitative framework for understanding how macroscopically irreversible phenomena emerge from microscopically reversible dynamics [14]. Furthermore, this theoretical breakthrough generalized the thermodynamic concept of entropy, extending its application from strictly equilibrium states to non-equilibrium systems. The progression of macroscopic irreversibility correlates directly with entropy growth, which is mathematically evaluated using the H -function:

$$H(f) = \int f \ln f \, d\mathbf{c}$$

Assuming spatial uniformity, the local H-theorem establishes that the temporal evolution of the H-function is directly proportional to the entropy production rate, σ :

$$\partial_t H = -\frac{1}{k_B} \sigma$$

When dealing with spatially varying distribution functions, the local entropy density is expressed as $S(x, t) = -k_B H(x, t)$. This quantity satisfies the general entropy balance equation:

$$\partial_t S(x, t) + \partial_\alpha J_s^\alpha(x, t) = \sigma(x, t)$$

where the entropy flux vector is defined as $J_s(x, t) = -k_B \int c f(x, t) \ln f(x, t) dc$.

2.2.3 Models for the collision integral. Solving the exact Boltzmann equation presents substantial mathematical challenges, primarily due to the inherent nonlinearity of its collision operator. To make the framework computationally tractable in applied scenarios, it is practically necessary to employ approximation strategies or surrogate models. Consequently, a variety of theoretical frameworks—commonly referred to as kinetic models—have been developed in the literature to approximate the complex collision dynamics. The overarching goal of these formulations is to significantly alleviate the mathematical complexity while preserving the fundamental physical behaviors and core properties of the Boltzmann equation.

The Bhatnagar-Gross-Krook (BGK) model. To circumvent the intractable complexities of the exact collision integral while strictly adhering to the fundamental principles of the H -theorem, the Bhatnagar-Gross-Krook (BGK) approximation [8] emerged as a widely adopted kinetic framework. The BGK collision operator is formulated as:

$$\Omega_{BGK} = -\frac{1}{\tau} [f - f^{\text{eq}}(c, \rho(f), \mathbf{u}(f), T(f))] \quad (17)$$

In this expression, the strictly positive parameter τ denotes the relaxation time, representing the characteristic temporal scale required for the particle distribution to locally converge toward the Maxwell-Boltzmann equilibrium state. From a physical standpoint, τ shares the same order of magnitude as the mean free time between successive molecular interactions [15]. While the BGK formulation remains inherently non-linear, it drastically reduces the mathematical dimensionality associated with the original Boltzmann equation. The macroscopic fluid variables are systematically recovered by evaluating the successive velocity moments of the particle distribution function, f . Specifically, the local fluid density ρ , the macroscopic momentum vector $\rho \mathbf{u}$, and the unclosed momentum flux tensor Π correspond to the zeroth, first, and second-order velocity integrals over the continuous phase space, respectively:

$$\rho = \int f dc; \quad \rho \mathbf{u} = \int c f dc; \quad \Pi = \int c c^T f dc \quad (18)$$

This second-order moment, Π , establishes the critical link to the macroscopic continuum regime. It is constitutively coupled to the thermodynamic pressure p and the viscous stress tensor $\boldsymbol{\sigma}$ through the macroscopic momentum flux decomposition: $\Pi = \rho \mathbf{u} \otimes \mathbf{u} + p \mathbf{I} - \boldsymbol{\sigma}$. Furthermore, within this kinetic framework, the scalar pressure p is completely determined by the variance of the fluctuations of what is sometimes referred to as the *peculiar velocity*, $\mathbf{c} - \mathbf{u}$:

$$p = \frac{1}{D} \int \|\mathbf{c} - \mathbf{u}\|^2 f dc \approx \frac{1}{D} \int \|\mathbf{c} - \mathbf{u}\|^2 f^{\text{eq}} dc = \rho RT = \rho c_s^2 \quad (19)$$

This represents a huge analytical simplification over the exact collision integral, which functionally depends on the entire continuous distribution and its infinite hierarchy of moments.

2.3 From Boltzmann-BGK Equation to N-S Equation

The Boltzmann-BGK Equation with Forcing Moments. The continuous Boltzmann equation with the Bhatnagar-Gross-Krook (BGK) collision operator and a constant (or slowly varying) external force is given by:

$$\partial_t f + c_\alpha \partial_\alpha f + a_\alpha \partial_{c_\alpha} f = -\frac{1}{\tau} (f - f^{\text{eq}})$$

where f is the particle distribution function, c_α is the microscopic velocity, $a_\alpha = F_\alpha/\rho$ is the acceleration caused by the external body force density F_α , τ is the relaxation time, where Einstein notation is used again to imply summation over repeated Greek indices. Before expanding, we calculate the first three velocity moments of the external forcing term $\Phi = a_\alpha \partial_{c_\alpha} f$. Assuming the distribution function vanishes at infinity, integration by parts yields: *Zeroth Moment* (Mass contribution):

$$\int a_\alpha \partial_{c_\alpha} f d\mathbf{c} = 0.$$

First Moment (Momentum contribution):

$$\int c_\alpha (a_\beta \partial_{c_\beta} f) d\mathbf{c} = -a_\beta \int f \delta_{\alpha\beta} d\mathbf{c} = -\rho a_\alpha = -F_\alpha.$$

Second Moment (Momentum flux contribution):

$$\int c_\alpha c_\beta (a_\gamma \partial_{c_\gamma} f) d\mathbf{c} = - \int f (\delta_{\alpha\gamma} c_\beta + \delta_{\beta\gamma} c_\alpha) a_\gamma d\mathbf{c} = -(F_\alpha u_\beta + F_\beta u_\alpha).$$

Chapman-Enskog Multi-Scale Expansion. We introduce a formal small parameter ϵ , associated with the Knudsen number, to characterize the separation between kinetic and hydrodynamic scales. To ensure the external force appears in the Navier-Stokes equations (balancing pressure gradients and convective terms), it is scaled at the first order: $a_\alpha = \epsilon a_{1\alpha}$. We apply a multiscale expansion for the differential operators and the distribution function:

$$\partial_t = \epsilon \partial_{t_1} + \epsilon^2 \partial_{t_2},$$

$$\partial_\alpha = \epsilon \partial_{1\alpha},$$

$$f = f^{(0)} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + O(\epsilon^3).$$

Substituting these expansions into the BGK equation and separating terms by powers of ϵ , we obtain the governing equations at different scales:

$O(\epsilon^0)$ order:

$$f^{(0)} = f^{\text{eq}}.$$

$O(\epsilon^1)$ order:

$$\partial_{t_1} f^{(0)} + c_\alpha \partial_{1\alpha} f^{(0)} + a_{1\alpha} \partial_{c_\alpha} f^{(0)} = -\frac{1}{\tau} f^{(1)}.$$

$O(\epsilon^2)$ order:

$$\partial_{t_2} f^{(0)} + (\partial_{t_1} + c_\alpha \partial_{1\alpha}) f^{(1)} + a_{1\alpha} \partial_{c_\alpha} f^{(1)} = -\frac{1}{\tau} f^{(2)}.$$

The compatibility conditions dictate that non-equilibrium parts contribute nothing to the conserved macroscopic quantities, that is,

$$\int f^{(k)} d\mathbf{c} = 0, \quad \int c_\alpha f^{(k)} d\mathbf{c} = 0 \quad \forall k \geq 1.$$

The Euler Equations with Forcing. Taking the zeroth (1) and first (c_α) velocity moments of the $O(\epsilon^1)$ equation, and utilizing the previously derived moments of the forcing term, we obtain:

Continuity Equation at t_1 scale:

$$\partial_{t_1} \rho + \partial_{1\alpha} (\rho u_\alpha) = 0.$$

Momentum Equation at t_1 scale:

$$\partial_{t_1} (\rho u_\alpha) + \partial_{1\beta} \int c_\alpha c_\beta f^{(0)} d\mathbf{c} - \rho a_{1\alpha} = 0.$$

Substituting the zeroth-order momentum flux tensor $\Pi_{\alpha\beta}^{(0)} = \int c_\alpha c_\beta f^{(0)} d\mathbf{c} = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta$ (where $p = \rho c_s^2$), we derive the inviscid Euler equations:

$$\partial_{t_1}(\rho u_\alpha) + \partial_{1\beta}(\rho u_\alpha u_\beta + p\delta_{\alpha\beta}) = F_{1\alpha}.$$

Exact Cancellation and Viscous Stress. Taking the first moment (c_α) of the $O(\epsilon^2)$ equation yields:

$$\partial_{t_2}(\rho u_\alpha) + \partial_{1\beta} \underbrace{\int c_\alpha c_\beta f^{(1)} d\mathbf{c}}_{\Pi_{\alpha\beta}^{(1)}} = 0$$

Note that the direct force term $\int c_\alpha (a_{1\gamma} \partial_{c_\gamma} f^{(1)}) d\mathbf{c}$ evaluates to $-a_{1\alpha} \int f^{(1)} d\mathbf{c} = 0$ due to compatibility conditions. All dissipative effects are encapsulated within the first-order non-equilibrium momentum-flux tensor $\Pi_{\alpha\beta}^{(1)}$. To evaluate $\Pi_{\alpha\beta}^{(1)}$, we substitute the expression for $f^{(1)}$ extracted from the $O(\epsilon^1)$ equation, yielding:

$$\Pi_{\alpha\beta}^{(1)} = -\tau \left[\partial_{t_1} \Pi_{\alpha\beta}^{(0)} + \partial_{1\gamma} \int c_\alpha c_\beta c_\gamma f^{(0)} d\mathbf{c} + \int c_\alpha c_\beta a_{1\gamma} \partial_{c_\gamma} f^{(0)} d\mathbf{c} \right]$$

This integral consists of three parts. The third part represents the direct contribution from the external force, which evaluates to $-(F_{1\alpha} u_\beta + F_{1\beta} u_\alpha)$. The first part, the time derivative of the equilibrium momentum flux $\partial_{t_1} \Pi_{\alpha\beta}^{(0)}$, must be expanded using the $O(\epsilon^1)$ continuity and Euler equations to replace time derivatives with spatial gradients. When substituting $\partial_{t_1}(\rho u_\alpha) = -\partial_{1\gamma}(\dots) + F_{1\alpha}$ into the expansion, it produces an additional term: $+(F_{1\alpha} u_\beta + F_{1\beta} u_\alpha)$. Consequently, when evaluating $\Pi_{\alpha\beta}^{(1)}$, the direct forcing term and the force-induced convective derivative perfectly cancel each other out:

$$[(F_{1\alpha} u_\beta + F_{1\beta} u_\alpha)] + [-(F_{1\alpha} u_\beta + F_{1\beta} u_\alpha)] \equiv 0$$

After algebraic reduction of the remaining spatial gradients given a low-Mach-number (weakly compressible) assumption, the non-equilibrium stress tensor yields the pure Newtonian viscous stress:

$$\Pi_{\alpha\beta}^{(1)} = -\tau \rho c_s^2 (\partial_{1\alpha} u_\beta + \partial_{1\beta} u_\alpha)$$

Substituting this back into the $O(\epsilon^2)$ momentum equation, one gets:

$$\partial_{t_2}(\rho u_\alpha) = \partial_{1\beta} [\mu (\partial_{1\alpha} u_\beta + \partial_{1\beta} u_\alpha)],$$

where the dynamic viscosity is $\mu = \tau \rho c_s^2$.

Reconstitution: The Navier-Stokes Equations. Finally, we recombine the multiple scales back into physical space-time by summing $\epsilon \times [O(\epsilon^1)]$ and $\epsilon^2 \times [O(\epsilon^2)]$. Utilizing $\partial_t = \epsilon \partial_{t_1} + \epsilon^2 \partial_{t_2}$ and $F_\alpha = \epsilon F_{1\alpha}$, we recover the macroscopic equations:

Continuity Equation:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0.$$

Momentum Equation: (Weakly Compressible Navier-Stokes):

$$\partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \nabla \cdot \left[\mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \right] + \mathbf{F}$$

This derivation shows that, under the adopted forcing scaling and low-Mach-number approximation, the force-dependent contributions to the first-order non-equilibrium stress cancel exactly. Consequently, the recovered viscous stress retains

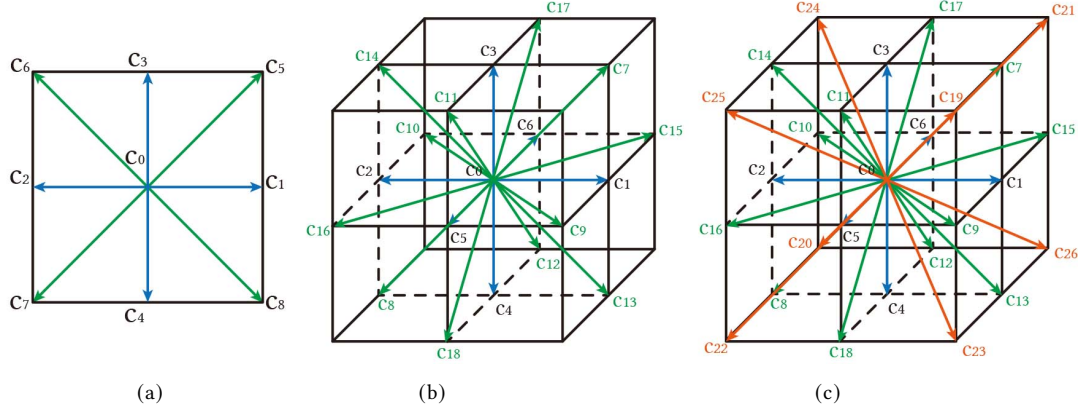


Fig. 4. **Lattice structures.** One typically uses D2Q9 (a) in 2D and either D3Q19 (b) or D3Q27 (c) in 3D for LBM fluid simulation, where the lattice directions c_i are discretized microscopic velocities. Each discretized distribution function f_i is associated with its corresponding velocity c_i .

the standard Newtonian form, without introducing spurious force-dependent terms, and the resulting macroscopic equations correspond to the weakly compressible Navier–Stokes equations with the prescribed body force.

2.4 Lattice Boltzmann method

The foundation of the lattice Boltzmann method (LBM) lies in replacing the *continuous particle velocity space* with a *finite, truncated set of microscopic velocities*. To circumvent the computational hurdles associated with the exact continuous kinetic equations, the Lattice Boltzmann Method (LBM) integrates the underlying Boltzmann dynamics by projecting the continuous particle distribution function, f , onto a finite discrete subspace [86].

In this nondimensional velocity space, the standard Maxwell-Boltzmann equilibrium weighting function modulates the continuous velocity domain, shown in Appendix B, mathematically defined as:

$$\omega(\mathbf{v}) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{\|\mathbf{v}\|^2}{2}\right). \quad (20)$$

Furthermore, the n -th order Hermite polynomial tensor is generated via the subsequent gradient relation:

$$\mathbf{H}^{(n)}(\mathbf{v}) = \frac{(-1)^n}{\omega(\mathbf{v})} \nabla^n \omega(\mathbf{v}). \quad (21)$$

Any continuous distribution function $f(\mathbf{x}, \mathbf{v}, t)$ can be expanded into an infinite series using this orthogonal basis:

$$f(\mathbf{x}, \mathbf{v}, t) = \omega(\mathbf{v}) \sum_{n=0}^{\infty} \frac{1}{n!} a^{(n)}(\mathbf{x}, t) : \mathbf{H}^{(n)}(\mathbf{v}). \quad (22)$$

To rigorously extract the expansion coefficient $a^{(n)}$, we utilize the orthogonality relation $\int \omega(\mathbf{v}) \mathbf{H}^{(m)}(\mathbf{v}) \mathbf{H}^{(n)}(\mathbf{v}) d\mathbf{v} = n! \delta_{mn} I$. Multiplying both sides of the expansion by $\mathbf{H}^{(n)}(\mathbf{v})$ and integrating over the entire velocity space yields the coefficient formula:

$$a^{(n)}(\mathbf{x}, t) = \int f(\mathbf{x}, \mathbf{v}, t) \mathbf{H}^{(n)}(\mathbf{v}) d\mathbf{v}. \quad (23)$$

To transition from a continuous analytical model to a computable numerical scheme, the continuous velocity integrals are approximated using Gauss-Hermite quadrature, which transforms integrals with weight $\omega(\mathbf{c})$ into a discrete sum

over quadrature nodes c_i with associated weights w_i :

$$\int \omega(\mathbf{v}) P(\mathbf{v}) d\mathbf{v} \approx \sum_{i=0}^{q-1} w_i P(v_i). \quad (24)$$

To apply this to our coefficient integral, we mathematically extract $\omega(\mathbf{v})$ into the integrand:

$$a^{(n)} = \int \omega(\mathbf{v}) \left[\frac{f(\mathbf{v})}{\omega(\mathbf{v})} \mathbf{H}^{(n)}(\mathbf{v}) \right] d\mathbf{v}. \quad (25)$$

Applying the quadrature rule yields the discrete assembly equation:

$$a^{(n)}(\mathbf{x}, t) \approx \sum_{i=0}^{q-1} w_i \left[\frac{f(v_i)}{\omega(v_i)} \right] \mathbf{H}^{(n)}(v_i). \quad (26)$$

At this point, we define the fundamental LBM variable — the *discrete distribution function* f_i — through the following expression:

$$f_i(\mathbf{x}, t) \equiv w_i \frac{f(v_i, \mathbf{x}, t)}{\omega(v_i)}. \quad (27)$$

Mathematically, the quadrature nodes v_i are the roots of Hermite polynomials (e.g., $v_i \in \{-\sqrt{3}, 0, \sqrt{3}\}$ for 3rd-order [85]). However, LBM requires particles to stream exactly onto adjacent grid nodes, necessitating integer lattice velocities $c_i \in \{-1, 0, 1\}$. To reconcile the irrational mathematical roots with the integer geometric lattice, a velocity scaling — the Lattice Speed of Sound (c_s) — is introduced. The mathematical roots v_i are mapped to the integer lattice directions c_i via:

$$v_i = \frac{c_i}{c_s}.$$

Setting the lattice velocity $c_i = 1$ to correspond to the mathematical root $v_i = \sqrt{3}$ dictates that:

$$c_s = \frac{1}{\sqrt{3}} \implies c_s^2 = \frac{1}{3}$$

This scaling constant is universally embedded into LBM numerical implementations to ensure geometric coupling while preserving integral accuracy.

This orthogonal projection is greatly advantageous for fluid simulation frameworks, as the lowest-order expansion coefficients seamlessly and directly map to the fundamental macroscopic conserved variables and the specific pressure-subtracted momentum-flux tensor S :

$$a^{(0)} = \rho, \quad a^{(1)} = \rho \mathbf{u}, \quad a^{(2)} = \Pi - \rho c_s^2 \mathbf{I} = \rho S. \quad (28)$$

From Standard Discretization to the Second-Order Forced Lattice Boltzmann Equation. By synergizing the orthogonal Hermite projection with the finite Gauss-Hermite quadrature scheme, the continuous velocity domain of the foundational Boltzmann transport equation can be rigorously discretized. In the absence of external forces, this yields the standard lattice Boltzmann equation (LBE):

$$f_i(\mathbf{x} + c_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t). \quad (29)$$

Physically, this indicates that particle distributions $f_i(\mathbf{x}, t)$ with discrete velocity c_i first undergo a collision process governed by the collision operator $\Omega_i(\mathbf{x}, t)$, and subsequently stream to the neighboring lattice node $\mathbf{x} + c_i \Delta t$ at the next time step $t + \Delta t$. Consequently, the macroscopic fluid fields are highly optimized for computation, evaluated efficiently

via direct summations over the permitted discrete lattice directions ($i = 0, 1, \dots, q-1$):

$$\rho = \sum_{i=0}^{q-1} f_i, \quad \rho \mathbf{u} = \sum_{i=0}^{q-1} c_i f_i, \quad \Pi = \sum_{i=0}^{q-1} c_i c_i^T f_i, \quad p = \frac{1}{D} \sum_{i=0}^{q-1} \|c_i - \mathbf{u}\|^2 f_i. \quad (30)$$

The parameter q dictates the total count of discrete microscopic velocity vectors, $\{c_i\}_{i=0}^{q-1}$, retained in the truncated lattice model, while w_i represents the associated quadrature weights (frequently referred to as lattice weights). While various discretization stencils can be formulated, the D2Q9 and D3Q27 lattice structures (shown in Fig 4) remain the standard, widely adopted configurations for two- and three-dimensional spatial domains, respectively. For the 2D D2Q9 scheme, the discrete velocity vectors, scaled by the fundamental lattice speed c_s , are classified as:

$$c_i = \begin{cases} (0, 0), & i = 0 \\ (\pm 1, 0), (0, \pm 1), & i = 1, 2, 3, 4 \\ (\pm 1, \pm 1), & i = 5, 6, 7, 8 \end{cases} \quad (31)$$

Similarly, the 3D D3Q27 stencil populates its velocity directions as follows:

$$c_i = \begin{cases} (0, 0, 0), & i = 0 \\ (\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1), & i = 1, \dots, 6 \\ (\pm 1, \pm 1, 0), (\pm 1, 0, \pm 1), (0, \pm 1, \pm 1), & i = 7, \dots, 18 \\ (\pm 1, \pm 1, \pm 1), & i = 19, \dots, 26 \end{cases}$$

To guarantee critical properties such as Galilean invariance and isotropy, the corresponding numerical lattice weights for the D2Q9 model are prescribed as:

$$w_i = \begin{cases} 4/9, & i = 0 \\ 1/9, & i = 1, 2, 3, 4 \\ 1/36, & i = 5, 6, 7, 8. \end{cases}$$

For the D3Q27 architecture, the exact quadrature weights are determined to be:

$$w_i = \begin{cases} 8/27, & i = 0 \\ 2/27, & i = 1, \dots, 6 \\ 1/54, & i = 7, \dots, 18 \\ 1/216, & i = 19, \dots, 26. \end{cases}$$

Crank-Nicolson Integration. While the above explicit Euler formulation is elegant for force-free flows, the introduction of a discrete external forcing term $\Psi(F)$ disrupts this simplicity: if Ψ_i is directly added to the standard explicit LBE, it introduces a first-order time truncation error, $\mathcal{O}(\Delta t)$. Through the Chapman-Enskog expansion, this error manifests as non-physical numerical viscosity, severely polluting the recovered Navier-Stokes hydrodynamics. To strictly preserve second-order temporal accuracy, the explicit integration must be replaced by the Crank-Nicolson (trapezoidal) rule along the characteristic line. Integrating the forced continuous equation over a time step Δt yields an implicit formulation:

$$f_i(\mathbf{x} + c_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = \frac{\Delta t}{2} [\Omega_i(t + \Delta t) + \Omega_i(t)] + \frac{\Delta t}{2} [\Psi_i(t + \Delta t) + \Psi_i(t)].$$

Explicit Formulation via an Auxiliary Variable. To circumvent the computationally expensive implicit evaluation of the collision and forcing terms at $t + \Delta t$, an auxiliary distribution function $\tilde{f}_i$ is introduced:

$$\tilde{f}_i(\mathbf{x}, t) = f_i(\mathbf{x}, t) - \frac{\Delta t}{2} \Omega_i(\mathbf{x}, t) - \frac{\Delta t}{2} \Psi_i(\mathbf{x}, t)$$

Rearranging the implicit Crank-Nicolson equation allows the temporal evolution to be expressed completely explicitly in terms of $\tilde{f}_i$, yielding:

$$\tilde{f}_i(\mathbf{x} + c_i \Delta t, t + \Delta t) = \tilde{f}_i(\mathbf{x}, t) + \Delta t \Omega_i(\mathbf{x}, t) + \Delta t \Psi_i(\mathbf{x}, t).$$

To mathematically close the system, the BGK collision operator $\Omega_i = -\frac{1}{\tau}(f_i - f_i^{\text{eq}})$ must be reformulated. Substituting Ω_i into the definition of $\tilde{f}_i$ and subtracting the equilibrium state f_i^{eq} from both sides isolates the non-equilibrium part:

$$f_i - f_i^{\text{eq}} = \frac{\tau}{\tau + \Delta t/2} \left(\tilde{f}_i - f_i^{\text{eq}} + \frac{\Delta t}{2} \Psi_i \right)$$

Substituting this back into the collision operator yields:

$$\Omega_i = -\frac{1}{\tau + \Delta t/2} (\tilde{f}_i - f_i^{\text{eq}}) - \frac{\Delta t/2}{\tau + \Delta t/2} \Psi_i$$

Inserting this modified Ω_i into the explicit evolution equation, setting the dimensionless time step $\Delta t = 1$, and defining the effective relaxation time as $\tau^* = \tau + 0.5$ for $\tau = \frac{\nu}{c_s^2}$ yield the final *explicit, second-order accurate LBE with forcing*:

$$\tilde{f}_i(\mathbf{x} + c_i, t + 1) = \tilde{f}_i(\mathbf{x}, t) - \frac{1}{\tau^*} \left[\tilde{f}_i(\mathbf{x}, t) - f_i^{\text{eq}}(\mathbf{x}, t) \right] + \left(1 - \frac{1}{2\tau^*} \right) \Psi_i(\mathbf{x}, t). \quad (32)$$

The Split-Step Operator. For notational simplicity, the auxiliary variable $\tilde{f}$ will hereafter be denoted simply as f . In a practical computational implementation, the monolithic explicit evolution equation derived above is decoupled using a fractional split-step (or splitting operator) approach. This elegantly divides the complex kinetic dynamics into two sequential, highly efficient update rules. The first is a strictly local collision and forcing step, which models the redistribution of particles due to microscopic collisions alongside the instantaneous momentum injection from the external body force. This step calculates the post-collision state, denoted by $f_i^*(\mathbf{x}, t)$:

$$f_i^*(\mathbf{x}, t) = f_i(\mathbf{x}, t) - \frac{1}{\tau^*} \left[f_i(\mathbf{x}, t) - f_i^{\text{eq}}(\mathbf{x}, t) \right] + \left(1 - \frac{1}{2\tau^*} \right) \Psi_i(\mathbf{x}, t). \quad (33)$$

Because this operation does not involve any spatial gradients or neighbor information, it is completely local to each grid node, making it perfectly suited for massive parallelization. Subsequently, a purely advective streaming step models the exact spatial propagation of these post-collision particles to their designated neighboring lattice nodes along the discrete velocity directions:

$$f_i(\mathbf{x} + c_i, t + 1) = f_i^*(\mathbf{x}, t). \quad (34)$$

This step involves straightforward memory fetching and writing, preserving the exact advection nature of the lattice Boltzmann framework.

Macroscopic Reconstitution (The Half-Step Correction). Because the numerical algorithm now strictly evolves the auxiliary variable $\tilde{f}_i$ rather than the raw distribution f_i , the standard moment summations defined previously must be modified. The physical macroscopic variables are reconstituted via the inverse relation $f_i = \tilde{f}_i + \frac{\Delta t}{2} \Omega_i + \frac{\Delta t}{2} \Psi_i$. Leveraging the mass and momentum invariant properties of the collision operator ($\sum \Omega_i = 0$, $\sum c_i \Omega_i = 0$) alongside the moments of the forcing term ($\sum F_i = 0$, $\sum c_i F_i = F$), the macroscopic variables are retrieved as follows:

The mass density remains unaltered:

$$\rho = \sum_{i=0}^{q-1} f_i.$$

The macroscopic velocity now involves a half-step shift:

$$\rho \mathbf{u} = \sum_{i=0}^{q-1} c_i f_i + \frac{1}{2} \mathbf{F}.$$

As a result, the physical macroscopic velocity required for equilibrium calculations becomes:

$$\mathbf{u} = \frac{1}{\rho} \sum_{i=0}^{q-1} c_i f_i + \frac{1}{2\rho} \mathbf{F}.$$

In this kinetic framework, f_i^{eq} represents the discrete *equilibrium distribution function* (EDF). It is analytically constructed by projecting the continuous local Maxwellian equilibrium state onto the aforementioned discrete Hermite tensor basis. For standard hydrodynamic regimes, it is typically sufficient to truncate this Hermite polynomial expansion at the second order to accurately recover the macroscopic Navier-Stokes equations. This widely adopted second-order truncation yields the classical quadratic approximation for the discrete BGK equilibrium distribution:

$$f_i^{\text{eq},2}(\rho, \mathbf{u}) = w_i \rho \left(1 + \frac{c_i \cdot \mathbf{u}}{c_s^2} + \frac{(c_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\|\mathbf{u}\|^2}{2c_s^2} \right) \quad (35)$$

To achieve better simulation accuracy, a higher-order expansion is needed. As proposed in [86], we can employ the highest order Hermite series expansion of f^{eq} . The maximum order is determined by the number of lattice velocities: in 2D (for the D2Q9 lattice structure), it is fourth order, and the discrete equilibrium distribution functions f_i^{eq} are

$$\begin{aligned} f_i^{\text{eq},(4)} \approx w_i \rho & \left[1 + \frac{c_i \cdot \mathbf{u}}{c_s^2} + \frac{1}{2c_s^4} \mathbf{H}^{(2)}(c_i) : \mathbf{u} \otimes \mathbf{u} \right. \\ & + \frac{1}{2c_s^6} \left(\mathbf{H}_{ixxy}^{(3)} u_x^2 u_y + \mathbf{H}_{ixyy}^{(3)} u_x u_y^2 \right) \\ & \left. + \frac{1}{4c_s^8} \mathbf{H}_{ixxyy}^{(4)} u_x^2 u_y^2 \right], \end{aligned} \quad (36)$$

where $\mathbf{H}_{i\alpha_1 \dots \alpha_n}^{(n)}$ is a shorthand notation of the tensor element of $\mathbf{H}^{(n)}(c_i)$ corresponding to the $\alpha_1 \dots \alpha_n$ component. In 3D (for the D3Q27 lattice structure), the highest expansion order is the sixth order, and the full expansion of the

equilibrium distribution functions yields

$$\begin{aligned}
f_i^{\text{eq},(6)} \approx w_i \rho & \left[1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{1}{2c_s^4} \mathbf{H}^{(2)}(\mathbf{c}_i) : \mathbf{u} \otimes \mathbf{u} \right. \\
& + \frac{1}{2c_s^6} \left(\mathbf{H}_{ixxy}^{(3)} u_x^2 u_y + \mathbf{H}_{ixxz}^{(3)} u_x^2 u_z + \mathbf{H}_{ixyy}^{(3)} u_x u_y^2 \right. \\
& + \left. \mathbf{H}_{ixzz}^{(3)} u_x u_z^2 + \mathbf{H}_{iyzz}^{(3)} u_y u_z^2 + \mathbf{H}_{iyyz}^{(3)} u_y^2 u_z + \mathbf{H}_{ixyz}^{(3)} u_x u_y u_z \right) \\
& + \frac{1}{4c_s^8} \left[\mathbf{H}_{ixxyy}^{(4)} u_x^2 u_y^2 + \mathbf{H}_{ixxzz}^{(4)} u_x^2 u_z^2 + \mathbf{H}_{iyyzz}^{(4)} u_y^2 u_z^2 \right. \\
& + \left. 2 \left(\mathbf{H}_{ixyzz}^{(4)} u_x u_y u_z^2 + \mathbf{H}_{ixyyz}^{(4)} u_x u_y^2 u_z + \mathbf{H}_{ixxyz}^{(4)} u_x^2 u_y u_z \right) \right] \\
& + \frac{1}{4c_s^{10}} \left(\mathbf{H}_{ixxyzz}^{(5)} u_x^2 u_y u_z^2 + \mathbf{H}_{ixxyyz}^{(5)} u_x^2 u_y^2 u_z \right. \\
& + \left. \mathbf{H}_{ixyyzz}^{(5)} u_x u_y^2 u_z^2 \right) + \frac{1}{8c_s^{12}} \mathbf{H}_{ixxyyz}^{(6)} u_x^2 u_y^2 u_z^2 \left. \right]. \tag{37}
\end{aligned}$$

A defining hallmark of this kinetic paradigm is its strict data locality. Dictated by the finite discrete velocity stencils $\mathbf{c}_i$, both the streaming and collision operations strictly fetch information from the local grid point and its immediate topological neighbors. This highly localized memory access pattern stands in stark contrast to traditional macroscopic Navier-Stokes solvers, which inherently necessitate the computationally expensive resolution of a globally coupled pressure Poisson equation. Consequently, the LBM algorithm is intrinsically highly parallel. The absence of global linear system solves renders spatial domain decomposition trivial, thereby facilitating massive parallelization and delivering exceptionally high computational throughput on modern concurrent hardware architectures.

2.5 From LBM Space to Physical Space

Lattice Boltzmann simulations are performed in dimensionless normalized units. Indeed, we introduced our lattice Boltzmann equation (LBE) solver in Eq. (29) with $\Delta t = 1$ and $c = 1$. Formally, it takes two steps to convert dimensional physical scales to dimensionless lattice scales [60]: the first step converts physical scales to dimensionless scales, while the second step further converts the result to normalized lattice scales. In what follows, we denote time, position, density, velocity, acceleration, and pressure variables as t , $\mathbf{x}$, ρ , $\mathbf{u}$, $\mathbf{a}$ and p . These variables are in physical units. After choosing a fixed characteristic length l_{ref} , as well as a fixed characteristic density ρ_{ref} , and time scale t_{ref} , all in physical units, we obtain dimensionless variables by dividing by their corresponding characteristic values:

$$\hat{t} = \frac{t}{t_{\text{ref}}}, \quad \hat{\mathbf{x}} = \frac{\mathbf{x}}{l_{\text{ref}}}, \quad \hat{\rho} = \frac{\rho}{\rho_{\text{ref}}}, \quad \hat{\mathbf{u}} = \frac{\mathbf{u}}{u_{\text{ref}}} = \mathbf{u} \frac{t_{\text{ref}}}{l_{\text{ref}}}, \quad \hat{\mathbf{a}} = \frac{\mathbf{a}}{a_{\text{ref}}} = \mathbf{a} \frac{t_{\text{ref}}^2}{l_{\text{ref}}}, \quad \text{and} \quad \hat{p} = \frac{p}{\rho_{\text{ref}} l_{\text{ref}}^2}, \tag{38}$$

where we used, for consistency, a reference velocity magnitude u_{ref} and acceleration magnitude a_{ref} such that $u_{\text{ref}} = l_{\text{ref}}/t_{\text{ref}}$ and $a_{\text{ref}} = u_{\text{ref}}/t_{\text{ref}}$. This dimensionless normalization process gives rise to an important notion, called the *Reynolds number*, which is usually denoted as Re and defined as

$$Re = \frac{u_{\text{ref}} l_{\text{ref}}}{\nu} = \frac{l_{\text{ref}}^2}{t_{\text{ref}} \nu}, \tag{39}$$

where ν is the fluid viscosity, and all the variables are in physical units. The Reynolds number reflects the importance of the viscosity effect in a physical fluid system. A high Reynolds number indicates a low effective (and dimensionless) viscosity $\hat{\nu}$ (i.e., $\hat{\nu} = 1/Re$), and thus often suggests the existence of strong turbulence. According to the principle of dynamic similarity, two geometrically similar incompressible Newtonian flows with corresponding boundary conditions

exhibit dynamically similar behavior when their Reynolds numbers are equal, provided that other relevant dimensionless parameters are negligible or matched.

Next, we convert the variables into normalized lattice units. Characteristic length l_{ref} and time scale t_{ref} being unit in dimensionless units, the finest spatial and time lengths that a solver can resolve are denoted in dimensionless units as

$$\Delta\hat{x} = \frac{1}{N}, \quad \text{and} \quad \Delta\hat{t} = \frac{1}{T}, \quad (40)$$

where N and T are the spatial and time resolution of the LBE solver. Based on $\Delta\hat{x}$ and $\Delta\hat{t}$, we can convert the variables in lattice units, and denote these variables by outlined letters as follows:

$$\begin{aligned} \mathfrak{t} &= \frac{\hat{t}}{\Delta\hat{t}} = \frac{t}{t_{\text{ref}}\Delta\hat{t}} = \frac{Tt}{t_{\text{ref}}}, \quad \mathfrak{x} = \frac{\hat{x}}{\Delta\hat{x}} = \frac{\mathbf{x}}{l_{\text{ref}}\Delta\hat{x}} = \frac{N\mathbf{x}}{l_{\text{ref}}}, \\ \rho &= \hat{\rho} = \frac{\rho}{\rho_{\text{ref}}}, \quad \mathfrak{u} = \hat{\mathbf{u}} = \frac{\mathbf{u}}{l_{\text{ref}}\Delta\hat{x}} = \frac{t_{\text{ref}}}{l_{\text{ref}}} \frac{\Delta\hat{t}}{\Delta\hat{x}} \mathbf{u} = \frac{t_{\text{ref}}}{l_{\text{ref}}} \frac{N}{T} \mathbf{u}, \\ \mathfrak{a} &= \hat{a} = \frac{\Delta\hat{t}^2}{\Delta\hat{x}} = \frac{t_{\text{ref}}^2}{l_{\text{ref}}} \frac{\Delta\hat{t}^2}{\Delta\hat{x}} = \frac{t_{\text{ref}}^2}{l_{\text{ref}}} \frac{N}{T^2}, \quad \mathfrak{p} = \hat{p} = \frac{p}{\rho_{\text{ref}}} \frac{\Delta\hat{t}^2}{\Delta\hat{x}^2} = \frac{p}{\rho_{\text{ref}}} \frac{t_{\text{ref}}^2}{l_{\text{ref}}^2} \frac{\Delta\hat{t}^2}{\Delta\hat{x}^2} = \frac{p}{\rho_{\text{ref}}} \frac{t_{\text{ref}}^2}{l_{\text{ref}}^2} \frac{N^2}{T^2}, \quad \text{and} \\ \mathfrak{v} &= \hat{v} = \frac{\Delta\hat{t}}{\Delta\hat{x}^2} = \frac{t_{\text{ref}}}{l_{\text{ref}}^2} \frac{\Delta\hat{t}}{\Delta\hat{x}^2} = \frac{t_{\text{ref}}}{l_{\text{ref}}^2} \frac{N^2}{T}. \end{aligned} \quad (41)$$

The grid spacing and time step are set to unity in lattice units. In what follows, we rewrite the conversions listed in Eq. (41) into formulas that allow easier computation of the quantities in lattice units. Usually, it is not straightforward to choose t_{ref} directly. We therefore compute it based on other quantities. If the system has a reference velocity, then we have $t_{\text{ref}} = l_{\text{ref}}/u_{\text{ref}}$ as we mentioned earlier. For example, one might choose the magnitude of the fluid inlet velocity $\mathbf{u}_{\text{in}}$ as the reference velocity, i.e., $\mathbf{u}_{\text{in}} = u_{\text{ref}} \mathbf{r}$ where $\mathbf{r}$ is the inlet normal direction, and the inlet size as the reference length l_{ref} . We then substitute $t_{\text{ref}} = l_{\text{ref}}/u_{\text{ref}}$ in Eq. (41) and obtain

$$\begin{aligned} \mathfrak{t} &= \frac{Tt}{t_{\text{ref}}} = \frac{u_{\text{ref}}Tt}{l_{\text{ref}}}, \quad \mathfrak{x} = \frac{N\mathbf{x}}{l_{\text{ref}}}, \quad \rho = \frac{\rho}{\rho_{\text{ref}}}, \quad \mathfrak{u} = \frac{t_{\text{ref}}}{l_{\text{ref}}} \frac{N}{T} \mathbf{u} = \frac{N}{u_{\text{ref}}T} \mathbf{u}, \quad \mathfrak{a} = \frac{t_{\text{ref}}^2}{l_{\text{ref}}} \frac{N}{T^2} = \frac{l_{\text{ref}}}{u_{\text{ref}}^2} \frac{N}{T^2}, \\ \mathfrak{p} &= \frac{p}{\rho_{\text{ref}}} \frac{t_{\text{ref}}^2}{l_{\text{ref}}^2} \frac{N^2}{T^2} = \frac{p}{\rho_{\text{ref}}} \frac{N^2}{u_{\text{ref}}^2 T^2}, \quad \mathfrak{v} = \frac{t_{\text{ref}}}{l_{\text{ref}}^2} \frac{N^2}{T} = \frac{N^2}{u_{\text{ref}} l_{\text{ref}} T}. \end{aligned} \quad (42)$$

To ensure stable simulation, the condition $\Delta\hat{t} = \lambda \Delta\hat{x}$ for a value λ satisfying $\lambda < c_s$ must hold. From (40), we have the relation $T = \lambda^{-1}N$, which we then use in Eq. (42) and obtain

$$\mathfrak{t} = \frac{u_{\text{ref}}Nt}{\lambda l_{\text{ref}}}, \quad \mathfrak{x} = \frac{N\mathbf{x}}{l_{\text{ref}}}, \quad \rho = \frac{\rho}{\rho_{\text{ref}}}, \quad \mathfrak{u} = \lambda \frac{\mathbf{u}}{u_{\text{ref}}}, \quad \mathfrak{a} = \lambda^2 \frac{l_{\text{ref}}}{Nu_{\text{ref}}^2}, \quad \mathfrak{p} = \frac{p}{\rho_{\text{ref}}} \frac{\lambda^2}{u_{\text{ref}}^2}, \quad \mathfrak{v} = \lambda v \frac{N}{u_{\text{ref}} l_{\text{ref}}}. \quad (43)$$

Next, we eliminate the parameter λ . Oftentimes, in a given simulated scenario, we can roughly estimate the maximum fluid velocity magnitude that will happen, and set it as u_{ref} . In other scenarios, like when a still fluid is accelerated by gravity throughout a reference length l_{ref} , we may use the gravity g_0 in physical unit to estimate the maximum velocity, and set the reference velocity accordingly as $u_{\text{ref}} = \sqrt{2g_0 l_{\text{ref}}}$. Meanwhile, the LBE solver has a maximum fluid velocity that it can simulate stably; it is often around 0.5 Mach number. We therefore set it as the corresponding $\mathfrak{u}_{\text{ref}}$. From the fourth relation in Eq. (43), we have $\lambda = \mathfrak{u}_{\text{ref}}$, and using it to replace λ gives us the following relations:

$$\mathfrak{t} = \frac{u_{\text{ref}}}{\mathfrak{u}_{\text{ref}} \Delta x} t, \quad \mathfrak{x} = \frac{\mathbf{x}}{\Delta x}, \quad \rho = \frac{\rho}{\rho_{\text{ref}}}, \quad \mathfrak{u} = \frac{\mathfrak{u}_{\text{ref}} \mathbf{u}}{u_{\text{ref}}}, \quad \mathfrak{a} = \frac{\mathfrak{u}_{\text{ref}}^2 \Delta x}{u_{\text{ref}}^2} \mathfrak{a}, \quad \mathfrak{p} = \frac{p}{\rho_{\text{ref}}} \frac{\mathfrak{u}_{\text{ref}}^2}{u_{\text{ref}}^2}, \quad \mathfrak{v} = \frac{\mathfrak{u}_{\text{ref}}}{u_{\text{ref}} \Delta x} v. \quad (44)$$

where $\Delta x = l_{\text{ref}}/N$ is the lattice element's physical size. These relations also reveal how the simulation cost changes as we tune the parameters. For example, as shown in Eq. (29), the LBE solver uses a unit timestep size $\Delta\mathfrak{t} = 1$. Then,

according to the first relation in Eq. (44), a larger fluid velocity u_{ref} in physical unit demands a smaller physical timestep size Δt , as expected.

With these relations, the external force density $F = \rho \mathbf{a}$ can also be converted in lattice units:

$$\mathbb{F} = \rho \mathbf{a} = \frac{\rho}{\rho_{\text{ref}}} \frac{u_{\text{ref}}^2 \Delta x}{u_{\text{ref}}^2} \mathbf{a} = \frac{u_{\text{ref}}^2 \Delta x}{\rho_{\text{ref}} u_{\text{ref}}^2} F. \quad (45)$$

$$\mathbb{F} = \frac{u_{\text{ref}}^2 \Delta x}{\rho_{\text{ref}} u_{\text{ref}}^2} F. \quad (46)$$

From the conversion between physical and normalized lattice units, another important parameter is the number of timesteps needed per animation frame. Suppose the expected framerate is K . The number of simulation timesteps required per frame is

$$n_K = \frac{u_{\text{ref}}}{K u_{\text{ref}} \Delta x}. \quad (47)$$

When the fluid is coupled with moving solids, the solid object simulator often performs in physical units, not in lattice units. Therefore, to compute the penalty forces in the immersed boundary method, we first convert the fluid quantities in lattice units to those in physical units so that we can compare the fluid boundary velocity with the surface velocity of the solid body. After computing penalty forces, we rescale them to the normalized LBE scale in order to update velocities at the lattice nodes.

2.6 Setting Simulation Parameters

There are two typical routes to set simulation parameters.

Physical to LBE units. A natural way starts with user-specified physical parameters (such as ρ_{ref} , u_{ref} , and ν) and the lattice resolution N . From these physical parameters, the LBE parameters (ν), initial simulation values (ρ , $\mathbf{u}$, and $\mathbf{g}$) and the number of timesteps (n_K given a frame rate K) are all determined.

LBE to physical units. When tweaking the animation effects, we often directly specify the parameters in LBE units (such as u_{ref} , ν and n_K) and the grid resolution N . We must then convert them to physical units through

$$u_{\text{ref}} = u_{\text{ref}} \sqrt{\frac{K n_K \nu}{\nu}} \text{ and } \Delta x = \sqrt{\frac{\nu}{K n_K \nu}}, \quad (48)$$

from which we know the physical scale of the LBE simulation.

Notice that the viscosity ν is a fixed material parameter in physical units. Eq. (48) shows that when u_{ref} and the frame rate K are given, increasing the viscosity in lattice units decreases the simulated physical scale. However, we can tune the number of time steps n_K to match the same u_{ref} or Δx . For example, when decreasing $\nu \rightarrow \alpha \nu$ by a scale $\alpha < 1$, to keep the same u_{ref} one must set $n_K \rightarrow \alpha n_K$, but in this case Δx is reduced to $\Delta x \rightarrow \Delta x / \alpha$. On the contrary, to keep Δx unchanged when $\nu \rightarrow \alpha \nu$, we can select $n_K \rightarrow n_K / \alpha$, which in turn reduces to $u_{\text{ref}} \rightarrow u_{\text{ref}} / \alpha$.

3 Collision Modeling

In this chapter, we review a number of different collision models. As we discussed earlier, the LBE defined in Eq. (29) can be split into two processes. The first is a local collision or relaxation step that models the redistribution of particles caused by particle collisions,

$$f_i^*(\mathbf{x}, t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t). \quad (49)$$

Subsequently, a purely advective streaming step models the propagation of particles,

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = f_i^*(\mathbf{x}, t). \quad (50)$$

Rewriting the collision step modeled by Eq. 49 using vector notation as

$$\mathbf{f}^* = \mathbf{f} + \Omega(\mathbf{f}), \quad (51)$$

we can obtain a general description of the collision operation, where $\mathbf{f} = (f_0, f_1, \dots, f_{q-1})^T$ and Ω can be seen as a general collision operator. The term $\Omega(\mathbf{f})$ can be evaluated in a number of ways, of varying complexity and accuracy.

3.1 Single-Relaxation Time (SRT)

In the previous chapter, we introduced the BGK collision model,

$$\Omega_{\text{BGK}} = -\omega_v (\mathbf{f} - \mathbf{f}^{\text{eq}}),$$

where $\omega_v = 1/\tau = \frac{1}{3v+0.5}$ is the relaxation rate associated with kinematic viscosity v . f_i^{eq} is usually chosen as the second-order equilibrium distribution function, see Eq. (35). Since there is only one parameter in BGK model controlling the collision process, BGK model is also often called *Single-Relaxation-Time (SRT)* collision model.

While having only one relaxation time offers a simple and efficient implementation, it also suffers from the fact that all non-conserved modes decay at the same rate, which makes the SRT model exhibit pronounced instability for high Reynolds numbers. As the fluid viscosity decreases, the relaxation time τ approaches the stability limit of 0.5, leading to severe numerical stiffness. Because the SRT model forces all higher-order "ghost modes" to relax at the same rate as the physical shear stress, the system becomes highly sensitive to localized perturbations, often resulting in divergence or spurious oscillations.

3.2 Multiple-Relaxation Time (MRT)

In order to lift the main limitations of the SRT model, a collision model should include additional degrees of freedom, allowing different modes to decay at different rates. The idea is to introduce an invertible linear transformation matrix $\mathbf{M}$ that maps the distribution functions to moment space:

$$\mathbf{m} = \mathbf{M}\mathbf{f}.$$

Similarly, the equilibrium moments are defined as:

$$\mathbf{m}^{\text{eq}} = \mathbf{M}\mathbf{f}^{\text{eq}}.$$

The reason for mapping distribution functions to moment space is that the macroscopic quantities of interest, such as density ρ , flow momentum $\rho\mathbf{u}$, etc., are directly related to velocity moments of the distribution function $\mathbf{f}$ (as described in Eq. (30)). After transformation, we can represent the dynamics in a space where different modes (conserved and non-conserved quantities) can be separated. The collision process can then be expressed in moment space as

$$\mathbf{m}^* = \mathbf{m} - \mathbf{S}(\mathbf{m} - \mathbf{m}^{\text{eq}}),$$

where $\mathbf{S} = \text{diag}(\omega_0, \omega_1, \dots, \omega_{Q-1})$ is the relaxation matrix. Transforming back to distribution functions, the whole process is written as

$$\mathbf{f}^* = \mathbf{f} - \mathbf{M}^{-1}\mathbf{S}\mathbf{M}(\mathbf{f} - \mathbf{f}^{\text{eq}}), \quad (52)$$

which yields

$$\Omega_{\text{MRT}} = -\mathbf{M}^{-1} \mathbf{S} \mathbf{M} (\mathbf{f} - \mathbf{f}^{\text{eq}}).$$

This collision model is usually referred to as the *Multiple-Relaxation Time (MRT)* collision model [59]. The transformation matrix $\mathbf{M}$ in the MRT model can be defined in a general form as the projection of the distribution function onto a set of basis functions (ϕ_k) evaluated at the discrete velocities, i.e., the k -th row of $\mathbf{M}$ can be given by

$$\mathbf{M}_k = (\phi_k(\mathbf{c}_0), \phi_k(\mathbf{c}_1), \dots, \phi_k(\mathbf{c}_{q-1})),$$

leading to

$$m_k = \sum_{i=0}^8 \phi_k(\mathbf{c}_i) f_i.$$

Hence, constructing $\mathbf{M}$ reduces to selecting a set of linearly independent polynomial basis functions. Using the D2Q9 lattice for illustration, a set of moments that are commonly chosen is

$$(\rho, e, \epsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy}),$$

where ρ is the density, j_x and j_y are components of momentum-density components, q_x and q_y correspond to the energy flux components, e and ϵ correspond to the energy and the energy squared, p_{xx} and p_{xy} correspond to the diagonal and off-diagonal components of the stress tensor. The ordering of the moments is arbitrary; here, we follow the conventional ordering given by Lallemand and Luo [59]. The corresponding basis functions are defined as

$$\begin{aligned} \phi(\mathbf{c}) \in \{ & \phi^\rho, \phi^e, \phi^\epsilon, \phi^{j_x}, \phi^{q_x}, \phi^{j_y}, \phi^{q_y}, \phi^{p_{xx}}, \phi^{p_{xy}} \}, \\ & \phi^\rho = 1, \quad \phi^e = -4 + 3(c_x^2 + c_y^2), \\ & \phi^\epsilon = 4 - \frac{21}{2}(c_x^2 + c_y^2) + \frac{9}{2}(c_x^2 + c_y^2)^2, \\ & \phi^{j_x} = c_x, \quad \phi^{q_x} = \left(-5 + 3(c_x^2 + c_y^2)\right)c_x, \\ & \phi^{j_y} = c_y, \quad \phi^{q_y} = \left(-5 + 3(c_x^2 + c_y^2)\right)c_y, \\ & \phi^{p_{xx}} = c_x^2 - c_y^2, \quad \phi^{p_{xy}} = c_x c_y. \end{aligned}$$

The superscript of ϕ denotes the physical quantity represented by the corresponding moment. Such a construction of the transformation matrix is also called the *raw moment* basis. However, it is important to note that the choice of basis functions is not unique, and different sets of linearly independent polynomials can be used to construct $\mathbf{M}$, leading to different MRT formulations. Taking in the values of the discrete velocities $\mathbf{c}_i$ (the order is defined in Eq. (31)), we can obtain the transformation matrix

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}.$$

The relaxation matrix is then expressed as:

$$\mathbf{S} = \text{diag}(0, \omega_e, \omega_e, 0, \omega_q, 0, \omega_q, \omega_v, \omega_v),$$

Relaxation rates that are zero indicate that the moments are conserved (density and momentum). The value of ω_v remains the same as for the BGK model, and ω_e is related to the bulk viscosity ζ as

$$\zeta = c_s^2 \left(\frac{1}{\omega_e} - \frac{1}{2} \right).$$

The relaxation rates corresponding to higher-order non-hydrodynamic moments (ω_e and ω_q) do not directly influence the Navier-Stokes limit and can therefore be treated as free parameters. While we explained the MRT collision operator in 2D cases, in 3D cases like D3Q19 or D3Q27, there are even more free parameters. While a few methods provide guidance on tuning these parameters, how to choose them in an optimal way remains an open question.

3.3 Two-Relaxation Time (TRT)

The so-called *Two-Relaxation Time (TRT)* collision model was proposed to improve numerical stability and accuracy while avoiding the full complexity of the MRT formulation. The key idea of TRT is to decompose the distribution functions into symmetric (even) and antisymmetric (odd) components, using

$$\begin{aligned} f_i^+ &= \frac{f_i + f_{\bar{i}}}{2}, & f_i^- &= \frac{f_i - f_{\bar{i}}}{2}, \\ f_i^{\text{eq},+} &= \frac{f_i^{\text{eq}} + f_{\bar{i}}^{\text{eq}}}{2}, & f_i^{\text{eq},-} &= \frac{f_i^{\text{eq}} - f_{\bar{i}}^{\text{eq}}}{2}, \end{aligned}$$

with velocity $\mathbf{c}_{\bar{i}} = -\mathbf{c}_i$. and to relax these components independently using two distinct relaxation rates through

$$f_i^{*,+} = f_i^+ - \omega^+ (f_i^+ - f_i^{\text{eq},+}), \quad f_i^{*,-} = f_i^- - \omega^- (f_i^- - f_i^{\text{eq},-}),$$

where ω^+ controls viscosity and ω^- is a parameter that can be tuned to control numerical dissipation and boundary behavior. For the rest of the population (the discrete velocity $\mathbf{c} = \mathbf{0}$), $f^+ = f, f^- = 0$. The post-collision distribution function is then reconstructed as

$$f_i^* = f_i^{*,+} + f_i^{*,-}.$$

One might notice that the TRT model can be seen as a particular form of the MRT models when the MRT basis vectors ϕ corresponding to the non-conserved moments can be categorized into even (ϕ^+) and odd (ϕ^-) sub-bases that satisfy

$$\begin{cases} \phi_k(-\mathbf{c}) = \phi_k(\mathbf{c}), & \phi_k \in \phi^+, \\ \phi_k(-\mathbf{c}) = -\phi_k(\mathbf{c}), & \phi_k \in \phi^-. \end{cases}$$

For example, for the above-mentioned D2Q9 lattice model in Sec. 3.2, $\phi^+ = \{\phi^e, \phi^\epsilon, \phi^{p_{xx}}, \phi^{p_{xy}}\}$, $\phi^- = \{\phi^{q_x}, \phi^{q_y}\}$. So TRT model is equivalent to the MRT model when

$$\mathbf{S} = \text{diag}(0, \omega^+, \omega^+, 0, \omega^-, 0, \omega^-, \omega^+, \omega^+).$$

A distinctive feature of TRT is the introduction of a free parameter known as the magic parameter

$$\Lambda = \left(\frac{1}{\omega^+} - \frac{1}{2} \right) \left(\frac{1}{\omega^-} - \frac{1}{2} \right).$$

Since ω^+ relates to viscosity, by fixing Λ , one can solve for the other relaxation rate ω^- . Through mathematical derivation, it can be shown that higher-order truncation errors of the TRT model are related to parameter Λ [24, 39].

Also, the value of Λ can exhibit different behavior for different problems. For example, $\Lambda = 1/4$ is shown to be most stable [40]. $\Lambda = 1/12$ and $\Lambda = 1/6$ cancel the third-order and the fourth-order spatial error of steady state solutions, respectively [38].

3.4 Central moment and cumulant

We have introduced MRT model built with raw moments above. While this construction is natural and convenient, it has an important limitation: raw moments mix the effect of microscopic fluctuations with the effect of bulk advection. In other words, when the same local non-equilibrium state is observed in two frames moving with different uniform velocities, the corresponding raw moments are not identical. This issue is tantamount to a lack of *Galilean invariance*, namely the requirement that the physical behavior of the fluid should not depend on the observer's uniform translational motion. As a result, the relaxation of higher-order modes in raw-moment space may be contaminated by the local mean velocity, which introduces unnecessary coupling between physical transport and non-hydrodynamic degrees of freedom.

One way to reduce this coupling is to evaluate moments in the comoving frame of the fluid element, namely relative to the local velocity $\mathbf{u}$. This leads to *central moments* [34, 35]. The key modification is that the velocity is shifted for the polynomial basis to construct moments

$$\tilde{\mathbf{c}} = \mathbf{c} - \mathbf{u}.$$

Denoting the transformation from distribution functions to central moments by

$$\tilde{\mathbf{m}} = \tilde{\mathbf{M}}(\mathbf{f}),$$

the relaxation is performed in central moment space using

$$\tilde{\mathbf{m}}^* = \tilde{\mathbf{m}} - S(\tilde{\mathbf{m}} - \tilde{\mathbf{m}}^{\text{eq}}).$$

The corresponding post-collision distribution can then be written as

$$\mathbf{f}^* = \mathbf{f} - \tilde{\mathbf{M}}^{-1} S(\tilde{\mathbf{m}} - \tilde{\mathbf{m}}^{\text{eq}}),$$

which gives the collision operator in symbolic form

$$\Omega_{\text{CM}} = -\tilde{\mathbf{M}}^{-1} S \tilde{\mathbf{M}} (\mathbf{f} - \mathbf{f}^{\text{eq}}). \quad (53)$$

This model is called *cascaded* or *central moment (CM)* collision model. Performing relaxation in central moment space notably improves Galilean invariance and stability of simulation [1, 27, 78].

The *cumulant* collision model can be viewed as a further improvement based on CM collision model. Instead of relaxing central moments directly, it transforms the distribution function into *cumulants*, which are quantities derived from the logarithm of the moment-generating function. In the literature, the classical three-dimensional cumulant formulation is most commonly constructed on the D3Q27 lattice, for which the tensor-product velocity set is particularly well suited to the cumulant representation and its Galilean-invariance properties [36]. Therefore, we use the three-dimensional notation here for illustration. Symbolically, one may write

$$\kappa_{pqr} = \frac{\partial^{p+q+r}}{\partial \xi_x^p \partial \xi_y^q \partial \xi_z^r} \ln \left(\sum_i f_i e^{\xi_x(c_{ix}-u_x) + \xi_y(c_{iy}-u_y) + \xi_z(c_{iz}-u_z)} \right) \Big|_{\xi_x=\xi_y=\xi_z=0}.$$

Low-order cumulants coincide with familiar physical quantities, while higher-order cumulants measure genuinely independent deviations from equilibrium. If we denote the mapping from distribution functions to cumulants by

$$\boldsymbol{\kappa} = \mathcal{T}_{\kappa}(f),$$

then the post-collision distribution is reconstructed by

$$f^* = \mathcal{T}_{\kappa}^{-1}(\boldsymbol{\kappa} - \mathcal{S}_{\kappa}(\boldsymbol{\kappa} - \boldsymbol{\kappa}^{\text{eq}})),$$

and the corresponding collision operator can be written as

$$\Omega_K = -\mathcal{T}_{\kappa}^{-1} \mathcal{S}_{\kappa} \mathcal{T}_{\kappa}.$$

Because cumulants remove some of the statistical redundancy still present in moment descriptions, they further weaken the coupling among high-order non-equilibrium modes. This often results in better stability and lower numerical artifacts than both MRT and central-moment formulations, especially for under-resolved turbulent flows and simulations with very low viscosity [36].

3.5 Regularization before collision

The collision models reviewed above mainly differ in how the non-equilibrium part of the distribution is relaxed. A *regularized* approach starts from a slightly different observation: in practical computations, the distribution functions often contain high-order non-hydrodynamic components that are poorly represented on a finite lattice. These components do not contribute to the target Navier–Stokes dynamics, but after streaming, they may alias back into lower-order modes and trigger spurious oscillations or instability. The idea of regularization is therefore to filter the non-equilibrium distribution before collision, keeping only the part that is consistent with the resolved hydrodynamic subspace [61, 100].

To this end, one projects the non-equilibrium part of f^{neq} onto a truncated Hermite subspace, i.e.,

$$\begin{aligned} \tilde{f}_i^{\text{neq},(n)} &= w_i \sum_{n=2}^N \frac{1}{n! c_s^{2n}} \mathbf{a}^{\text{neq},(n)} : \mathbf{H}_i^{(n)}, \\ \mathbf{a}^{\text{neq},(n)} &= \sum_i \mathbf{H}_i^{(n)} f_i^{(1)} \approx \sum_i \mathbf{H}_i^{(n)} (f_i - f_i^{\text{eq}}). \end{aligned}$$

In the original regularization method, a projection up to order $N = 2$ is chosen. Later the model was improved by *Recursive Regularized (RR)* model [19, 77] to further filter the non-hydrodynamic contributions in higher-order terms. The pre-collision state is then replaced by

$$\tilde{f}_i = f_i^{\text{eq}} + \tilde{f}_i^{\text{neq}},$$

and the regularized collision step is written as

$$f^* = \tilde{f} + \Omega(\tilde{f}).$$

Since the non-equilibrium part is first filtered by the projection step, the regularized collision can be implemented with a simple BGK-type relaxation without sacrificing robustness. Hence, regularization can be interpreted as a combination of projection and relaxation: non-hydrodynamic components are first removed, and only the filtered non-equilibrium part is allowed to survive the collision step.

A further extension is the *Hybrid Recursive Regularized* (HRR [55]) approach. “Hybrid” refers to a hybrid reconstruction of the second-order non-equilibrium Hermite coefficient. More specifically, instead of taking $\mathbf{a}_1^{(2)}$ purely from projection

of the off-equilibrium distributions, HRR blends this projected value with a macroscopic estimate obtained from the strain-rate tensor via

$$\mathbf{a}_1^{(2),\text{HRR}} = \sigma \mathbf{a}_1^{(2),\text{proj}} + (1 - \sigma) \mathbf{a}_1^{(2),\text{macro}}, \quad 0 \leq \sigma \leq 1.$$

Here

$$\mathbf{a}_1^{(2),\text{proj}} = \sum_i \mathbf{H}_i^{(2)} (f_i - f_i^{\text{eq}}),$$

while the macroscopic estimate is computed from the Chapman–Enskog relation between the second-order non-equilibrium moment and the viscous stress, that is,

$$\begin{aligned} \mathbf{a}_1^{(2),\text{macro}} &= -2\rho c_s^2 \tau \mathbf{S}, \\ \mathbf{S} &= \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T). \end{aligned}$$

In component form, this reads

$$a_{1,\alpha\beta}^{(2),\text{macro}} = -\rho c_s^2 \tau (\partial_\beta u_\alpha + \partial_\alpha u_\beta).$$

Once the hybridized second-order coefficient is obtained, higher-order non-equilibrium Hermite coefficients are again reconstructed recursively, exactly in the spirit of RR, so that the filtered non-equilibrium distribution becomes

$$\tilde{f}_i^{\text{neq,HRR}} = w_i \sum_{n=2}^N \frac{1}{n! c_s^{2n}} \mathbf{a}_1^{(n),\text{HRR}} : \mathbf{H}_i^{(n)}. \quad (54)$$

The motivation behind this construction is that the projected value $\mathbf{a}_1^{(2),\text{proj}}$ is fully local but can be contaminated by lattice noise, while the macroscopic estimate $\mathbf{a}_1^{(2),\text{macro}}$ is more directly tied to the target Navier–Stokes stress tensor. Their blending therefore combines the locality of the kinetic description with the robustness of a macroscopic reconstruction. In practice, HRR often improves stability over standard regularization and RR, especially in under-resolved simulations and high-Reynolds-number flows.

4 Boundary Conditions

This chapter explores boundary treatments within the Lattice Boltzmann Method (LBM), a process that differs fundamentally from traditional macroscopic discretizations. In standard fluid solvers, boundary conditions are typically imposed by directly prescribing field values or derivatives at solid walls. In LBM, however, boundary conditions are integrated directly into the mesoscopic update. This necessity arises because the streaming step often leaves certain populations at boundary-adjacent fluid nodes unknown, as their trajectories originate from within the solid region where no fluid information exists.

Addressing these missing populations is the primary challenge of LBM boundary modeling. In sharp-interface methods, these unknown populations must be reconstructed explicitly to maintain a clean geometric boundary. Conversely, force-based coupling treats the wall’s influence as a distributed forcing term within the fluid domain rather than a direct, link-by-link population closure. Ultimately, wall treatment in LBM encompasses more than simple population reconstruction; it requires a strategic decision on how to close the mesoscopic update near solids. The goal is to ensure that macroscopic behaviors—such as no-slip conditions, moving-wall kinematics, or specific stress profiles—are recovered with high numerical accuracy and physical stability.

The mesoscopic nature of these treatments explains why boundary conditions are often the most critical component of a practical LBM solver. Inaccuracy at the wall is rarely a localized issue; it frequently propagates throughout the domain,

perturbing drag and pressure predictions or distorting momentum exchange on moving solids. In high-Reynolds-number simulations, a poor boundary closure can easily become the primary driver of numerical slip or instability. This challenge is compounded by the inherent tension between a rigid, regular lattice and complex, irregular geometries. Consequently, one must decide whether to prioritize geometric flexibility, sharp-interface accuracy, strict locality, or robustness on coarse grids, often while working with only a few cells across the boundary layer.

The existing literature can be viewed as a series of evolving responses to three fundamental questions. First, how should the influence of a solid be communicated to the fluid when the boundary is moving or geometrically complex? Second, if a sharp interface is required, how should missing populations be reconstructed on cut lattice links? Third, when the grid is too coarse to resolve near-wall physics, what additional modeling is necessary beyond a purely geometric condition?

This chapter follows that logical progression. We begin with force-based coupling, an approach favored for its natural handling of motion and topological changes. We then examine sharp-interface reconstruction, starting with the bounce-back family and moving toward interpolated and single-node variants, where increased boundary sharpness is traded for a more intricate geometric closure. Finally, we discuss near-wall modeling, which addresses a different fundamental limitation: representing unresolved wall stress when a wall-resolved simulation is computationally inaccessible.

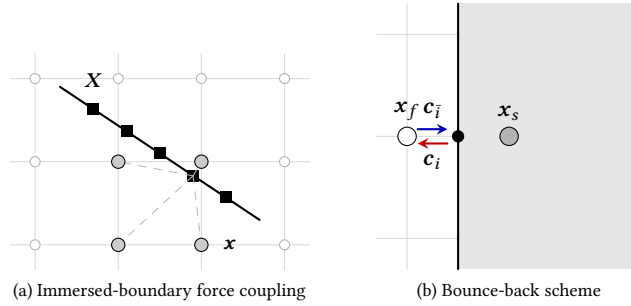


Fig. 5. The two main families of wall treatment discussed in this chapter. Force-based immersed-boundary coupling communicates wall action through a distributed forcing step, whereas bounce-back starts from a sharp-interface reflection rule on lattice links.

The distinctions between these methods should be viewed through an operational lens rather than as a strict taxonomy. *Force-based coupling* is most advantageous when dealing with moving, deforming, or topologically changing boundaries. By expressing the coupling via marker motion and force spreading, it avoids the computational burden of rebuilding a sharp geometric closure on every cut link at each timestep.

Conversely, *sharp-interface reconstruction* is the preferred approach when the primary goals are precise wall location, accurate momentum exchange, and boundary-layer fidelity. This method closes the missing populations directly at the interface where streaming encounters the wall, ensuring a crisp physical boundary. Finally, *near-wall modeling* serves a different purpose altogether; it is triggered not by the physical placement of the wall, but by the limitations of grid resolution when the simulation cannot fully resolve the underlying boundary-layer physics.

4.1 Force-based boundary coupling

Force-based boundary treatments do not try to determine the missing populations from wall geometry alone. Instead, they represent the action of the solid on the fluid through a body force density F that is spread over nearby lattice

nodes. This idea is closely related to immersed-boundary (IB) methods [82], as sketched in Figure 5a, and is attractive when the solid geometry is highly complex, strongly deforming, or moving in a way that makes an explicit link-by-link reconstruction cumbersome.

The wall is then described by a Lagrangian surface or set of markers embedded in the Eulerian fluid grid, and the coupling is achieved by exchanging information between these two representations.

At the continuous level, one may write the force spreading in the form

$$\mathbf{F}(\mathbf{x}, t) = \int_{\Gamma_w} \mathbf{F}_b(\mathbf{X}, t) \delta(\mathbf{x} - \mathbf{X}) d\mathbf{X}, \quad (55)$$

where Γ_w denotes the wall, $\mathbf{X}$ is a Lagrangian coordinate on the wall, and $\mathbf{F}_b$ is the force density required to enforce the desired boundary motion. In an LBM solver, this force enters the update through a forcing term, so that the stream-and-collide step becomes symbolically

$$f_i(\mathbf{x} + \mathbf{c}_i, t + 1) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) + \left(1 - \frac{1}{2\tau}\right) \Psi_i(\mathbf{F}). \quad (56)$$

The discrete forcing operator Ψ_i is chosen so that the force contributes to momentum while preserving mass, i.e.,

$$\sum_i \Psi_i = 0, \quad \sum_i \mathbf{c}_i \Psi_i = \mathbf{F}. \quad (57)$$

For example, the second-order forcing scheme of Guo et al. [45] writes

$$\Psi_i = w_i \left(\frac{\mathbf{c}_i - \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u}) \mathbf{c}_i}{c_s^4} \right) \cdot \mathbf{F}, \quad (58)$$

where τ is the relaxation time and the prefactor $(1 - 1/2\tau)$ is required to recover the correct macroscopic momentum equation.

The conceptual advantage of this family is flexibility. Because the wall is enforced through a distributed interaction rather than by identifying an exact cut position on each lattice link, the method naturally handles moving boundaries, flexible structures, and even topological changes. In particular, the wall velocity is already part of the forcing construction, so moving-boundary kinematics come “for free” once the Lagrangian markers are advanced. This has made immersed-boundary-style couplings popular in fluid–structure interaction, and recent graphics-oriented work has pushed this line further toward turbulent flow simulation and large-scale virtual testing facilities [65, 74, 75].

Force-based coupling methods achieve their geometric flexibility by replacing the direct mesoscopic enforcement of boundary conditions with a body force distributed across the Eulerian lattice. This substitution is necessary when the boundary does not coincide with lattice nodes, but it carries a structurally unavoidable cost: the forcing must be spread over a finite stencil using a kernel that approximates the Dirac delta, which smears the wall into a diffuse region of thickness proportional to the grid spacing. Crucially, this layer narrows only proportionally to the grid element size under refinement, while the bulk scheme converges quadratically with the grid element size; the mismatch in convergence rates means that pointwise accuracy near the interface consistently lags behind bulk accuracy, regardless of how fine the mesh becomes. The fidelity of the diffuse-layer approximation depends on three interdependent choices. The interpolation kernel governs the trade-off between spatial localization and stability: narrow kernels concentrate the forcing but risk oscillatory artifacts in the velocity field, while wider kernels suppress these at the cost of a thicker effective interface. Marker spacing must be commensurate with the lattice resolution to avoid gaps or redundancy in force coverage. The constraint enforcement strategy (whether direct forcing, feedback control, or a predictor-corrector

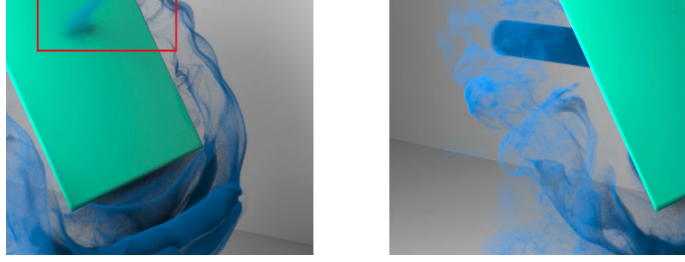


Fig. 6. **Why diffuse immersed-boundary forcing can fail on thin objects.** For a kinetic fluid simulation coupled with a thin plate, the diffuse-interface immersed-boundary treatment on the left produces visible leakage through the plate because force spreading affects both sides of the structure at once. The sharper coupling on the right avoids this failure mode by design and preserves the separation between the two sides of the plate.

scheme) determines how accurately the no-slip condition is imposed and how residual velocity errors accumulate across time steps. Two practical consequences are worth distinguishing. First, no-slip is enforced only at discrete marker positions, not continuously throughout the diffuse layer, so the boundary exhibits a finite effective slip that diminishes with refinement but persists as the leading near-wall error. Second, spatial accuracy near the wall degrades to first order regardless of the bulk discretization — a fundamental limitation of the regularized forcing approach that can only be mitigated by reintroducing local reflection rules, either through a hybrid scheme or by designing kernels that concentrate the force more aggressively near the actual boundary surface.

These drawbacks are most evident when simulating thin structures, such as plates or shells. If the support of the spreading kernel overlaps both sides of a thin geometry, the two sides are no longer numerically isolated. The force intended to enforce a boundary on one side "leaks" through to the other, contaminating the local flow. In practice, this manifests as leakage artifacts, a blurred momentum jump across the obstacle, and a significant loss of boundary sharpness—ironically, in the very scenarios where force-based methods are otherwise most attractive.

Recent generalized coupling methods in computer graphics, such as those proposed by [75] and [65], attempt to address these limitations. These approaches aim to retain the geometric versatility of IB-style coupling while achieving a crisper, more reliable interaction for thin moving structures. This specific trade-off—balancing ease of use with physical precision—serves as the primary motivation for the next family of methods. Rather than spreading the solid's influence over a local fluid neighborhood, these techniques focus on identifying the wall location with high precision. By doing so, they can explicitly reconstruct the specific populations that remain unknown after the streaming process, ensuring a sharper boundary representation.

4.2 Sharp-interface reconstruction on lattice links

This family begins with the simplest bounce-back rule and progresses toward increasingly refined closures that balance geometric accuracy with computational locality. It uses a classical LBM viewpoint, in which the wall is represented by reconstructing the populations that are unknown after streaming. Compared with force-based coupling, the philosophy is almost opposite: instead of smearing the solid effect into a body force, one first asks where the lattice links intersect the wall and then closes the missing incoming distributions directly on those cut links. This usually leads to a sharper effective boundary, better control of wall location, and a cleaner interpretation of momentum exchange, which is why reconstruction-based methods remain the default choice in high-fidelity single-phase LBM. Let Ω_s denote the solid region and let $\mathbf{x}_f$ be a fluid node adjacent to the wall. After collision, the known post-collision populations $f_i^*(\mathbf{x}_f, t)$

stream along the discrete directions $\mathbf{c}_i$. Whenever the departure point $\mathbf{x}_f - \mathbf{c}_i$ lies inside Ω_s , the incoming population along direction i is not available from the standard streaming step. The set of missing directions can thus be written as

$$\mathcal{M}(\mathbf{x}_f) = \{i \mid \mathbf{x}_f - \mathbf{c}_i \in \Omega_s\}. \quad (59)$$

Boundary reconstruction amounts to prescribing, for each $i \in \mathcal{M}(\mathbf{x}_f)$,

$$f_i(\mathbf{x}_f, t+1) = \mathcal{B}_i(\mathbf{f}^*(\mathbf{x}_f, t), \mathbf{f}^{\text{eq}}(\rho_w, \mathbf{u}_w), \Gamma_w), \quad (60)$$

where Γ_w summarizes the local wall geometry and $\mathbf{u}_w$ is the wall velocity. Here ρ_w denotes the density used to construct the wall-equilibrium input. Different wall treatments mainly differ in how much geometric information is used in $\mathcal{B}_i$, whether neighboring fluid nodes are consulted, and how accurately the resulting reconstruction places the effective wall location. From this point onward, the design space is best understood along two coupled axes. The first is geometric fidelity: how accurately does the reconstruction place the effective wall on a cut link? The second is stencil width: how much extra information must be gathered beyond the boundary-adjacent node itself? Much of the modern development of sharp-interface LBM can be read as an attempt to improve the first quantity without paying too much in the second.

At the family level, the contrast with force-based IB coupling is now direct. IB methods communicate the wall action through a distributed forcing step on nearby Eulerian nodes, whereas reconstruction-based methods close the problem directly on the missing lattice populations after streaming. The bounce-back family is the simplest expression of this sharp-interface viewpoint: instead of smearing the wall into a force density, it treats the wall as a reflection condition attached to discrete lattice links, as already previewed in Fig. 5b.

Bounce-back. The simplest and still most widely used reconstruction is called *bounce-back* [58]. Its physical interpretation is straightforward: populations that would have streamed into the solid are reflected back along the opposite lattice direction. If $\bar{i}$ denotes the direction opposite to i , then the stationary-wall half-way bounce-back rule can be written as

$$f_i(\mathbf{x}_f, t+1) = f_{\bar{i}}^*(\mathbf{x}_f, t), \quad i \in \mathcal{M}(\mathbf{x}_f). \quad (61)$$

For a moving wall with velocity $\mathbf{u}_w$, a momentum correction is added so that the reflected population carries the wall velocity [58],

$$f_i(\mathbf{x}_f, t+1) = f_{\bar{i}}^*(\mathbf{x}_f, t) + \frac{2w_i\rho(\mathbf{c}_i \cdot \mathbf{u}_w)}{c_s^2}, \quad (62)$$

where ρ is the local fluid density at $\mathbf{x}_f$. This rule is extremely attractive because it is local, easy to parallelize, and introduces almost no additional arithmetic. Moreover, when the effective wall lies halfway between a fluid node and a solid node, half-way bounce-back recovers no-slip with second-order accuracy for straight walls aligned with the lattice. Its main weakness is geometric rigidity. Relative to force-based methods, bounce-back gives a much sharper mesoscopic wall, but only by making a very specific assumption about where that wall is located. For curved or off-lattice boundaries, the actual wall position is generally not located exactly at the half-link, so the method suffers from a wall-location error that can noticeably affect drag, pressure drop, and boundary-layer resolution. This is also the point where moving-boundary treatment starts to differ conceptually from immersed-boundary coupling: the wall velocity is not built into the method automatically, so it must be reintroduced explicitly through the correction term in Eq. (62).

Interpolated bounce-back. Interpolated bounce-back should be read as the minimal geometric refinement of plain bounce-back rather than as a disconnected new family. Simple half-way bounce-back assumes that the effective wall

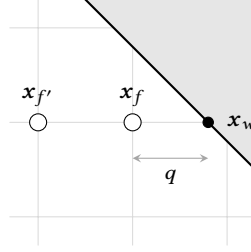


Fig. 7. Interpolated bounce-back as bounce-back plus wall-position information. Plain half-way bounce-back fixes the wall at the half-link; IBB instead measures the fractional wall position q from $\mathbf{x}_f$ to $\mathbf{x}_w$ on the same link and uses that geometric information to reconstruct the missing population more accurately.

always sits at the half-link, which is exactly why it is cheap and local but also why it incurs a wall-location error on curved or off-lattice boundaries. Interpolated bounce-back relaxes only that assumption: it keeps the same reflected-link picture, but replaces the fixed half-link wall position by the actual intersection point of the wall with the lattice link [10]. Let $q \in (0, 1]$ denote the fractional distance from the fluid node to the wall along the missing link, normalized by the lattice spacing in that direction, and let $\mathbf{x}_w$ denote the corresponding wall intersection point, as sketched in Figure 7. The piecewise form reflects a geometric necessity: when the wall sits very close to one end of the link, the reconstruction must adapt its stencil accordingly. When $q \leq \frac{1}{2}$, the wall lies close to the boundary-adjacent fluid node, so the reconstruction should resemble a reflected value corrected by information farther away from the wall. When $q > \frac{1}{2}$, the wall is closer to the opposite end of the link segment, and the reconstruction must instead lean more heavily on the local post-collision state to preserve consistency.

The explicit piecewise form proposed by Bouzidi et al. [10] is

$$f_i(\mathbf{x}_f, t+1) = \begin{cases} 2q f_i^*(\mathbf{x}_f, t) + (1-2q) f_i^*(\mathbf{x}_f + \mathbf{c}_i, t) & q \leq \frac{1}{2}, \\ \frac{1}{2q} f_i^*(\mathbf{x}_f, t) + \frac{2q-1}{2q} f_i^*(\mathbf{x}_f, t) & q > \frac{1}{2}, \end{cases} \quad (63)$$

where $\mathbf{x}_f + \mathbf{c}_i$ is the nearest fluid node on the same link beyond $\mathbf{x}_f$ (away from the wall), and $f_i^*(\mathbf{x}_f, t)$ is the post-collision value of the i -direction distribution at $\mathbf{x}_f$, which was reconstructed at the boundary step of the previous time step and subsequently updated by the collision operator. When $q = \frac{1}{2}$, both branches reduce to simple bounce-back, which makes the connection to the underlying reflection rule explicit. Equation (63) therefore highlights the basic mechanism: compared with simple bounce-back, the method uses the wall position q to place the same reflected-link closure more accurately in physical space. As a result, interpolated schemes significantly improve accuracy for curved and moving boundaries while preserving the intuitive link-based structure of bounce-back. The price is that the method is no longer purely local, since one typically needs information from an additional fluid node on the same link. This slightly increases implementation complexity and can complicate memory access on highly parallel architectures. So the trade-off has now shifted: bounce-back was simple but geometrically crude, whereas IBB is more accurate but begins to erode the strict locality that makes LBM so attractive on GPUs.

Single-node treatments. More recent boundary schemes aim to recover curved-wall accuracy while remaining as local as possible. Their common idea is to reconstruct the missing populations using only information available at the boundary-adjacent node itself, together with compact geometric information about the wall. Instead of tracing farther along the link to obtain an auxiliary population, these methods enforce moment constraints directly at one node

and incorporate the wall location through a correction of the equilibrium and non-equilibrium parts [36, 88, 102]. In symbolic form, such a reconstruction may be written as

$$f_i(\mathbf{x}_f, t + 1) = f_i^{\text{eq}}(\rho_b, \mathbf{u}_w) + \Phi_i(\mathbf{f}^*(\mathbf{x}_f, t), q, \mathbf{u}_w), \quad (64)$$

where ρ_b is a locally reconstructed density and Φ_i denotes a closure chosen so that the resulting populations satisfy the desired accuracy and consistency constraints.

The main advantage of this family is locality. Because all information is taken from the same lattice node, the method fits naturally into cache-friendly or GPU-friendly implementations and avoids the wider stencil required by classical interpolation. At the same time, a carefully designed closure can retain second-order accuracy for curved walls and offer better robustness at high Reynolds number than basic bounce-back. This improvement is particularly important when the collision model already uses higher-order moments, because the wall treatment should not become the dominant source of anisotropy or numerical slip. The main caveat is that the closure is more subtle than in bounce-back: one must ensure that the reconstructed populations satisfy the correct low-order moments while damping spurious non-hydrodynamic components. Even so, single-node methods have become increasingly attractive in modern high-performance LBM solvers because they balance geometric fidelity, numerical stability, and strict locality. This trend is particularly natural when the collision model already operates in moment variables, as in many regularized, central-moment, or cumulant formulations discussed in Chapter 3. In that setting, the boundary closure is no longer an isolated geometric trick: it becomes part of the same moment-consistent design philosophy as the collision operator itself. This is also why recent GPU-oriented solvers such as HOME-LBM emphasize local single-node coupling as a first-class ingredient rather than a secondary implementation detail; see Chapter 6 and in particular Sec. 6.2.3 for one concrete realization.

For practical use, the comparison is simpler to state in words than in a table. Bounce-back remains the most local and cheapest closure, but it achieves this simplicity by fixing the effective wall at the half-link and therefore accepting a geometric wall-location error on curved or off-lattice boundaries. Interpolated bounce-back corrects that wall position by using the cut-link location explicitly, but it pays for this gain with a wider stencil and less regular memory access. Single-node treatments try to recover the same curved-wall accuracy while staying local, which is why they are attractive in modern GPU solvers, although their closure design is more delicate than the classical reflection rule.

Overall, reconstruction-based wall treatments can be viewed as a sequence of increasingly refined answers to the same question posed by Eq. (60). Bounce-back offers the most economical closure, interpolated bounce-back corrects the effective wall position by using off-node geometric information, and single-node methods attempt to recover the same geometric fidelity without sacrificing locality. In other words, the design axis has moved from “force vs. sharp wall” to a more delicate balance between geometric accuracy and stencil width. Which option is best depends on the application: simple, axis-aligned rigid walls often favor bounce-back, while curved moving boundaries and high-Reynolds-number simulations usually justify the additional complexity of interpolated or single-node formulations. For moving boundaries, the comparison with immersed-boundary coupling remains instructive. IB methods absorb wall motion naturally through the forcing term, whereas reconstruction-based methods must encode that motion explicitly in the reflected or reconstructed populations. The reward is a sharper and more controllable wall representation; the cost is that geometry handling and boundary closure remain tightly coupled at the discrete level.

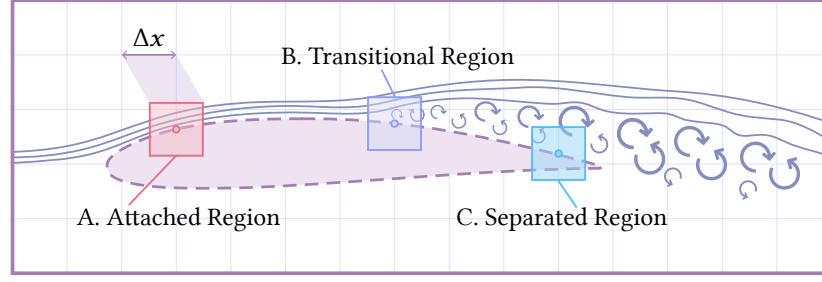


Fig. 8. **Under-resolved boundary layers motivate near-wall modeling.** When the turbulent boundary layer is thinner than the effective grid filter, the wall-adjacent velocity seen by the solver is already a filtered quantity rather than a resolved no-slip state. In that regime, improving the geometric wall placement alone is not enough; an additional near-wall closure is needed to recover the correct stress transfer.

4.3 Accuracy limits and near-wall modeling

The wall reconstructions above determine how the solid boundary is represented geometrically and how the no-slip or moving-wall condition is imposed at the mesoscopic level. However, in high-Reynolds-number flows the lattice spacing is often too coarse to resolve the full near-wall structure, especially the steep velocity gradient inside the boundary layer. In this regime, even an accurate geometric boundary condition is not sufficient by itself, because the missing physics is no longer only the wall location but also the unresolved stress transfer close to the wall. This is the regime where many of the difficulties practitioners complain about as “boundary-condition problems” are in fact resolution problems. The lattice may capture the global geometry, yet still fail to place enough nodes across the near-wall region where velocity gradients are steep. As a result, the wall-normal velocity gradient implied by the first off-wall node is severely underestimated, and the wall stress inferred from a purely geometric reconstruction becomes systematically biased. Put differently, there is a point beyond which improving the wall location alone yields diminishing returns. Once the first few off-wall cells already represent a filtered average over an unresolved boundary layer, even a perfectly placed no-slip wall does not restore the correct near-wall stress by itself. The remaining error is no longer geometric; it is a missing-physics error, and it must be addressed by an additional closure.

In traditional CFD practice, near-wall modeling is built on the experimental observation that the time-averaged velocity profile near a turbulent wall follows a universal relation when expressed in wall units: the normalized wall-distance $y^+ = yu_\tau/\nu$ versus the normalized tangential velocity $u^+ = u/u_\tau$ (where u_τ is the friction velocity, u is the local tangential velocity, and ν is the kinematic viscosity). This universality, well documented in boundary-layer textbooks, allows the near-wall region to be divided into three layers: a viscous sublayer where viscous stress dominates, a buffer layer where viscous and turbulent effects compete, and a logarithmic layer where the velocity obeys the log-law. The most widely used closure strategy is the wall-stress model, which places the first off-wall grid node inside the logarithmic region, typically at $y^+ \in [30, 300]$, and then uses the log-law relation to recover the wall shear stress from the resolved velocity at that node. This approach works well in finite-volume or finite-element solvers that operate on body-fitted meshes, because the mesh can be designed to maintain a nearly uniform wall distance for the first layer of nodes. Such control is essential: if the first node falls outside the log-law region, the wall-stress model loses its theoretical foundation and accuracy degrades sharply. Moreover, these models typically assume quasi-steady attached flow, which limits their applicability in separated or highly transient regions.

For lattice Boltzmann solvers, however, the situation is considerably more difficult. The regular Cartesian grid that makes LBM efficient and easy to parallelize also means that, for general geometries, the first off-wall nodes cannot be aligned with a constant y^+ layer. Instead, as the wall geometry varies, these nodes fall inconsistently among the viscous sublayer, the buffer layer, and the logarithmic layer; in extreme cases they may even reside inside or outside the turbulent boundary layer altogether. This inconsistency undermines the reliability of any wall-law correction that assumes a fixed flow regime at the first node. Several workarounds have been explored in the LBM literature: eliminating nodes very close to the wall to avoid the viscous sublayer, shifting the effective boundary outward to target a specific y^+ range, or enforcing the wall-stress model as a strict constraint during the bounce-back reconstruction step. Yet each of these strategies introduces its own difficulties. Removing or relocating near-wall nodes complicates the implementation significantly, especially for moving boundaries or dynamic fluid-solid interactions, while strictly enforcing a wall-law condition on individual lattice links can lead to numerical instability when the local geometry or flow state deviates from the expected attached-flow profile. As a result, traditional wall-stress approaches that work smoothly on body-fitted meshes become fragile or cumbersome when translated directly into an LBM framework.

Given these challenges, near-wall models for LBM need to adapt or extend standard wall-stress closures rather than simply importing them directly from body-fitted solvers. Rather than replacing the geometric wall treatment, they complement it by injecting additional information about the wall stress or the near-wall velocity profile into the mesoscopic update. In solver terms, the geometric boundary condition still decides where the wall is and which incoming populations must be reconstructed, while the near-wall model supplies the unresolved stress-transfer information that those populations cannot recover on a coarse grid by geometry alone. The key idea is not to abandon cut-cell or bounce-back reconstruction, but to augment it with a closure that distinguishes between flow regimes near the wall. For attached flows, the main difficulty is that the filtered wall-adjacent velocity remains too large on a coarse LES grid, so the model should reconstruct a more realistic near-wall shear. For separated flows, by contrast, the wall-normal transport and recirculation dominate, so the same aggressive wall-law correction can over-constrain the dynamics and must be damped. Recent LBM work has therefore explored parametric and hybrid mesoscopic closures that adapt the near-wall correction to the local boundary-layer state while preserving the regular lattice structure and local character of kinetic solvers. These approaches avoid the multiple-neighbor stencils, local equation solves, or time-history storage that would complicate the implementation, though at the cost of requiring careful calibration of model parameters [71, 72]. Representative application-scale results from this line of work are shown in Figure 9.

In the context of these notes, the key message is that wall reconstruction and near-wall modeling play different roles: the former determines how the boundary is imposed geometrically, while the latter compensates for unresolved near-wall physics when the grid is too coarse to resolve the boundary layer directly. This distinction matters in practice because it changes what kind of promise a method can make. A better geometric reconstruction may move the effective wall from the wrong place to the right place; a wall model, by contrast, tries to repair the stress transfer that would still be wrong even if the geometry were represented perfectly. This is precisely why recent near-wall LBM work is important for industrial applications. Once the wall stress is modeled rather than fully resolved, one can obtain meaningful aerodynamic predictions on large vehicle- or aircraft-scale geometries at resolutions that are compatible with present GPU memory budgets [71, 72]. In that sense, near-wall treatment is not a cosmetic refinement added after the “real” boundary condition is settled. It is often the ingredient that turns a numerically elegant wall treatment into a practically useful aerodynamic tool. For modern virtual wind-tunnel scenarios, the question is no longer only whether the wall is geometrically well placed, but whether the solver can still return the right large-scale pressure pattern, streamline structure, and integral force trends at an affordable resolution. This is exactly the regime in which

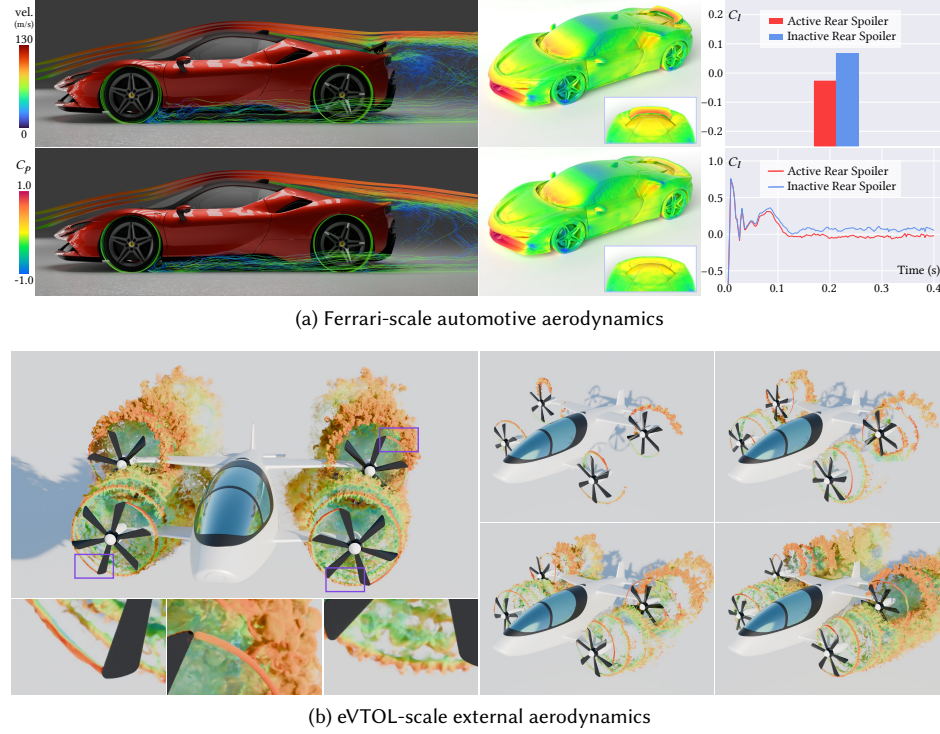


Fig. 9. **Near-wall modeling as an application enabler.** These examples illustrate the real payoff of near-wall LBM closures: they make it possible to run large-scale aerodynamic analyses on industrial geometries without fully resolving the near-wall turbulence everywhere. In that setting, the value of the wall model is not abstract accuracy at one cut link, but the ability to recover meaningful pressure patterns, flow structures, and force trends at an affordable grid resolution [71, 72].

recent hybrid LBM wall models become valuable: they allow one to keep the regular-grid efficiency of kinetic solvers while extending them toward the industrial prediction tasks that motivate many large-scale applications today.

Practical takeaway. From a practical standpoint, these boundary treatments should not be viewed as mutually exclusive competitors; rather, they address different bottlenecks. First, force-based coupling remains attractive when the geometry is moving, deforming, topologically changing, or otherwise inconvenient to represent through explicit cut links. Second, sharp-interface reconstruction is preferred when wall location, drag prediction, and clean momentum exchange are the primary concerns, with bounce-back, IBB, and single-node methods offering different compromises between simplicity, accuracy, and locality. Third, near-wall modeling enters as an additional layer when the geometry may be adequately represented but the boundary-layer physics is not adequately resolved. This regime is particularly challenging for Cartesian-grid LBM solvers, where mesoscopic closures must compensate for both under-resolution and the lack of y^+ -aligned first nodes. Taken together, the modern LBM view of boundary treatment is therefore not just about imposing no-slip: it is about choosing the right combination of geometric representation, mesoscopic closure, and near-wall physics for the target application.

5 Grid Refinement

Although the use of a regular Cartesian lattice is one of the main reasons why LBM is simple, efficient, and highly parallelizable, its requirement for uniform spatial resolution also introduces a major limitation. In many practical simulations, the flow features that require fine resolution are strongly localized, such as thin boundary layers near solid walls. If the entire domain is discretized using the finest spacing required to resolve these local structures, both memory cost and computational time quickly become prohibitive. This issue becomes even more severe in high-fidelity simulations, such as wind-tunnel cases, where the region of interest must be placed far from the domain boundaries, typically more than ten object lengths away, to reduce artificial boundary effects.

Grid refinement is introduced to resolve this tension. The goal is to retain fine lattices only where accuracy truly requires them, while using coarser lattices in the rest of the domain to reduce overall cost. Achieving this goal in LBM requires careful design of the coupling between different resolution levels, especially at the interfaces where fine and coarse grids meet. Particle distribution functions must be consistently transferred across different spatial and temporal resolutions at refinement interfaces while preserving conservation, maintaining the correct level of non-equilibrium information, and avoiding spurious reflections or excessive dissipation.

Broadly speaking, existing LBM refinement methods can be grouped into two main families: *node-based* (or *vertex-centered*) and *volume-based* (or *cell-centered*) methods. Node-based methods usually rely on nested lattice nodes shared across refinement levels, and the coupling is achieved by interpolating distribution functions at the shared nodes. Volume-based methods instead interpret refinement as subdividing coarse cells into fine ones, and information is exchanged more directly across coarse–fine interfaces. Building on these two families, a number of techniques have been proposed to reduce transition errors and improve robustness near level boundaries, including compact interpolation [41], directional splitting [37], and direct coupling [4]. In the remainder of this chapter, we first introduce the basic ideas behind these refinement frameworks, then discuss the two representative interface treatments.

5.1 Basic concepts

Before introducing specific grid-refinement algorithms, it is helpful to summarize a few basic concepts that appear in almost all multiresolution LBM formulations. We consider two neighboring refinement levels, denoted by a coarse level c and a fine level f , and a spatial refinement ratio can be defined as

$$r = \frac{\Delta x^{(c)}}{\Delta x^{(f)}}. \quad (65)$$

Once the spatial refinement ratio is fixed, the temporal refinement ratio must still be determined. In LBM, this is usually done through either diffusive or convective scaling. Diffusive scaling assumes

$$\Delta t \propto \Delta x^2,$$

whereas convective scaling assumes

$$\Delta t \propto \Delta x.$$

Diffusive scaling is more consistent with the incompressible limit of LBM and can help reduce compressibility-related errors, while convective scaling is more efficient and easier to implement. In this chapter, we restrict our discussion to convective scaling, which is a more popular choice in LBM. Thus, the relation between spatial and temporal scale is

$$\frac{\Delta t^{(c)}}{\Delta t^{(f)}} = \frac{\Delta x^{(c)}}{\Delta x^{(f)}} = r. \quad (66)$$

In this chapter, we only cover the case $r = 2$, for which the grids can be aligned in both space and time. Also, it is a rather balanced refinement ratio, making the difference across the interface between the coarse and fine grids not too large, while the number of intermediate refinement levels remains limited. For the same reasons, most LBM grid-refinement methods are formulated based on $r = 2$. We note that this choice is not mandatory and the refinement ratio can even be non-integer (as shown in Li et al. [63]), but such cases are beyond the scope of this chapter.

Rescaling of physical quantities. The change of Δx and Δt across different levels naturally raises a question, that is: how should physical quantities be rescaled between levels? We can find this relationship through the conversion from lattice units to physical units. Let a quantity with superscript (ℓ) denote its representation in lattice units on level $\ell \in \{c, f\}$. From Eq. (41), we found that density ρ remains unchanged across levels,

$$\rho^{(f)} = \rho^{(c)}, \quad (67)$$

while physical velocity and kinematic viscosity are represented on level ℓ as

$$\mathbf{u}^{\text{phys}} = \frac{\Delta x_\ell}{\Delta t_\ell} \mathbf{u}^{(\ell)}, \quad (68)$$

$$\nu^{\text{phys}} = \frac{\Delta x_\ell^2}{\Delta t_\ell} \nu^{(\ell)}. \quad (69)$$

With convective scaling (Eq. (66)), velocity is unchanged across levels,

$$\mathbf{u}^{(f)} = \mathbf{u}^{(c)}, \quad (70)$$

whereas the lattice viscosity must be rescaled as

$$\nu^{(f)} = r \nu^{(c)}. \quad (71)$$

As a consequence, the relaxation time of different levels is different,

$$\nu^{(\ell)} = c_s^2 \left(\tau^{(\ell)} - \frac{1}{2} \right). \quad (72)$$

This difference in lattice relaxation parameters also means that distribution functions cannot be copied directly between two refinement levels. The equilibrium part is determined only by the macroscopic variables and can be rebuilt locally once ρ and $\mathbf{u}$ are known on the target level. The non-equilibrium part, however, represents viscous stress in lattice units and therefore depends on the local time step, grid spacing, and relaxation time. For this reason, grid-refinement algorithms usually transfer distributions in the form

$$f_i^{(\ell)} = f_i^{\text{eq}}(\rho, \mathbf{u}) + f_i^{\text{neq},(\ell)}, \quad (73)$$

and apply a rescaling only to f_i^{neq} . A coarse-to-fine transfer can be written as

$$f_i^{\text{neq},(f)} = \alpha f_i^{\text{neq},(c)}, \quad (74)$$

where α is the rescaling coefficient. Based on Eq. (54), we can obtain that at the leading order

$$\begin{aligned} f_i^{\text{neq},(\ell)} &\approx f_i^{(1)} \approx \frac{w_i}{2c_s^4} \mathbf{a}_1^{(2),(\ell)} : \mathbf{H}_i^{(2)} \\ &= -\frac{\rho w_i \tau}{c_s^2} \mathbf{S}^{(\ell)} : \mathbf{H}_i^{(2)}. \end{aligned}$$

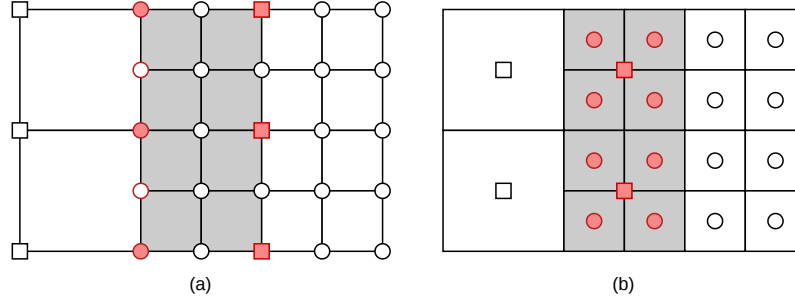


Fig. 10. **Schematic representation of grid layout in node-based approach (a) and volume-based approach (b).** The gray-shaded background indicates the overlapping region between the fine and coarse grids. Black open circle ($\circ$) denotes regular fine nodes, red solid circle ($\bullet$) denotes interface fine nodes that have collocated coarse nodes, red open circle ($\circ$) denotes interface fine nodes without collocated coarse nodes, black open square ($\square$) denotes regular coarse nodes, red solid square ($\blacksquare$) denotes interface coarse nodes with collocated fine nodes.

Under convective scaling, the scaling of lattice-unit velocity gradient is

$$S^{(f)} = \frac{\Delta t^{(f)}}{\Delta t^{(c)}} S^{(c)} = \frac{1}{r} S^{(c)}. \quad (75)$$

Therefore,

$$\alpha = \frac{\tau^{(f)}}{r\tau^{(c)}},$$

$$f_i^{\text{neq},(f)} = \frac{\tau^{(f)}}{r\tau^{(c)}} f_i^{\text{neq},(c)}.$$

For the refinement ratio used in this chapter, $r = 2$, this reduces to $\alpha = \tau^{(f)} / (2\tau^{(c)})$.

5.2 Node-based methods

Node-based methods, pioneered by Filippova and Hänel [30], generally require an overlapping region between two grids that needs communication. Therefore, there will be two interfaces, which is a fine-to-coarse interface and a coarse-to-fine interface. Both interfaces have collocated nodes, as illustrated in Fig. 10. The key part of this method lies in how to obtain the missing information at these two interfaces.

For collocated nodes, the information can be transferred without interpolation, resulting in the following coarse-to-fine and fine-to-coarse relationships, respectively,

$$f_i^{(f)} = f_i^{\text{eq}} + \frac{\tau^{(f)}}{2\tau^{(c)}} f_i^{\text{neq},(c)}, \quad (76)$$

$$f_i^{(c)} = f_i^{\text{eq}} + \frac{2\tau^{(c)}}{\tau^{(f)}} f_i^{\text{neq},(f)}. \quad (77)$$

This is sufficient for fine-to-coarse coupling, since all coarse grid nodes that require missing information have corresponding collocated nodes in fine grid. However, for coarse-to-fine coupling, the hanging nodes in the fine grid need spatial interpolation.

Another point worth noting is that, due to the difference in $\Delta t^{(c)}$ and $\Delta t^{(f)}$, When the coarse grid reaches $t + \Delta t^{(c)}$, the fine grid can only first advance to $t + \Delta t^{(f)} = t + 1/2\Delta t^{(c)}$. Therefore, a temporal interpolation is also required for the fine grid,

$$f_i^{(c)}(\mathbf{x}, t + \Delta t^{(f)}) = \frac{1}{2} \left(f_i^{(c)}(\mathbf{x}, t + \Delta t^{(c)}) + f_i^{(c)}(\mathbf{x}, t) \right). \quad (78)$$

The complete algorithm of node-based grid coupling is summarized in Algorithm 1.

Algorithm 1: Node-based grid coupling.

Input: Distribution functions of coarse grid $f_i^{(c)}$ and fine grid $f_i^{(f)}$ at time t .

- 1: Collision of all coarse and fine nodes.
 - 2: Streaming of all coarse and fine nodes, except for the nodes on the interfaces.
 - 3: Reconstruct $f_i^{(f)}(\mathbf{x}, t + \Delta t^{(f)})$ of collocated interface nodes on fine grid through temporal interpolation (Eq. (78)) and rescaling (Eq. (76)).
 - 4: Reconstruct $f_i^{(f)}(\mathbf{x}, t + \Delta t^{(f)})$ of hanging interface nodes on fine grid through spatial interpolation of neighboring collocated interface nodes. After this step, all nodes of fine grids reach $t + \Delta t^{(f)}$.
 - 5: Collision of all fine nodes.
 - 6: Streaming of all fine nodes, except for the nodes on the interfaces.
 - 7: Reconstruct $f_i^{(f)}(\mathbf{x}, t + 2\Delta t^{(f)})$ of collocated interface nodes on fine grid from $f_i^{(c)}(\mathbf{x}, t + \Delta t^{(c)})$ with rescaling (Eq. (76)).
 - 8: Reconstruct $f_i^{(f)}(\mathbf{x}, t + 2\Delta t^{(f)})$ of hanging interface nodes on the fine grid through spatial interpolation of neighboring collocated interface nodes. After this step, all nodes of fine grids reach $t + 2\Delta t^{(f)} = t + \Delta t^{(c)}$.
 - 9: Reconstruct $f_i^{(c)}(\mathbf{x}, t + \Delta t^{(c)})$ of interface nodes on coarse grid with rescaling (Eq. (77)). After this step, all nodes of coarse grids reach $t + \Delta t^{(c)}$.
 - 10: Go to step 1 for next iteration.
-

5.3 Volume-based methods

Volume-based methods [16, 83] adopt a different grid layout and logically store all quantities at the center of grid cells instead of grid nodes. From this viewpoint, refinement means that one coarse cell is replaced by several fine cells that together occupy the same volume. This geometric reinterpretation leads to an important conceptual difference from node-based approaches. In node-based refinement, the coupling is naturally phrased in terms of reconstructing missing data on shared nodes. In volume-based refinement, by contrast, one thinks more directly in terms of transport between adjacent control volumes. When a particle population streams out of a coarse cell toward a refined region, its contribution must be distributed to the fine cells on the other side of the interface, and vice versa. In this way, volume-based methods can often be designed to obey conservation of mass and momentum, while this is often violated in node-based refinement, since temporal and spatial interpolation is inevitably involved.

In practice, a volume-based update proceeds as follows (see Fig. 11 for schematic representation). Initially, all nodes are at post-streaming state (Fig. 11 (1)). At the beginning, all nodes perform collision except for fine interface nodes (Fig. 11 (2)). Then distributions of coarse interface nodes are redistributed onto fine interface nodes, which is often called *explosion* (Fig. 11 (3)). The most basic implementation is to homogeneously distribute post-collision distributions f_i^* from coarse cells to fine cells as in

$$f_i^{*,(f)}(\mathbf{x}_f, t) = f_i^{*,(c)}(\mathbf{x}_c, t), \quad (79)$$

where $\mathbf{x}_c$ is a coarse interface node and $\mathbf{x}_f$ are fine nodes that occupy part of the same volume of $\mathbf{x}_c$.

The next step is to perform the streaming step on both the coarse and fine grids. In this process, distributions originating from the coarse grid are propagated to the fine grid without any modification, and vice versa (Fig. 11 (4)).

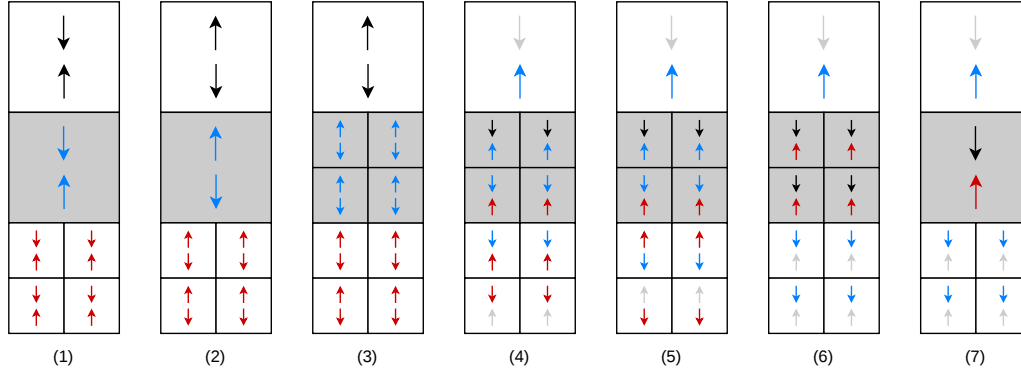


Fig. 11. **Schematic representation of coupling between coarse and fine grids in volume-based grid refinement.** Subfigures (1)–(7) illustrate the temporal evolution of the coupling procedure. Arrows of different colors denote distribution functions associated with different types of nodes. The gray-shaded background indicates the overlapping region between the fine and coarse grids. Arrows pointing toward the cell center represent post-streaming distribution functions, whereas arrows pointing outward from the center represent post-collision distribution functions.

Thus, this is essentially a communication procedure between the two grid levels. Next, perform another collision-and-streaming step on all fine nodes. Collision is performed on regular fine nodes only (that is, without the interface nodes, see Fig. 11 (5)) while streaming is performed on all fine nodes (Fig. 11 (6)).

The final step is to redistribute f_i from fine to coarse grid cells, which is often called *coalescence* (Fig. 11 (7)). For D spatial dimensions, this reads

$$f_i^{(c)}(\mathbf{x}_c, t) = \frac{1}{2^D} \sum f_i^{(f)}(\mathbf{x}_f, t). \quad (80)$$

The complete algorithm of volume-based grid coupling is summarized in Algorithm 2.

Algorithm 2: Volume-based grid coupling.

Input: Distribution functions of coarse grid $f_i^{(c)}$ and fine grid $f_i^{(f)}$ at time t .

- 1: Collision of all coarse and fine nodes, except for fine interface nodes.
 - 2: Perform coarse-to-fine coupling (explosion) according to Eq. (79).
 - 3: Streaming on all coarse and fine nodes. After this step, coarse grid reaches $t + \Delta t^{(c)}$ and fine grid reaches $t + \Delta t^{(f)}$.
 - 4: Collision of all regular fine nodes.
 - 5: Streaming of all fine nodes. After this step, fine grid reaches $t + 2\Delta t^{(f)} = t + \Delta t^{(c)}$.
 - 6: Perform fine-to-coarse coupling (coalescence) according to Eq. (80). After this step, all nodes of coarse grids reach $t + \Delta t^{(c)}$.
 - 7: Go to step 1 for next iteration.
-

6 Moment Representation

This chapter introduces the HOME-LBM formulation [68], a state-of-the-art moment-based LBM that encodes the distribution functions using only three velocity moments per grid node to significantly reduce memory and improve GPU efficiency, while retaining the accuracy needed for turbulent-flow simulation. We first present a high order distribution construction based on a higher-order Hermite reconstruction to recover discrete distributions for streaming. We then introduce a moment-based high-order collision model that achieves high accuracy via efficient and simple closed-form moment updates. Next, we describe a single-node boundary treatment for robust, local fluid–solid coupling

on cut links. Finally, we discuss how the HOME-LBM pipeline can be extended from single-phase settings to two-phase and more general n -phase flows.

6.1 Towards lightweight LBM

However, all single-phase turbulent fluid simulations in computer graphics thus far need to store two distribution functions for their temporal update and the first two macroscopic quantities from Eq. (30), leading to at least 58 variables per node — without considering extra temporary memory used, for instance, for the projection from distribution to moment spaces in the evaluation of the collision terms. A moment representation for LBM (MR-LBM [29, 42, 91]) has recently been formulated. It requires far fewer variables per node, thus relaxing memory requirements at least threefold compared to LBM solvers in graphics such as [64], seemingly ensuring its applicability to consumer GPUs.

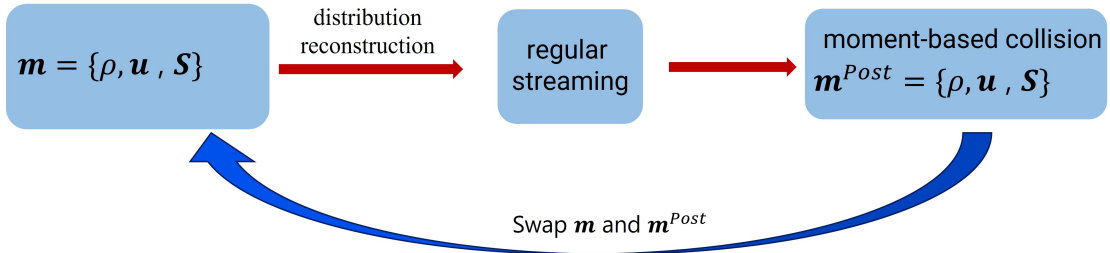


Fig. 12. MR-LBM

The MR-LBM consists of three steps: (a) from the three stored moments $\rho, \rho u, \rho S$, a distribution function is reconstructed via (we use Einstein notation to indicate tensor contraction):

$$f_i(\mathbf{x}, t) = \rho(\mathbf{x} - \mathbf{c}_i, t) w_i \left(1 + 3\mathbf{u}(\mathbf{x} - \mathbf{c}_i, t) \cdot \mathbf{c}_i + \frac{1}{2c_s^4} S_{\alpha\beta}(\mathbf{x} - \mathbf{c}_i, t) H_{\alpha\beta}^{[2]}(\mathbf{c}_i) \right), \quad (81)$$

where w_i are lattice weights and $\mathbf{H}^{[2]} = \{H_{\alpha\beta}^{[2]}\}_{\alpha,\beta}$ represents the Hermite polynomials of order 2; (b) the reconstructed distribution function is then streamed using Eq. (34); (c) the resulting distribution is converted back to its three moments using Eq. (30), and the BGK collision model and external forces are incorporated (effectively replacing Eq. (33)) by simply updating the current moments through:

$$\rho u_\alpha(\mathbf{x}, t+1) = \rho u_\alpha(\mathbf{x}, t) + \left(1 - \frac{1}{2\tau}\right) F_\alpha, \quad (82)$$

$$\rho S_{\alpha\beta}(\mathbf{x}, t+1) = \left(1 - \frac{1}{\tau}\right) \rho S_{\alpha\beta}(\mathbf{x}, t) + \frac{1}{\tau} \rho u_\alpha u_\beta(\mathbf{x}, t) + \left(1 - \frac{1}{2\tau}\right) (F_\alpha u_\beta + F_\beta u_\alpha)(\mathbf{x}, t). \quad (83)$$

For simplicity, in this chapter, we call the first step (a) the “distribution reconstruction” step, the second step (b) “regular streaming” while the third will be denoted by the “moment-based collision” step.

6.2 High-order moment-encoded LBM

We now introduce our High-Order Moment-Encoded LBM (HOME-LBM) approach, which removes most of the shortcomings of MR-LBM by modifying its first and third steps to gain substantial accuracy and stability. In this section, we will assume a D3Q27 lattice discretization in 3D (resp., D2Q9 in 2D) as typically recommended for turbulent flows to ensure numerical accuracy.

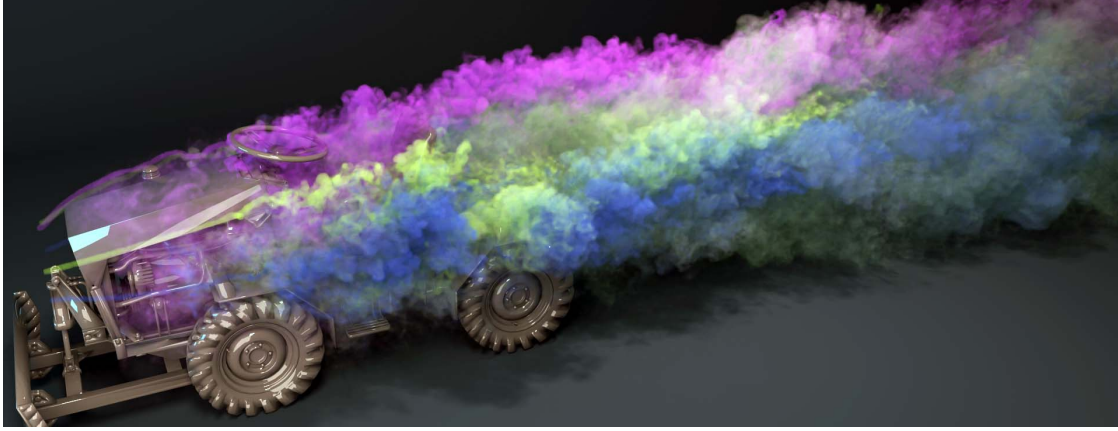


Fig. 13. **Lightweight and lightspeed LBM simulation:** HOME-LBM is a new kinetic solver for incompressible and single-phase fluid simulation interacting with complex solids. By relying on only three moments per grid node instead of the usual 27-direction velocity discretization, we reduce memory usage threefold and improve efficiency by an order of magnitude, with limited impact on accuracy. As a consequence, a high-quality visual simulation of a tractor in a wind tunnel (with passively-advected colored particles to visualize the flow) can be achieved efficiently on a commodity GPU.

Discrete distributions. From Sec. 2.4, we have known when a lattice discretization of the local velocity $\mathbf{v}$ into q discrete microscopic velocities $\{\mathbf{c}_i\}_{i=0}^{q-1}$ ($q=9$ in 2D, $q=27$ in 3D, see Fig. 4) is used and using the lattice weights w_i from Eq. (81) as quadrature weights, the *discrete* distribution function corresponding to $f(\cdot, \mathbf{x}, t)$ are the q values $\{f_i\}_{i=0}^{q-1}$ per node and per timestep defined as:

$$f_i(\mathbf{x}, t) = \frac{w_i}{\omega(\mathbf{c}_i)} f(\mathbf{c}_i, \mathbf{x}, t), \quad (84)$$

and coefficients are calculated through Gauss-Hermite quadrature as:

$$\mathbf{a}^{[n]}(\mathbf{x}, t) = \sum_{i=0}^{q-1} f_i(\mathbf{x}, t) \mathbf{H}^{[n]}(\mathbf{c}_i). \quad (85)$$

One can verify that the coefficients $\mathbf{a}^{[0]}$, $\mathbf{a}^{[1]}$, and $\mathbf{a}^{[2]}$ correspond to the discrete moments from Eqs. (28).

6.2.1 High-order distribution reconstruction. In the MR-LBM approach [29, 42, 91], discrete distribution functions are not stored on the computational grid to avoid large memory footprint, but are replaced by only the first three velocity moments ρ , $\rho\mathbf{u}$, and ρS at each grid node instead. From these moments, a second-order Hermite approximation of the distribution of the form of Eq. (22) that matches these velocity moments is derived by simply choosing only three non-zero coefficients ($\mathbf{a}^{[0]}$, $\mathbf{a}^{[1]}$, $\mathbf{a}^{[2]}$) in the Hermite expansion and setting these coefficients to the three known moments (Eq. (28)). This low-order reconstruction of the 27-direction values f_i leads to a very simple expression of the distribution function given in Eq. (81) — and hence, a very cheap reconstruction — but introduces significant truncation errors due to its particularly low-order in the case of turbulent flows, shown in Fig 14 (a).

Regularized distributions. Instead, we can leverage existing works to reconstruct a third-order Hermite approximation of the distribution which will improve the precision and numerical stability of our HOME-LBM. Sec. 3.5 introduced the concept of *regularized distribution* for standard LBM, which stemmed from making sure that the BGK collision model leads to the correct macroscopic equations in the hydrodynamic limit, by *filtering* distribution functions after streaming and before collision to remove unwanted oscillations (“ghost modes”) and instabilities in the simulation. From the current distribution $\mathbf{f}$ of a node, the first three velocity moments ρ , $\mathbf{u}$, and $\mathbf{a}^{[2]}$ are evaluated, and a particular

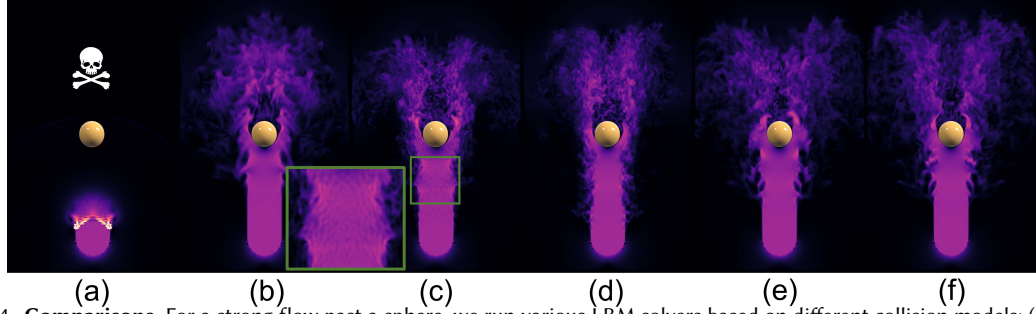


Fig. 14. **Comparisons.** For a strong flow past a sphere, we run various LBM solvers based on different collision models: (a) MR-LBM [29] which blows up at an early frame, (b) original NOCM-MRT [20], (c) optimized NOCM-MRT [64], (d) CU-MRT [54], followed by HOME-LBM with cumulant-based (e) and central-moment-based (f) collisions. While our approach does not capture exactly the high-order cumulant collision model (e), both our options behave similarly, without the ringing artifacts of (c) or the clear oversmoothing of (b) as exhibited by a cross-section of the flow field indicating the velocity magnitude.

choice of a second-order Hermite truncated series for the non-equilibrium part f^{neq} (i.e., the original distribution minus the equilibrium distribution $f^{\text{eq}}(\rho, \mathbf{u})$) is chosen to ensure that the regularized distribution satisfies key symmetries. Malaspinas [77] proposed another regularization, now relying on a full sixth-order Hermite series (in 3D), where the various coefficients $\mathbf{a}^{[n>2]}$ are recursively evaluated from ρ , $\mathbf{u}$, and $\mathbf{a}^{[2]}$ to remove the typical numerical inaccuracies of high-order moments.

Third-order Hermite reconstruction. Compared to the previous works, HOME-LBM is *not trying to filter an existing distribution, but to reconstruct one instead* — but one can use their approach to turn the three velocity moments into a valid distribution function. While regularizations are often based on second-order Hermite truncations or full sixth-order Hermite series, one can opt for a third-order Hermite truncation to offer a good balance between accuracy and computational efficiency. Since the moment representation of LBM stores ρ (to which we set $\mathbf{a}^{[0]}$), $\rho\mathbf{u}$ (to which we set $\mathbf{a}^{[1]}$), and $\rho\mathbf{S}$ (to which we set $\mathbf{a}^{[2]}$), we can use the recursive computation of third-order coefficients $\mathbf{a}^{[3]}$ from Malaspinas [77] as is for the reconstruction of a third-order Hermite approximation of the distribution. The advantages are obvious: due to their regularization procedure, the reconstructed distribution is void of significant ghost modes; but our truncation at the third term of the Hermite expansion offers improved computational efficiency, obviously.

Reconstruction in closed-form. While the recursive evaluation of $\mathbf{a}^{[n>2]}$ from [77] can be used, we can also express the resulting coefficients directly in closed form as a function of the first three velocity moments ρ , $\rho\mathbf{u}$, and $\rho\mathbf{S}$, leading in 3D to the following third-order Hermite reconstruction expression for each of the distribution function f_i :

$$\begin{aligned}
 f_i = & \rho w_i \left[1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{\mathbf{H}^{[2]}(\mathbf{c}_i) : \mathbf{S}}{2c_s^4} \right. \\
 & + \frac{1}{2c_s^6} \left(\mathbf{H}_{ixxy}^{[3]} (S_{xx}u_y + 2S_{xy}u_x - 2u_xu_yu_y) \right. \\
 & + \mathbf{H}_{ixyy}^{[3]} (S_{yy}u_x + 2S_{xy}u_y - 2u_xu_yu_y) \\
 & + \mathbf{H}_{ixxz}^{[3]} (S_{xx}u_z + 2S_{xz}u_x - 2u_xu_zu_z) \\
 & + \mathbf{H}_{ixzz}^{[3]} (S_{zz}u_x + 2S_{xz}u_z - 2u_xu_zu_z) \\
 & + \mathbf{H}_{iyzz}^{[3]} (S_{zz}u_y + 2S_{yz}u_z - 2u_yu_zu_z) \\
 & + \mathbf{H}_{iyyz}^{[3]} (S_{yy}u_z + 2S_{yz}u_y - 2u_yu_yu_z) \\
 & \left. \left. + \mathbf{H}_{ixyz}^{[3]} (S_{xz}u_y + S_{yz}u_x + S_{xy}u_z - 2u_xu_yu_z) \right) \right]. \tag{86}
 \end{aligned}$$

6.2.2 Moment-based Collision in HOME-LBM. In order to improve efficiency, we can use an alternate evaluation based on the velocity moments, still affording a high-order collision evaluation. After streaming the reconstructed distribution, we directly compute the three new moments ρ^* , $\rho^* \mathbf{u}^*$, and $\rho^* \mathbf{S}^*$ for each of the resulting distributions per node based on Eq. (28) *before the collision step*. (The use of the superscript asterisk indicates that these moments are temporary: they will be altered by the collision.) Knowing ρ^* and $\mathbf{u}^*$ on a grid node allows us to know the equilibrium distribution defined by Eq. (15). The resulting central-moment $\mathbf{m}^{\text{eq}} = \mathbf{M} \mathbf{f}^{\text{eq}}$ and the external force $\mathbf{F}$ are then projected into the Hermite-based form of Eq. (22): following [64], we use a sixth-order Hermite expansion here, leading to mostly zero terms except for:

$$m_0^{\text{eq}} = m_9^{\text{eq}} = \rho, \quad m_{17}^{\text{eq}} = \frac{1}{3}\rho, \quad m_{18}^{\text{eq}} = \frac{1}{9}\rho, \quad m_{26}^{\text{eq}} = \frac{1}{27}\rho \quad (87)$$

for the central moments, and, for the force-related terms,

$$\begin{aligned} K_1 &= F_x, & K_2 &= F_y, & K_3 &= F_z, \\ K_{10} &= \frac{2}{3}F_x, & K_{11} &= \frac{2}{3}F_y, & K_{12} &= \frac{2}{3}F_z, \\ K_{23} &= \frac{1}{9}F_x, & K_{24} &= \frac{1}{9}F_y, & K_{25} &= \frac{1}{9}F_z. \end{aligned} \quad (88)$$

To evaluate the actual post-streaming distribution f from the updated velocity moments, we chose a third-order Hermite expansion to improve upon previous works (which all used a second-order one) while keeping the collision evaluation simple. Indeed, the resulting effect of the collision encoded by Eq. (53) ends up being quite straightforward: the post-collision velocity moments do not require the lengthy evaluation of (and multiplications by) matrices $\mathbf{M}$ and its inverse, as the closed-form expressions of the three velocity moments are:

$$\rho(\mathbf{x}, t+1) = \rho^*; \quad (89)$$

$$u_\alpha(\mathbf{x}, t+1) = u_\alpha^* + \frac{1}{2\rho^*} F_\alpha; \quad (90)$$

$$S_{xy}(\mathbf{x}, t+1) = (1 - \frac{1}{\tau}) S_{xy}^* + \frac{1}{\tau} u_x^* u_y^* + \frac{2\tau-1}{2\tau\rho^*} (F_x u_y^* + F_y u_x^*); \quad (91)$$

$$\begin{aligned} S_{xx}(\mathbf{x}, t+1) &= \frac{\tau-1}{3\tau} (2S_{xx}^* - S_{yy}^* - S_{zz}^*) + \frac{1}{3} (u_x^{*2} + u_y^{*2} + u_z^{*2}) \\ &\quad + \frac{1}{3\tau} (2u_x^{*2} - u_y^{*2} - u_z^{*2}) + \frac{1}{\rho^*} F_x u_x^* + \frac{\tau-1}{3\tau\rho^*} (2F_x u_x^* - F_y u_y^* - F_z u_z^*); \end{aligned}$$

$$\begin{aligned} S_{yy}(\mathbf{x}, t+1) &= \frac{\tau-1}{3\tau} (2S_{yy}^* - S_{xx}^* - S_{zz}^*) + \frac{1}{3} (u_x^{*2} + u_y^{*2} + u_z^{*2}) \\ &\quad + \frac{1}{3\tau} (2u_y^{*2} - u_x^{*2} - u_z^{*2}) + \frac{1}{\rho^*} F_y u_y^* + \frac{\tau-1}{3\tau\rho^*} (2F_y u_y^* - F_x u_x^* - F_z u_z^*); \end{aligned}$$

$$\begin{aligned} S_{zz}(\mathbf{x}, t+1) &= \frac{\tau-1}{3\tau} (2S_{zz}^* - S_{xx}^* - S_{yy}^*) + \frac{1}{3} (u_x^{*2} + u_y^{*2} + u_z^{*2}) \\ &\quad + \frac{1}{3\tau} (2u_z^{*2} - u_x^{*2} - u_y^{*2}) + \frac{1}{\rho^*} F_z u_z^* + \frac{\tau-1}{3\tau\rho^*} (2F_z u_z^* - F_x u_x^* - F_y u_y^*). \end{aligned}$$

Note that these expressions (except for the trivial Eqs. (89)-(90) which just reflect density and linear momentum preservation of the collision process) are far simpler than having to deal with matrix $\mathbf{M}$ and its inverse in the usual LBM collision process: therefore, the moment-encoded LBM provides a great boost in efficiency due to these simple collision updates, see Fig. 15.

6.2.3 Moment-based single-node coupling. Now that we have addressed how to integrate the lattice Boltzmann equations in the moment-encoded LBM context, the last issue to tackle is fluid-solid coupling. One of the simplest approaches to coupling is the moving bounce-back method [58], which alters the streaming process near a boundary by reversing the

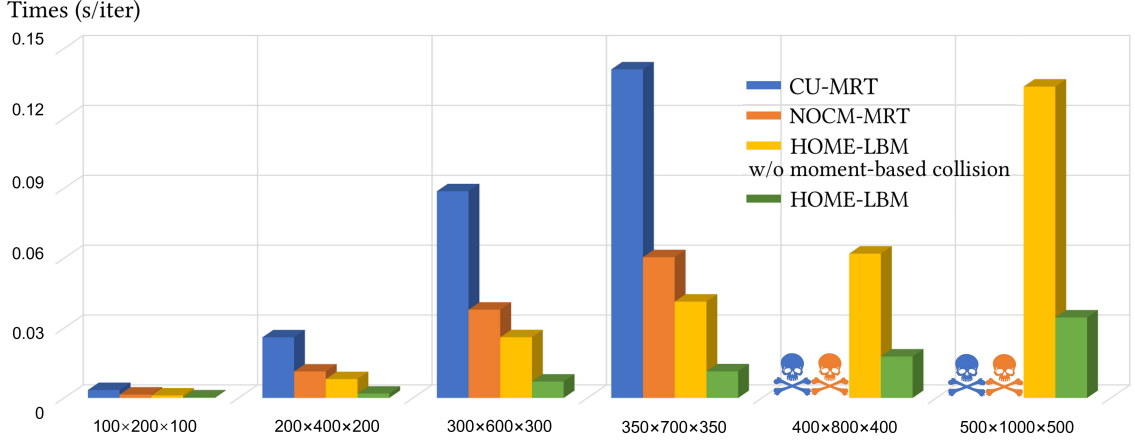
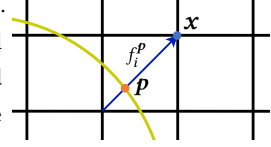


Fig. 15. **Efficiency of LBM methods:** By running the same “flow past sphere” example with three implementations (CU-LBM [54], NOCM-MRT [75], and our HOME-LBM) run on an NVIDIA GeForce RTX 4090 GPU card, we compare the different average times per iteration of these solvers. HOME-LBM is on average 5 times faster than NOCM-MRT and 10 times more efficient than CU-LBM. Moreover, the smaller memory footprint allows us to reach fine grid sizes for which the two other methods run out of memory. In particular, our moment-based collision is a key factor in timing gains.

distribution function advection against the boundary and applying a momentum exchange based on the velocity of the solid encountered. However, its first-order nature often leads to spurious oscillations, especially in turbulent flows, which spurred the introduction of a hybrid velocity-correction method [75] to offer stable two-way coupling, at the cost of larger stencils. One may favor a more direct and local approach that requires fewer computations by leveraging our high-order distribution reconstruction.

Streaming. Streaming is performed on all unobstructed links without any change. Now if a link to a grid node $\mathbf{x}$ intersects a solid boundary (a case often called a “cut-cell link”), we alter streaming so that an approximated distribution function from the solid boundary is streamed to the grid node. Let $\mathbf{p}$ be the point of the intersected solid on the link (see inset). This approach simply amounts to stream a reconstructed distribution value $f_i^{\mathbf{p}}$ to $\mathbf{x}$, as we detail next, so that any type of obstacle geometry (even thin shells) can be handled.



Solid distribution. Given our three-velocity-moment setup, we can approximate $f_i^{\mathbf{p}}$ easily. First, since we are simulating a homogeneous weakly-compressible fluid, $\rho^{\mathbf{p}} = \rho^{\mathbf{x}}$. As for the velocity $\mathbf{u}^{\mathbf{p}}$, we simply pick it based on the motion of the solid boundary at $\mathbf{p}$, which is known exactly based on the linear velocity and angular velocity of the solid. We thus just need to form an approximation of the rank-2 tensor $S^{\mathbf{p}}$. We know that the equilibrium tensor given the known moments $\rho^{\mathbf{p}}$ and $\mathbf{u}^{\mathbf{p}}$ would be $S_{\alpha\beta}^{\mathbf{p},\text{eq}} = u_{\alpha}^{\mathbf{p}} u_{\beta}^{\mathbf{p}}$, so we now need to add an approximation of its off-equilibrium part. We simply use the off-equilibrium moment of node $\mathbf{x}$, resulting in the expression:

$$S_{\alpha\beta}^{\mathbf{p}} = u_{\alpha}^{\mathbf{p}} u_{\beta}^{\mathbf{p}} + (S_{\alpha\beta}^{\mathbf{x}} - u_{\alpha}^{\mathbf{x}} u_{\beta}^{\mathbf{x}}). \quad (92)$$

This approximation is in fact tantamount to assuming $\nabla \mathbf{u}|_{\mathbf{x}} = \nabla \mathbf{u}|_{\mathbf{p}}$ since we know that the off-equilibrium part of S is proportional to the local stress tensor [77]. While this may be too coarse of an approximation for very high Reynolds number (generating thin, fast varying boundary layers), we found it to be sufficient for graphics purposes — and better approximations could be derived from neighboring values of nodes on the same side of the obstacle, albeit at a higher

computational cost. Now that we have the three velocity moments at $\mathbf{p}$, we evaluate the outcoming value f_i^p using Eq. (86) and stream it to $\mathbf{x}$.

Force on solids. In kinetic theory, momentum exchange is used to calculate the resulting force on a solid when fluid particles hit it, expressed in a Galilean-invariant way [81] as:

$$\mathbf{F}dt = f_{\text{in}}(\mathbf{c}_{\text{in}} - \mathbf{u}_p) - f_{\text{out}}(\mathbf{c}_{\text{out}} - \mathbf{u}_p), \quad (93)$$

where {in,out} refer to the velocity index of the cut link and its opposite. One can thus evaluate the impact of the fluid motion onto a solid by summing all the contributions for all nodes $\mathbf{x}$ whose links ℓ_k intersect the solid to get the force $\mathbf{F}_B$ and torque $\boldsymbol{\tau}_B$ through:

$$\mathbf{F}_B \equiv \sum_{\mathbf{x}} \sum_{\ell_k \in L_{\mathbf{x}}} f_{k'}(\mathbf{x}, t) (\mathbf{c}_{k'} - \mathbf{u}_{p(\mathbf{x})}) - f_k(\mathbf{x}, t+1) (\mathbf{c}_k - \mathbf{u}_{p(\mathbf{x})}), \quad (94)$$

$$\boldsymbol{\tau}_B \equiv \sum_{\mathbf{x}} (\mathbf{x} - \mathbf{x}_C) \times \sum_{\ell_k \in L_{\mathbf{x}}} f_{k'}(\mathbf{x}, t) (\mathbf{c}_{k'} - \mathbf{u}_{p(\mathbf{x})}) - f_k(\mathbf{x}, t+1) (\mathbf{c}_k - \mathbf{u}_{p(\mathbf{x})}), \quad (95)$$

where k' is the opposite direction to k , and we use $\mathbf{p}(\mathbf{x})$ to denote the solid point $\mathbf{p}$ on the link associated to grid node $\mathbf{x}$, and $\mathbf{x}_C$ to denote the barycenter of the solid. As force and torque expressions are in LBM space, we further map them to physical space before sending them to the rigid-body integrator, as in Sec. 2.5.

6.3 HOME LBM in free-surface flows and foams with sharp interfaces

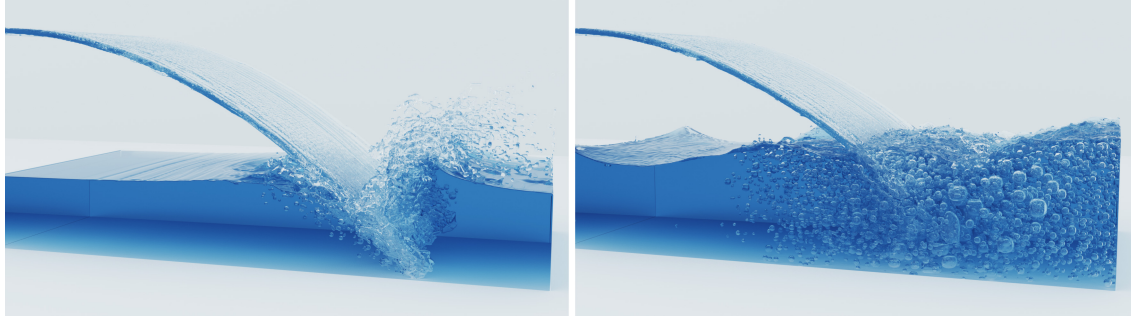


Fig. 16. **HOME-FREE LBM solver.** The HOME-FREE fluid solver mixes a kinetic flow simulator and a volume-of-fluid sharp moving interface. For any given grid resolution, our approach captures far more detailed fluid-air interactions than diffuse-interface based solvers. Moreover, we leverage the fact that the ratio of density and viscosity between air and liquid is large enough that simulating the gas phase can be avoided, the resulting free-surface approximation thus saving a big fraction of computational time. In this example, a fast-evolving sheet of water is flowing into a container, creating turbulence and a large amount of bubbles bursting over time, computed at a resolution of $600 \times 300 \times 300$ in 58s per frame and with only 8 GB of memory.

6.3.1 Free surface lattice Boltzmann model (FSLBM). In order to track the interface motion between the liquid and the gas (air) phase, the volume-of-fluid (VOF) method [48] is used in FSLBM: it consists in advecting a field ϕ in the fluid macroscopic velocity through a transport equation:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \quad (96)$$

where $\phi(\mathbf{x})$ is a dimensionless scalar value representing *the percentage of liquid volume in the dual cell of a node $\mathbf{x}$* , with $\phi=0$ indicating air and $\phi=1$ indicating a full dual cell of liquid. These volume fractions effectively partition the grid

nodes into three categories: L representing liquid nodes, G representing air nodes, and I representing the interface nodes where $0 < \phi < 1$, which should always come in between liquid and gas nodes. A fast mass advection algorithm [57] is used in FSLBM to discretize Eq. (96), which takes advantage of the evolution of distribution functions to track ϕ in the domain: by calculating the mass exchange between neighboring nodes $\mathbf{x}$ and $\mathbf{x} + \mathbf{c}_i$ based on the distribution functions f and since the VOF (see inset) is equal to the fluid mass divided by its density, we get the new value of ϕ at $\mathbf{x}$ as

$$\phi(\mathbf{x}, t+1) = \phi(\mathbf{x}, t) + \frac{1}{\rho(\mathbf{x}, t)} \sum_{i=0}^{q-1} \theta(\mathbf{x}) \cdot (f_i(\mathbf{x} + \mathbf{c}_i, t) - f_i(\mathbf{x}, t)), \quad (97)$$

where $\bar{i}$ is the reverse opposite direction of direction i and $\theta(\mathbf{x})$ is weighting the mass exchange through

$$\theta(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in L \\ 0 & \text{if } \mathbf{x} \in G \\ \frac{1}{2}(\phi(\mathbf{x}, t) + \phi(\mathbf{x} + \mathbf{c}_i, t)) & \text{if } \mathbf{x} \in I. \end{cases} \quad (98)$$

This interface advection approach is particularly convenient as it does not require the reconstruction of the interface or an approximation of its geometric properties such as normals or curvatures. One only needs to update the status of the nodes after advection: if $\phi(\mathbf{x}, t+1) \geq 1 + \epsilon_\phi$ for $\epsilon_\phi = 1e-4$ (resp., if $\phi(\mathbf{x}, t+1) \leq 0 - \epsilon_\phi$), an interface node is converted into a liquid (resp., gas) node and ϕ is clamped to 1 (resp., 0), and the clamped volumes are redistributed to neighboring cells to enforce mass preservation [62].

While free-surface LBM ignores the dynamics of the gas phase, setting correct pressures is crucial to allow for bubbles not to cavitate. This is achieved through a pressure boundary condition: in the LBM step of Eq. (34), the distribution f from the gas to the interface node is unknown, but approximated via

$$f_i^*(\mathbf{x}, t) = f_i^{\text{eq}}(\rho_g, \mathbf{u}(\mathbf{x}, t)) + f_{\bar{i}}^{\text{eq}}(\rho_g, \mathbf{u}(\mathbf{x}, t)) - f_{\bar{i}}(\mathbf{x}, t), \quad (99)$$

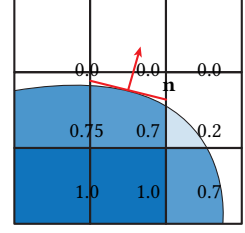
where $\bar{i}$ indicates the reversed index satisfying $\mathbf{c}_{\bar{i}} = -\mathbf{c}_i$. The gas density ρ_g is estimated through

$$\rho_g = (p_g - 2\gamma\kappa(\mathbf{x}))/c_s^2, \quad (100)$$

which accounts for the surface tension coefficient γ of the interface, the atmospheric pressure p_g , and the local mean curvature $\kappa(\mathbf{x})$ of the interface encoded by the volume fraction ϕ , where κ is evaluated from a local piecewise-linear interface construction (PLIC) of the interface [62, 99].

Extend HOME-LBM solver for FSLBM. The resulting HOME-FREE LBM solver thus proceeds as follows. First, we perform the streaming step just like in FSLBM, but the distribution function f_i from the neighboring node (the right term in Eq. (34)) is evaluated on the fly from the three velocity-moments of this node by a third-order Hermite-based “filtered” reconstruction through Eq (86). We then convert the streamed distributions back into their temporary velocity-moments ρ^* , $\mathbf{u}^*$ and S^* via Eq. (28). Finally, the collision step evaluates the final velocity-moments as Eqs. (89)-(90)

Incorporating both the HOME-LBM approach and a higher-order collision model into our kinetic free-surface method has already dramatic consequences on the stability of the solver: as Fig. 17 demonstrates, a basic dam break generates instabilities with the BGK or two-relaxation-time (TRT) collision model, see Ch. 3, but not with HOME-FREE LBM. The accuracy of HOME-FREE LBM improves several aspects of the free-surface treatment, such as the interface pressure



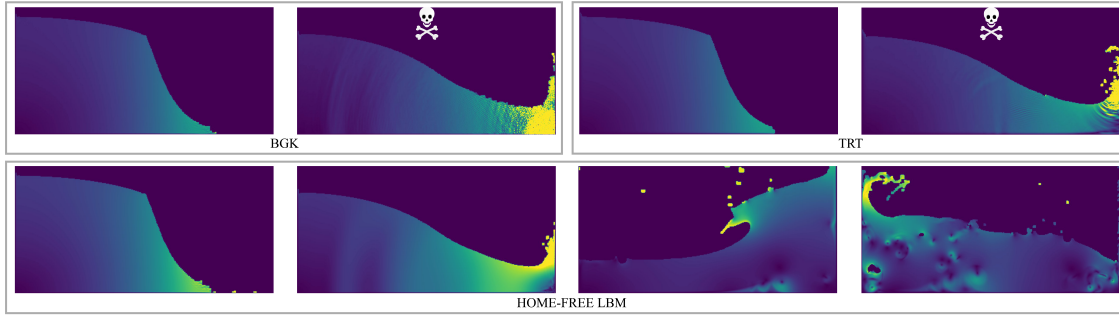


Fig. 17. **Comparisons between BGK, TRT, and HOME-FREE models.** For a simple 2D dam break, an FSLBM simulator using a BGK collision model crashes after 71 frames (top left); the TRT model allows the simulation to go a few frames further but, again, crashes at frame 75 (top right). Our HOME-FREE LBM simulator handles this case easily (bottom), never crashing.

boundary condition from Eq. (99): it uses the filtered distribution function f_i reconstructed using Eq. (86) which offers a more accurate treatment.

6.3.2 Efficient bubble model for thin shell. Although the HOME-FREE setup we presented above can simulate high Reynolds number liquid with splashes, it cannot support any type of bubbling effects thus far. Formulating an efficient bubble model in this current framework is thus crucial.

Existing treatment of bubbles in FSLBM. In traditional FSLBM, gas distribution function values are not simulated, but reconstructed on demand through Eq. (99), and gas pressure inside a bubble is always assumed to be the atmospheric pressure $p_g = p_{atmos} := c_s^2$. This is very limiting, as bubble pressure should change due to bubbles growing, shrinking, and/or merging. More involved bubble models [5] have been proposed where the volume change of a bubble b_i (defined as a connected region of gas and interface nodes surrounded by liquid) is recorded before and after one simulation step so that the pressure update can be evaluated based on the ideal gas law (for an isothermal simulation) as:

$$p(b_i, t) = p^{atmos} \frac{V(b_i, 0)}{V(b_i, t)}, \quad (101)$$

where $p(b_i, t)$ and $V(b_i, t)$ refer to bubble pressure and volume at the current time step t respectively, $V(b_i, 0)$ is the initial bubble volume, while p^{atmos} is the atmospheric pressure equal to c_s^2 in LBM units. Note that the bubble volume is simply the sum of the complement of the fluid volume fractions ϕ 's in the bubble region:

$$V(b_i, t) = \sum_{x \in b_i} (1 - \phi(x, t)). \quad (102)$$

Handling bubbles thus needs to store the values used in Eq. (101) for each bubble, and to indicate for every non-liquid node the index of the bubble it corresponds to. Indices are updated at each time step by referring to the previous neighboring indices so as to track the motion of the bubbles. Merging and splitting of bubbles is even more involved: it first requires to detect splits and merges based on the connectedness of nodes of a given index, then to update the initial volumes and pressures of the newly created bubbles. A serial algorithm to achieve this task was proposed by Foley [31] and Bogner [9], involving local flood filling algorithms and bipartite graphs between consecutive arrangements of bubble indices. Alas, this approach is not parallelizable and does not scale well with the number of bubbles; moreover, it cannot accommodate thin-shell obstacles in the fluid, which are particularly desirable in graphics.

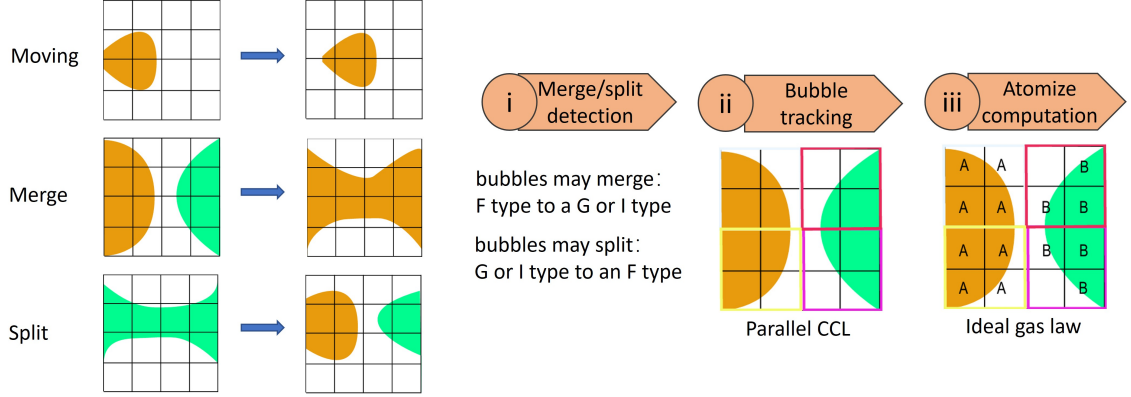


Fig. 18. Efficient bubble model.

HOME-FREE bubbles. In order to create an efficient bubble model, one can use an algorithmic approach to bubble tracking and updates which is easily implemented on GPUs and which leverages the fact that LBM simulators advect a VOF value by *at most one grid node at a time*, which facilitates bookkeeping of bubble indices, shown in Fig. 18. First, we quickly check if some nodes switched from an F type (fluid) to a G or I type (gas or interface) over the last time step as it may mean either that the node just entered a bubble, or multiple bubbles are merging; we determine which case it is by simply storing and checking the indices of the nodes in the immediate vicinity. Similarly, we check if nodes switched from a G or I type to an F type as it may mean that these nodes left a bubble or a bubble is splitting; again, a simple exploration of the 27 indices immediately adjacent to the processed node allows us to find which situation is happening, and the indices of the bubbles at play are stored. The detection of either one of these cases triggers also a *connected component labeling* (CCL) to identify all the bubbles in the fluid — one can use the parallel CCL from [3], which employs a fast block-based union-find data structure to identify the 3D connected interface/gas nodes and mark all connected ones with the same label. But unlike previous methods that go through the (algorithmically painful) exercise of matching old labels with new labels through bipartite graphs to be sure to properly track bubbles, we can proceed very differently from there on. For each grid node that is of type G or I, we store both the current index i of the bubble it belongs to, and also the previous index i^{old} (note that this index is set for all G/I nodes, since even nodes freshly covered by a bubble know the previous bubble index of which they are now part due to the earliest part of our algorithm explained above); and we also keep all the (current and initial) volumes and (current) pressures of all the bubbles from the previous time step, denoted as V_j^{old} , $V_j^{0,\text{old}}$, and p_j^{old} respectively. We then update the volumes and pressures of all bubbles in a way that *does not require to know correspondences between old and new indices*. I.e., after initializing all the current and initial volume values to zero for all the identified bubbles, we go through each G/I node $\mathbf{x}$ with new and old bubble indices i and i^{old} , and perform the following volume updates (through atomic adds):

$$V_i += 1 - \phi(\mathbf{x}, t); \quad V_i^0 += (1 - \phi(\mathbf{x}, t)) p_{i^{\text{old}}}^{\text{old}} / p^{\text{atmos}}. \quad (103)$$

This approach, massively parallel by design, will gather all the (current and initial) volumes of the current bubbles directly and efficiently, even if each bubble has changed its index assigned by the new CCL pass. Note that this update relies on accurate floating-point operations to maintain the right volumes and pressures: one should use double-precision. Once all nodes are updated, we go over each current bubble index and update its pressure via:

$$p_i = p^{\text{atmos}} V_i^0 / V_i. \quad (104)$$

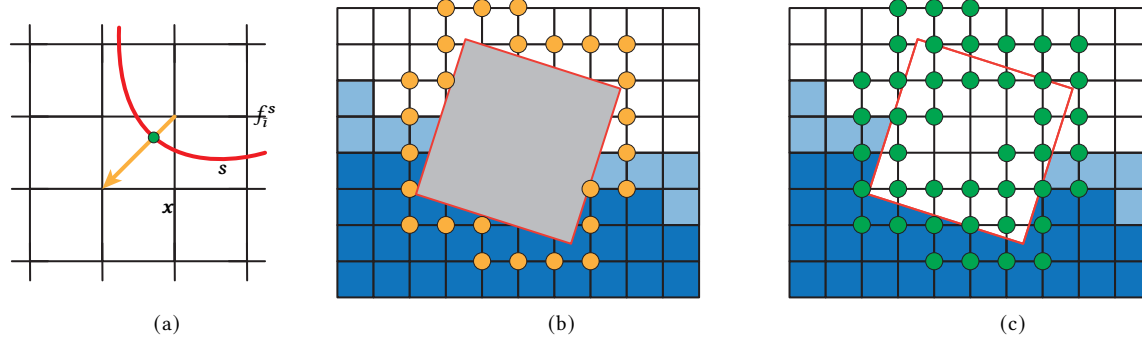


Fig. 19. **Cut-cell treatment of obstacles.** A cut-cell link (a) is a lattice edge (orange) intersecting the boundary of an obstacle, on which we store the intersection point s ; cut-cell nodes for the regular bounce-back approach (b) are grid nodes that have cut-cell links (in orange); our double-sided bounce-back treatment uses all the cut-cell nodes (in green) with cut-cell links, even for those inside the obstacle.

Dealing with cut-cell nodes. While existing FSLBM techniques do not allow for thin-shell obstacles, we can modify our approach slightly to handle this common case in graphics. One can follow the cut-cell representation for thin shells in single-phase LBM [75]; that is, a grid node for which one (or more) of its links intersect a boundary is marked as a *cut-cell node* — and we simply ignore cut-cell nodes during the CCL labeling phase. This amounts to considering the gas in a cut-cell region as being air at the atmospheric pressure. While this allows the use of thin-shell obstacles, it changes the treatment of grid nodes touching thick obstacles, as they would be treated in the CCL instead of being skipped.

Turbulence model to tame small bubbles. When dealing with turbulent flows, very small bubbles bring additional challenges: as they often impose large pressures locally due to Eq. (104), strong ringing artifacts in the velocity field can appear. Consequently, velocities exceeding the CFL condition introduce errors in the advection of small bubbles, generating erratic displacements and causing some bubbles to disappear. These dispersive errors in the velocity field are often dealt with numerically through the addition of a turbulence model. We thus include *eddy viscosity* to all nodes less than four cells away from a bubble to prevent these issues. In practice, this means that we add to the usual fluid viscosity ν an eddy viscosity $\nu_e = 4\|S\|_F$, i.e., the eddy viscosity is set proportional to the Frobenius norm of the second velocity-based moment of the fluid to counteract any sharp local changes in the flow.

6.3.3 Fluid-solid coupling. An important part of any flow simulator is fluid-structure interaction, also called coupling.

Existing FSLBM coupling. In FSLBM, fluid-solid coupling is often handled via bounce-back in Eq. (62). However, FSLBM uses the original bounce-back approach, which relies on nodes being tagged as inside or outside moving solids at each time-step [9] and thus cannot handle thin structures (Fig. 19(b)). Moreover, its first-order nature leads to spurious oscillations in turbulent flows. To transfer the effect of the fluid motion to the obstacle, FSLBM uses a moment-exchange formulation where a moment Δj_i is evaluated per lattice direction, and the coupling force and torque on the solid is the sum of all these contributions; that is,

$$F_B \equiv \sum_{\mathbf{x} \in C_s} \sum_{i \in L_s} \Delta j_i(\mathbf{x}), \quad (105)$$

$$\boldsymbol{\tau}_B \equiv \sum_{\mathbf{x} \in C_s} (\mathbf{x} - \mathbf{x}_c) \times \sum_{i \in L_s} \Delta j_i(\mathbf{x}), \quad (106)$$

where C_s represents the set of all grid nodes adjacent to the solid s (orange nodes in Fig. 19 (b)), L_s is the lattice direction set towards the solid, while $\mathbf{x}_c$ is the barycenter of the solid. Note that FSLBM does not store a distribution function on

gas nodes, so the gas pressure estimated from the equilibrium function assumes a zero velocity $\mathbf{u}_g = \mathbf{0}$. Therefore, if C_s contains gas, interface, and liquid nodes, the lattice moment Δj_i is evaluated through:

$$\Delta j_i(\mathbf{x}) = (f_i^*(\mathbf{x}, t) \mathbf{c}_i - f_i(\mathbf{x}, t) \mathbf{c}_i) \phi_x + 2f_i^{\text{eq}}(\rho_g, \mathbf{0}) \mathbf{c}_i (1 - \phi_x), \quad (107)$$

where a blending between the two phases is achieved through a weight based on the local VOF value ϕ_x . Yet, similar to the bounce-back case, thin obstacles cannot be handled with this approach; moreover, Galilean invariance is not even preserved, which creates spurious artifacts in turbulent flows. These issues can be remedied as we review next.

Cut-cell treatment. Inspired by the double-sided bounce-back approach proposed in [75], we first mark the links intersecting a solid boundary surface as *cut-cell links* (Fig. 19(a)), and any node containing a cut-cell link is marked as a cut-cell node (Fig. 19(c) in green). We then apply the bounce-back only on fluid and interface cut-cell nodes — not on gas nodes (nodes covered by the solid obstacle in Fig. 19(c)). This cut-cell boundary treatment naturally supports thin and non-watertight objects. We also incorporate the fluid-solid coupling treatment from HOME-LBM in Sec 6.2.3: the approximated distribution function from the solid boundary point $\mathbf{s}$ (at the intersection of a cut-cell link and a solid) that is streamed to a grid node $\mathbf{x}$ is reconstructed from Eq. (86) using $\rho = \rho_x$, $\mathbf{u} = \mathbf{u}^s$, and $S_{\alpha\beta} = u_\alpha^s u_\beta^s + (S_{\alpha\beta}^x - u_\alpha^x u_\beta^x)$.

Fluid-to-solid force exchange. To accommodate general obstacles, we modified Eq. (107) with a central moment scheme for F/I nodes to ensure Galilean invariance [81] through

$$\Delta j_i = (f_i^*(\mathbf{x}, t) (\mathbf{c}_i - \mathbf{u}^s) - f_i(\mathbf{x}, t) (\mathbf{c}_i - \mathbf{u}^s)) \phi_x + ((\mathbf{c}_i - \mathbf{u}^s) f_i^{\text{eq}}(\rho_g, \mathbf{0}) - (\mathbf{c}_i - \mathbf{u}^s) f_i^{\text{eq}}(\rho_g, \mathbf{0})) (1 - \phi_x), \quad (108)$$

where $f_i^*(\mathbf{x}, t)$ is reconstructed from Eq. (86) with the three velocity-moments derived from the velocity of the obstacle as described above. Finally, the total force remains expressed as Eq. (105), while the total torque is changed to:

$$\boldsymbol{\tau}_B \equiv \sum_{\mathbf{x} \in C_s} (\mathbf{x}_s - \mathbf{x}_c) \times \sum_{i \in L_s} \Delta j_i(\mathbf{x}), \quad (109)$$

where C_s represents the set of all cut-cell nodes (Fig. 19(c) in green), and $\mathbf{x}_s$ is an intersection point between the link and the obstacle boundary (point $\mathbf{s}$ in Fig. 19(a)), instead of Eq. (106) which used the current grid node. Note that in our implementation, we re-express Eqs. (105), (108) and (109) by noticing that $f_i^{\text{eq}}(\rho_g, \mathbf{0}) = w_i = w_i(\rho_g = 1)$ and $\mathbf{c}_i = -\mathbf{c}_i$, which yields the following expressions:

$$\begin{aligned} \Delta j_i &= \phi_x (f_i^*(\mathbf{x}, t) + f_i(\mathbf{x}, t) - 2w_i) \mathbf{c}_i - \phi_x (f_i^*(\mathbf{x}, t) - f_i(\mathbf{x}, t)) \mathbf{u}^s + 2c_i w_i, \\ \Delta \hat{j}_i &= \phi_x (f_i^*(\mathbf{x}, t) + f_i(\mathbf{x}, t) - 2w_i) \mathbf{c}_i - \phi_x (f_i^*(\mathbf{x}, t) - f_i(\mathbf{x}, t)) \mathbf{u}^s, \\ F_B &= \sum_{\mathbf{x} \in C'_s, i \in L_s} \Delta \hat{j}_i, \quad \boldsymbol{\tau}_B = \sum_{\mathbf{x} \in C'_s} (\mathbf{x}_s - \mathbf{x}_c) \times \sum_{i \in L_s} \Delta \hat{j}_i. \end{aligned} \quad (110)$$

where now the node set C'_s only includes fluid and interface cut-cell nodes, thus simplifying the evaluation since gas nodes are skipped.

6.3.4 Foam modeling. Foam, where many gas bubbles are surrounded by thin films (lamellae) of liquid, is a highly recognizable phenomenon often seen on beaches and coffee cups. Capturing this phenomenon is impractical for diffuse phase models such as [67], as the proper handling of thin films would require an extremely fine grid. Ataei et al. [5] showed that one can model this complex phenomenon with FSLBM by *solving an advection-diffusion equation for the content of dissolved gas in the liquid and modeling disjoining pressure* which arises from the attractive interaction between two bubbles. One can adapt this approach next to make it robust, once again, to more turbulent cases.

Advection-diffusion equation. Ataei et al. [5] proposed to model advection and diffusion of a field representing the dissolved gas concentration φ (in terms of mass fraction) using

$$\frac{\partial \varphi}{\partial t} + \nabla \cdot (\varphi \mathbf{u}) = \nabla \cdot (\mu_g \nabla \varphi) + q, \quad (111)$$

where q is a source term responsible for gas generation due to, for example, chemical reactions, and μ_g is the diffusion coefficient. This PDE is solved through LBM discretization as well, using another distribution function $\mathbf{g}$ on a D3Q7 lattice structure this time (with lattice directions $\mathbf{d}_i$, weights $w_0^g = 1/4$ and $w_i^g = 1/8$ for $i \in \{1, \dots, 6\}$, with speed of sound $d_s = \frac{1}{2}$, see Fig. 4) as it does not require the same accuracy as for the fluid. The lattice Boltzmann equations for the distribution function values g_i then read

$$g_i(\mathbf{x} + \mathbf{d}_i, t + 1) - g_i(\mathbf{x}, t) = \Omega_i^g + q_i, \quad (112)$$

where $q_i = w_i^g q$ and $\Omega_i^g = -(\mathbf{g}_i - \mathbf{g}_i^{\text{eq}})/\tau_g$ for the BGK collision model (sufficiently accurate for this advection-diffusion simulation) where the relaxation time is linked to the diffusion constant via $\tau_g = 4\mu_g + \frac{1}{2}$. As usual, the zeroth-order moment of the distribution function $\mathbf{g}$ allows for the reconstruction of the gas concentration φ , i.e.,

$$\varphi(\mathbf{x}, t) = \sum_{i=0}^6 g_i(\mathbf{x}, t). \quad (113)$$

Moreover, from the local macroscopic fluid velocity $\mathbf{u}$ and the local concentration $\varphi(\mathbf{x}, t)$, the equilibrium distribution $\mathbf{g}^{\text{eq}}$ can be well approximated with a first-order Maxwell distribution:

$$\mathbf{g}_i^{\text{eq}}(\mathbf{x}, t) = w_i \varphi(\mathbf{x}, t) (1 + \mathbf{d}_i \cdot \mathbf{u} / d_s^2). \quad (114)$$

Ataei et al. [5] also proposes to account for the gas diffusing into the bubble volumes by adding to all bubble initial volumes:

$$V_i^0 \leftarrow \frac{\rho}{p^{\text{atmos}}} \sum_{\mathbf{x} \in b_i} \left(\sum_{\mathbf{x} + \mathbf{c}_i \in F} [g_i(\mathbf{x} + \mathbf{c}_i, t) - g_i(\mathbf{x}, t)] - \varphi(\mathbf{x}, t) [\phi(\mathbf{x}, t) - \phi(\mathbf{x}, t - 1)] \right), \quad (115)$$

where V_i^0 is the initial volume of bubble b_i as discussed in Sec. 6.3.2. Finally, the concentration of gas in a bubble interface is known to obey Henry's law [47] in foams, which expresses a linear relationship between gas concentration and bubble pressure:

$$\varphi(\mathbf{x}, t) = k_H p_g, \quad \forall \mathbf{x} \in I, \quad (116)$$

where k_H is Henry's law constant and p_g is the bubble pressure. One can thus use this property as a way to set boundary conditions similar to Eq. (99) for the free surface treatment through:

$$g_i^*(\mathbf{x}, t) = g_i^{\text{eq}}(\varphi(\mathbf{x}, t), \mathbf{u}) + g_i^{\text{eq}}(\varphi(\mathbf{x}, t), \mathbf{u}) - g_i(\mathbf{x}, t). \quad (117)$$

Disjoining pressure. A disjoining pressure Π has also been proposed to stabilize the lamellae between bubbles, based on the theory of thin liquid films, in order to allow the formation of foam. This disjoining pressure shows a linear dependence in the distance d between two interfaces of bubbles and starts acting from the maximum distance d_{max} (typically set to 4 grid-cell widths) to minimize the variation of Gibbs free energy for two interfaces [53], yielding:

$$\Pi = \begin{cases} 0 & \text{if } d > d_{\text{max}} \\ k_\pi (1 - d/d_{\text{max}}) & \text{if } d \leq d_{\text{max}}, \end{cases} \quad (118)$$

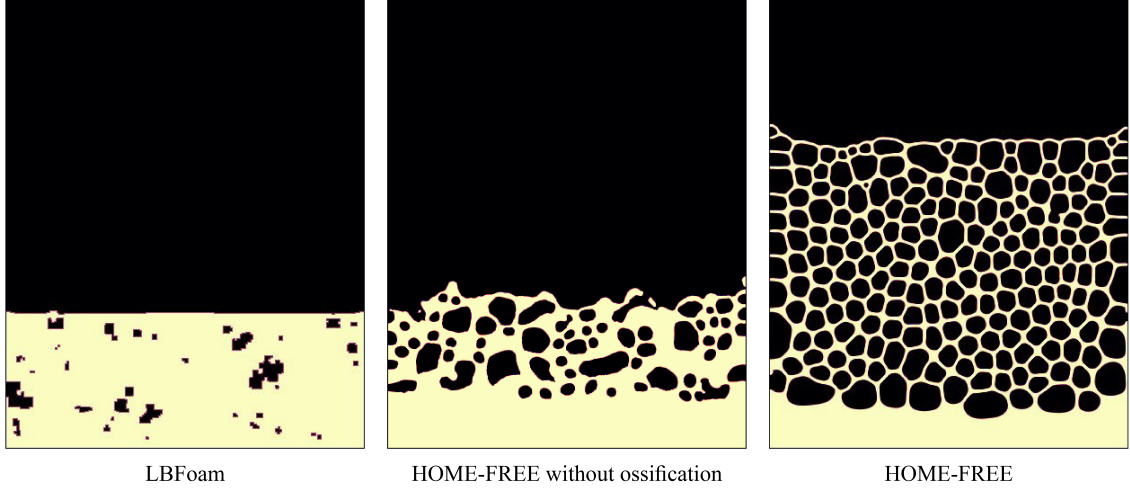


Fig. 20. **Comparisons on 2D foam.** We test our foam modeling on bubble nucleation in 2D at $Re = 100,000$. While [5] crashes quickly (left), our solver generates the expected foaming drainage where the liquid flow within the foam is driven by gravity and capillarity (right). Foam lamellae do not persist without our foam-ossifying viscosity (middle).

where k_π is the disjoining pressure strength. This disjoining term gets added to the pressure and surface tension in Eq. (100) to become:

$$\rho_g = (p_g - 2\dot{\gamma}\kappa(x) - \Pi)/c_s^2. \quad (119)$$

New foam-ossifying treatment. However, the foam model above can only handle low Reynolds number foam cases and will crash in turbulent cases, as Fig. 20 (left) at $Re = 100,000$ demonstrates. Increasing surface tension is sometimes advocated to further stabilize bubble shapes, but we found in our experiments that high surface tension has a tendency to drastically slow down small bubbles due to the large surface tension forces overpowering the bubble's motion — thus adversely affecting the simulation away from foams. Thus, we make specific alterations to our solver to improve foam modeling without affecting bubble dynamics. First, we employ the D3Q7 (see in Fig. 22) Central-Moment-Relaxation (CMR) collision model from [67] to improve the accuracy and stability of Eq. (112), where the collision operator is expressed as:

$$\Omega^g = -(M^g)^{-1} R^g M^g (g - g^{eq}) + (M^g)^{-1} (I - \frac{1}{2} R^g) Q, \quad (120)$$

where the matrix M^g transforms the distribution function g and the source term q_i into central-moment space with $Q = M^g q$, which enforces Galilean invariance. The diagonal relaxation matrix, written $R^g = \text{diag}(1, 1/(4\mu_g + \frac{1}{2}), 1/(4\mu_g + \frac{1}{2}), 1/(4\mu_g + \frac{1}{2}), 1, 1, 1)$, has separate relaxation rates of each moment, reducing the instabilities that the BGK model creates. The closed-form expressions of the coefficients of $M^g = (M_0^g, M_1^g, M_2^g, M_3^g, M_4^g, M_5^g, M_6^g)^T$ were given in [67], and are expressed as

$$\begin{aligned} M_0^g &= (1, 1, 1, 1, 1, 1, 1)^T, \\ M_1^g &= \bar{d}_{i,x}, \quad M_2^g = \bar{d}_{i,y}, \quad M_3^g = \bar{d}_{i,z}, \quad M_4^g = \bar{d}_{i,x}^2 - \bar{d}_{i,y}^2, \\ M_5^g &= \bar{d}_{i,x}^2 - \bar{d}_{i,z}^2, \quad M_6^g = \bar{d}_{i,x}^2 + \bar{d}_{i,y}^2 + \bar{d}_{i,z}^2. \end{aligned}$$

where $\bar{d}_{i,\delta} = (d_i - u)_\delta$, with $i \in \{0, \dots, 6\}$ being the lattice index and $\delta \in \{x, y, z\}$ the coordinate. Second, we introduce a local viscosity ν_f for all nodes less than six cells away from bubble interfaces. This helps with the ossification of bubble lamellae but does not affect the rest of the flow. Without this foam-ossifying viscosity, foams will not keep their structure, collapsing quickly at such high Reynolds numbers (Fig. 20). Finally, we locally adjust the surface tension γ

based on the disjoint force value Π through

$$\dot{\gamma} = (\Pi == 0) ? \dot{\gamma} : \dot{\gamma}_{\text{oss}}, \quad (121)$$

i.e., we use a normal surface tension coefficient γ if no disjoint pressure is happening locally (that is, for free-moving bubbles), but help foam ossification through a higher surface tension coefficient $\dot{\gamma}_{\text{oss}}$. Note that one can also modify this switch to include a condition regarding the bubble size if one wants to control, e.g., the preservation of bubbles of a certain size range if needed.

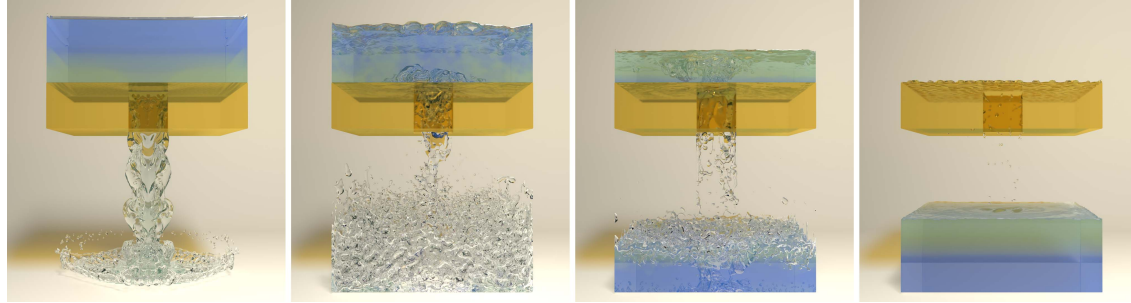


Fig. 21. **Glugging with Rayleigh–Taylor instability:** Our multiphase solver produces glugging for two colored immiscible fluids in the transparent upper container and air in the bottom one (water:oil:air with density ratios 800:400:1), with a square conduit in between. The two fluids are initially arranged with the heavier fluid (blue) on top of the lighter fluid (cyan), thus forming Rayleigh–Taylor instabilities early on, before flowing down and exhibiting splashes, bubbles and wetting, and ending up in stable layers based on their densities.

6.4 HOME LBM in Turbulent Multifluid Flows

Prior sections detailed the surface tension adjustment mechanisms of the HOME-FREE LBM solver when handling phenomena such as free surfaces, free-moving bubbles, and foam ossification. Having established these modeling techniques for local interfacial characteristics, the following section expands the scope to more complex macroscopic flow scenarios, focusing on the application of the HOME LBM framework to the simulation of Turbulent Multifluid Flows, see Fig. 21.

6.4.1 Phase fields for immiscible flows. When only two immiscible phases are present (typically, air and liquid), a single scalar phase field is needed, for which a value of 0 indicates the presence of one phase while a value of 1 indicates the presence of the other phase. The interface between the two phases is delineated with a hyperbolic tangent steep profile from 0 to 1 over an interfacial thickness ξ , see for instance [65–67]. The case of n immiscible phases or fluids (typically, air and a variety of liquids of different densities and viscosities) is, however, more involved. Each phase must be encoded by its own scalar phase field: the i^{th} phase is a scalar function $\phi_i(\mathbf{x}, t)$ where $\mathbf{x}$ is a position within the simulation domain Ω and t is time. Similar to the two-phase case, the immiscible phase-field profile at rest is:

$$\phi_i(\mathbf{x}, 0) = \frac{1}{2}(1 - \tanh(d_i(\mathbf{x})/\xi)), \quad (122)$$

where $d_i(\mathbf{x})$ is the distance from $\mathbf{x}$ to the interface of the i^{th} phase (defined as $\phi^{-1}(0.5)$), implying that the phase field profile satisfies:

$$|\nabla \phi_i| = \frac{4}{\xi} \phi_i(1 - \phi_i). \quad (123)$$

The set of all order parameters $\boldsymbol{\phi} = (\phi_1, \dots, \phi_i, \dots, \phi_n)$ then represents n -fluids *iff* they also satisfy:

$$\sum_{i=1}^n \phi_i(\mathbf{x}, t) = 1 \quad \forall \mathbf{x} \in \Omega. \quad (124)$$

i.e., the volume fractions of the n phases sum to one everywhere, to ensure they, together, fill up the whole simulation domain. This last constraint allows us to store only $(n-1)$ phase fields for n phases: the remaining phase field (say, ϕ_n) is trivially evaluated on the fly as $\phi_n(\mathbf{x}, t) = 1 - \sum_{i=1}^{n-1} \phi_i(\mathbf{x}, t)$.

6.4.2 Generalized phase-field model for immiscible fluids. However, compared to the two-phase case, the sets of constraints described above (Eqs. (123) for $1 \leq i \leq n$ and Eq. (124)) form an over-constrained system. Recent methods [26, 52, 97] have thus proposed a least squares method instead. They consider the following functional to minimize, where Eq. (123) is best enforced under the constraint of Eq. (124):

$$W(\boldsymbol{\phi}, \nabla \boldsymbol{\phi}) = \int_{\Omega} \sum_{i=1}^n \left[|\nabla \phi_i| - \frac{4}{\xi} \phi_i (1 - \phi_i) \right]^2 + \gamma \int_{\Omega} \left(\sum_{i=1}^n \phi_i - 1 \right) d\mathbf{x},$$

where γ is a Lagrangian multiplier to enforce the constraint. Minimizing W via a L^2 -gradient flow leads to the evolution equation:

$$\frac{\partial \phi_i}{\partial t} = -\mu_i \frac{\delta W}{\delta \phi_i}, \quad (125)$$

where $\mu_i > 0$ is the mobility of the i^{th} fluid. If we add an advection of the phase fields in the fluid velocity $\mathbf{u}$, we get the *generalized conservative phase field equations for immiscible fluids*:

$$\frac{\partial \phi_i}{\partial t} + \mathbf{u} \cdot \nabla \phi_i = \mu_i \left\{ \nabla^2 \phi_i - \frac{1}{\xi} \nabla \cdot \left[4\phi_i (1 - \phi_i) \frac{\nabla \phi_i}{|\nabla \phi_i|} \right] \right\} - \frac{8\mu_i (2\phi_i - 1)}{\xi} \left[|\nabla \phi_i| - \frac{4}{\xi} \phi_i (1 - \phi_i) \right] + \gamma, \quad (126)$$

where ξ is the interfacial thickness. Each phase's evolution equation is therefore similar to the two-phase case from previous LBM papers in graphics, except for the presence of the Lagrangian multiplier γ which we obtain by summing Eqs. (126) for i from 1 to n , yielding:

$$\gamma = \frac{1}{n} \sum_j \mu_j \left(\nabla \cdot \left[\frac{4\phi_j (1 - \phi_j)}{\xi} \frac{\nabla \phi_j}{|\nabla \phi_j|} \right] + \frac{8(2\phi_j - 1)}{\xi} \left[|\nabla \phi_j| - \frac{4}{\xi} \phi_j (1 - \phi_j) \right] \right).$$

Various authors came to the conclusion that this expression can be simplified without loss of accuracy and can be made to remain valid in the general case where the number of phases can come and go in the computational domain throughout the simulation. First, the second term can be discarded since this approach is based on making Eq. (123) hold in the least squares sense. Second, to enforce what Dong [26] and Wu and Xu [97] call “reduction consistency” (i.e., the evolution equation should be valid for any number of phases being present at a given time), Hu et al. [52] proposed to replace the single γ by a γ_i per phase, and to replace the $1/n$ term by χ_i/χ where $\chi_i = 1$ (resp., $\chi_i = 0$) indicates that the i^{th} phase is currently present (resp., absent) in the domain of simulation, while $\chi = \sum_i \chi_i$. Indeed, this slightly altered version enforces that χ_i/χ will always be $1/k$ when k fluids are present. Consequently, the following phase field equations were used:

$$\frac{\partial \phi_i}{\partial t} + \mathbf{u} \cdot \nabla \phi_i = \mu_i \left\{ \nabla^2 \phi_i - \frac{1}{\xi} \nabla \cdot \left[4\phi_i (1 - \phi_i) \frac{\nabla \phi_i}{|\nabla \phi_i|} \right] \right\} + \gamma_i \quad (127)$$

where γ_i is expressed as:

$$\gamma_i = \frac{\chi_i}{\xi \chi} \sum_{j=1}^n \mu_j \nabla \cdot \left[4\phi_j (1 - \phi_j) \frac{\nabla \phi_j}{|\nabla \phi_j|} \right]; \quad (128)$$

and as mentioned earlier, only $(n-1)$ phases are actually integrated, as the last one is deduced directly using the constraint in Eq. (124).

6.4.3 Novel lightweight phase-field model. We now introduce a generalized conservative phase field model for (im)miscible flows. Combined with a novel n -phase variant of the recently proposed high-order moment encoded LBM (HOME-LBM) method, we will show that our approach to encode and integrate in time the phase fields needed to simulate multiphase flows removes many of the most stringent shortcomings of related work [52].

6.4.4 Unifying immiscible and miscible flows. While Hu et al. [52] only considered immiscible fluids, we can take a hint from the two-phase case for which the profile is $|\nabla\phi| = \frac{4}{\xi}\phi(1-\phi)$, where the product of one phase (ϕ) and the other phase ($1-\phi$) is directly proportional to the gradient of the phase field. The immiscible case follows the same logic, where now it is the sum of all pairwise phase products, as explicitly stated in Eq. (123) since $1-\phi_i$ is equal to the sum of all other phase fields due to the constraint in Eq. (124). We can then extend these immiscible cases to handle miscible or even partially miscible fluids as we detail next, which matches the recent work of He et al. [46] in the case of three fluids. First, define pairwise miscibility values $\{\sigma_{ij}\}_{i \neq j}$ where $\sigma_{ij}=1$ means that phases i and j do not mix, while $\sigma_{ij}=0$ means that phases i and j are entirely miscible. (Note that values of σ_{ij} between 0 and 1 can also be used to deal with partial miscibility.) Then, the phase field profile can be re-expressed as:

$$|\nabla\phi_i| = \frac{4}{\xi}\phi_i \left(\sum_{j=1 \& j \neq i}^n \sigma_{ij}\phi_j \right), \quad (129)$$

to unify the case of miscible vs. immiscible fluids. We can now go through a least-squares minimization for these new profiles to find out that Eq. (127) has now changed into:

$$\frac{\partial\phi_i}{\partial t} + \mathbf{u} \cdot \nabla\phi_i = \mu_i \left\{ \nabla^2\phi_i - \frac{1}{\xi} \nabla \cdot \left[4\phi_i \left(\sum_{j=1 \& j \neq i}^n \sigma_{ij}\phi_j \right) \frac{\nabla\phi_i}{|\nabla\phi_i|} \right] \right\} + \gamma_i \quad (130)$$

for a Lagrange multiplier γ_i simplified down to:

$$\gamma_i = \frac{\chi_i}{\xi\chi} \sum_{j=1}^n \mu_i \nabla \cdot \left[4\phi_j \left(\sum_{\substack{k=1 \\ k \neq j}}^n \sigma_{jk}\phi_k \right) \frac{\nabla\phi_j}{|\nabla\phi_j|} \right]. \quad (131)$$

6.4.5 Integrating phase fields with D3Q7 CM-MRT LBM. At this point, we could use the lattice Boltzmann approach from [52] to integrate our unified model from Eq. (129) in order to advance the phase fields in time.

However, its use of a D3Q19 lattice structure requires a huge amount of memory, and the raw-moment MRT approximation for the Boltzmann collision terms is known to fail Galilean invariance [23] and to introduce too much dissipation to properly handle turbulence. Instead, we draw inspiration from Li and Desbrun [65] and Li et al. [67] who successfully developed a central-moment multiple-relaxation-time (CM-MRT) model on D3Q7 lattice structure. As their recent approach saves a lot of memory usage and improves the accuracy of collision terms, we extend it to multiple phases.

For each phase field ϕ_i of the $(n-1)$ first phases, we need to integrate in time a seven-velocity distribution $\mathbf{h}^i = \{h_0^i, \dots, h_6^i\}$ that is stored at each node of the regular grid discretizing the simulation domain. The time evolution of $\mathbf{h}^i$ for this LBM setup

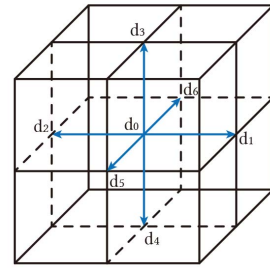


Fig. 22. D3Q7 lattice structure.

can then be written as a function of the “equilibrium” distribution function $\bar{h}^i$:

$$\bar{h}_\alpha^i = w_\alpha^d \phi_i \left(1 + \frac{\mathbf{d}_\alpha \cdot \mathbf{u}}{d_s^2} \right), \quad (132)$$

where w_α^d (resp., $\mathbf{d}_\alpha$) for $\alpha \in \{0, 1, \dots, 5, 6\}$ represents the α^{th} lattice weight (resp., lattice velocity) of our D3Q7 lattice at a given node (see Fig. 22), $\mathbf{u}$ is the local macroscopic velocity, and $d_s = \frac{1}{2}$ is the speed of sound for the D3Q7 lattice structure. The update equation in time of each distribution function reads

$$\mathbf{h}^i(\mathbf{x} + \mathbf{d}, t + 1) = \mathbf{h}^i(\mathbf{x}, t) - \mathbf{M}_h^{-1} \mathbf{S}_h \mathbf{M}_h (\mathbf{h}^i(\mathbf{x}, t) - \bar{\mathbf{h}}^i(\mathbf{x}, t)) + \mathbf{M}_h^{-1} (1 - \frac{1}{2} \mathbf{S}_h^i) \mathbf{M}_h \mathbf{H}^i, \quad (133)$$

where the matrix $\mathbf{M}_h$ is a matrix to transform the non-equilibrium distribution $\mathbf{h}^i - \bar{\mathbf{h}}^i$ into central-moment space ($\mathbf{M}_h^{-1}$ performs the inverse transform), while the diagonal matrix $\mathbf{S}_h = \text{diag}(1, 1/(4\mu_i + \frac{1}{2}), 1/(4\mu_i + \frac{1}{2}), 1/(4\mu_i + \frac{1}{2}), 1, 1, 1)$ stores the separate relaxation times of each moment — a key element of MRT models which reduces the instabilities that the single relaxation time of the original BGK model for the LBM collision operator typically generates. Because we use a central-moment space instead of a raw-moment space [52] to enforce Galilean invariance and improve accuracy of the collision term, the rows of matrix $\mathbf{M}_h = (\mathbf{M}_h^0, \mathbf{M}_h^1, \mathbf{M}_h^2, \mathbf{M}_h^3, \mathbf{M}_h^4, \mathbf{M}_h^5, \mathbf{M}_h^6)^T$ are defined as:

$$\begin{aligned} \mathbf{M}_h^0 &= (1, 1, 1, 1, 1, 1, 1)^T, \\ \mathbf{M}_h^1 &= \bar{d}_{\alpha,x}, \quad \mathbf{M}_h^2 = \bar{d}_{\alpha,y}, \quad \mathbf{M}_h^3 = \bar{d}_{\alpha,z}, \\ \mathbf{M}_h^4 &= \bar{d}_{\alpha,x}^2 - \bar{d}_{\alpha,y}^2, \quad \mathbf{M}_h^5 = \bar{d}_{\alpha,x}^2 - \bar{d}_{\alpha,z}^2, \\ \mathbf{M}_h^6 &= \bar{d}_{\alpha,x}^2 + \bar{d}_{\alpha,y}^2 + \bar{d}_{\alpha,z}^2. \end{aligned} \quad (134)$$

where $\bar{d}_{\alpha,\delta} = (\mathbf{d}_\alpha - \mathbf{u})_\delta$ for lattice index $\alpha \in \{0, \dots, 6\}$ and for $\delta \in \{x, y, z\}$ representing a vector coordinate.

Finally the source term $\mathbf{H}^i$ (last term in Eq. (133)), is expressed as:

$$\mathbf{H}_\alpha^i = \frac{4\phi_i \left(\sum_{j=1 \& j \neq i}^n \sigma_{ij} \phi_j \right)}{\xi} w_\alpha^d \mathbf{d}_\alpha \cdot \frac{\nabla \phi_i}{|\nabla \phi_i|} - \frac{\chi_i}{\xi \chi} \sum_{j=1}^n 4\phi_j \left(\sum_{k=1 \& k \neq j}^n \sigma_{jk} \phi_k \right) w_\alpha^d \mathbf{d}_\alpha \cdot \frac{\nabla \phi_j}{|\nabla \phi_j|}. \quad (135)$$

Each ϕ_i is then updated through the zeroth order moment of its associated distribution function $\mathbf{h}^i$, except for the last one which is directly deduced to enforce the constraint from Eq. (124), i.e.,

$$\phi_i = \begin{cases} \sum_\alpha h_\alpha^i & \text{if } 1 \leq i \leq n-1 \\ 1 - \sum_{j=1}^{n-1} \phi_j & \text{if } i = n \end{cases} \quad (136)$$

Note that we adopt the isotropic centered difference schemes [66] for the evaluation of $\nabla \phi_i$ and $\nabla^2 \phi_i$: they are computed respectively as:

$$\nabla \phi_i = 3 \sum_\alpha \mathbf{c}_\alpha w_\alpha \phi_i(\mathbf{x} + \mathbf{c}_\alpha, t); \quad (137)$$

$$\nabla^2 \phi_i = 6 \sum_\alpha w_\alpha [\phi_i(\mathbf{x} + \mathbf{c}_\alpha, t) - \phi_i(\mathbf{x}, t)], \quad (138)$$

except for the n^{th} phase where Eq. (136) implies:

$$\nabla \phi_n = - \sum_{i=1}^{n-1} \nabla \phi_i, \quad \nabla^2 \phi_n = - \sum_{i=1}^{n-1} \nabla^2 \phi_i. \quad (139)$$

6.4.6 Adapting HOME-LBM for multiphase flows. Now that we have detailed how phases are handled, we discuss how to simulate the flow part, i.e., the evolving velocity field (which we will encode via a distribution g) in which the phases are being advected. We will proceed in three steps: first, we will review the macroscopic model typically used for multiphase flow; then we will explain how the HOME-LBM approach of [68] can be modified not only to save memory, but also to accelerate computations and bring stability by avoiding spurious oscillations near obstacles; finally, we will provide the modified equations for distribution reconstruction and collision terms which are at the core of our contributions.

6.5 Macroscopic model and its mesoscopic discretization

The momentum and continuity equations for incompressible multiphase flows are often described macroscopically as:

$$\begin{cases} \partial \rho / \partial t + \nabla \cdot (\rho \mathbf{u}) = 0, & (140a) \\ \partial (\rho \mathbf{u}) / \partial t + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot \Pi + \mathbf{F}_s + \mathbf{F}_b, & (140b) \\ \nabla \cdot \mathbf{u} = 0, & (140c) \end{cases}$$

where ρ and $\mathbf{u}$ are respectively the spatially-varying fluid density and velocity of the fluids, p is the hydrodynamic pressure enforcing the incompressibility condition in Eq. (140c), and Π is the viscous stress tensor, while $\mathbf{F}_b$ and $\mathbf{F}_s$ are body and surface tension forces. Note that in our multiphase case, the density ρ is written in terms of the densities ρ_i and phase fields ϕ_i of all fluids as:

$$\rho(\mathbf{x}) = \sum_{i=1}^{n-1} \rho_i \phi_i(\mathbf{x}) + \rho_n \left(1 - \sum_{i=1}^{n-1} \phi_i(\mathbf{x})\right), \quad (141)$$

where the last term uses the phase field ϕ_n as defined in Eq. (136), and its gradient is: $\nabla \rho(\mathbf{x}) = \sum_{i=1}^{n-1} (\rho_i - \rho_n) \nabla \phi_i(\mathbf{x})$.

Alternative formulation. Another formulation of the momentum equation (140b) is worth noting: we can turn all the terms containing the density ρ explicitly as either new variables or external forces through the chain rule [28], yielding

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u} + p^* c_s^2 \mathbf{I} - [\nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)]) = \frac{1}{\rho} (\mathbf{F}_p + \mathbf{F}_v + \mathbf{F}_s + \mathbf{F}_b) \quad (142)$$

where three new terms involve ρ implicitly: $p^* = p / (\rho c_s^2)$ is the “normalized” pressure, $\mathbf{F}_p$ is a pressure force, and $\mathbf{F}_v$ is a viscosity force, with

$$\mathbf{F}_p = -p^* c_s^2 \nabla \rho, \quad (143)$$

$$\mathbf{F}_v = [\nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)] \nabla \rho, \quad (144)$$

the multi-fluid kinematic viscosity ν being written as a function of each phase viscosity ν_i as: $\nu = \left(\frac{1}{\nu_n} + \sum_{i=1}^{n-1} \left(\frac{1}{\nu_i} - \frac{1}{\nu_n} \right) \phi_i \right)^{-1}$. The value of this “velocity-based” formulation in Eq. (142) is explained next.

6.5.1 Velocity-based distribution function. Most early LBM works in graphics and CFD were based on a lattice-Boltzmann discretization of Eq. (140) using a distribution function f , whose zeroth-order velocity moment is the mass density ρ – considered constant since they were targeting single-phase fluid simulation. For the case of spatially-varying ρ with high-density ratios between phases, it was noticed that mixing $\mathbf{u}$ with ρ in the momentum equation was bringing numerical instability due to the rapid changes in density and thus in momentum, so discretizing Eq. (142) instead was proposed [28]: this allows for the velocity update and phase-field update to be mostly decoupled, except for the force terms based on the phase fields which influence the motion of the fluids and for the motion of the flow that carries

the phase fields along. This is precisely what was proposed recently in graphics [67, 68], using a different distribution function labeled $\mathbf{g}$ here to differentiate from the distribution $\mathbf{f}$ of traditional LBM methods: their “velocity-based” LBM discretization for incompressible two-phase flows ended up being formulated as:

$$g_i(\mathbf{x} + \mathbf{c}_i, t + 1) = g_i(\mathbf{x}, t) + \Omega_i^g(\mathbf{x}, t) + G_i(\mathbf{x}, t), \quad (145)$$

where $\mathbf{c}_i$ is a lattice velocity (see Fig. 22) and Ω_i^g and G_i are the influences on the distribution $\mathbf{g}$ from the collision operator and the external forces respectively. Macroscopic variables used in Eq. (142) can be recovered from the first three moments of distribution $\mathbf{g}$:

$$p^* = \sum_{i=0}^{q-1} g_i, \quad \mathbf{u} = \sum_{i=0}^{q-1} \mathbf{c}_i g_i + \frac{1}{2\rho} \mathbf{F}, \quad S_{\alpha\beta} = \sum_{i=0}^{q-1} (c_i^2 - \frac{1}{3} \delta_{\alpha\beta}) g_i, \quad (146)$$

where δ is the Kronecker delta function and $\mathbf{F}$ is the sum of external forces, while $S_{\alpha\beta}$ denotes the component of tensor $\mathbf{S}$ defined as

$$\mathbf{S} = \mathbf{\Pi} - \frac{1}{3} p^* \mathbf{I} = \mathbf{u} \otimes \mathbf{u} - \nu (\nabla \mathbf{u} + \nabla \mathbf{u}^T). \quad (147)$$

6.6 Adapting HOME-LBM for multiphase simulation

In the context of monophasic fluid simulation, Li et al. [68] recently proposed a very compact storage of distribution functions $\mathbf{f}$ through only their first three velocity moments ρ , $\rho\mathbf{u}$, and $\rho\mathbf{S}$ at each grid node instead of the discretization of $\mathbf{f}$ in 27 directions $\{f_i\}_{i=0..26}$ — saving nearly two thirds of the memory footprint. This *HOME-LBM* approach offers a high-order reconstruction of the distribution from its three moments to filter out non-physical oscillations in turbulent flow and near complex boundaries, as well as a moment-based collision model. Our use of a different type of “velocity-based” LBM discretization (based on Eq. (142) instead of Eq. (140b)) requires important alterations to their approach to adapt it to our multiphase context, but it will improve boundary pressure accuracy, accelerate performance, and reduce memory use for our solver. In the remainder of this section, we will assume a D3Q27 lattice discretization in 3D as typically recommended for turbulent flows to ensure numerical accuracy; we will also provide all necessary expressions for D3Q19-based distribution functions to further reduce computations with only minor accuracy loss for n -phase flow simulation.

6.6.1 Modifying distribution reconstruction from moments. For our n -phase case, the change of main variables proposed in [28] leads to a new distribution function $\mathbf{g}$ discretized as $\mathbf{g} = \{g_i\}_{i=0..26}$ on a regular grid and at discrete times, now defined as

$$g_i(\mathbf{x}, t) = w_i g(\mathbf{c}_i, \mathbf{x}, t) / \omega(\mathbf{c}_i) = \frac{f_i(\mathbf{x}, t)}{\rho(\mathbf{x}, t)} + w_i (p^*(\mathbf{x}, t) - 1). \quad (148)$$

Recall that in HOME-LBM, the encoding of the discrete distribution function $\mathbf{f}$ was done through only its first three velocity moments, i.e., ρ , $\rho\mathbf{u}$, and $\rho\mathbf{S}$. From Eq. (148), we can instead encode the distribution $\mathbf{g}$ by storing its first three moments p^* (scalar), $\mathbf{u}$ (vector), and $\mathbf{S}$ (symmetric tensor) defined in Eq. (146) — thus reducing the 27 components g_i into 10 values. Given the three moments stored in our HOME-LBM for multiphase flows, we follow [68] and derive through Mathematica a new third-order Hermite-based “filtered” reconstruction of the distribution components g_i from

the velocity moments p^* , $\mathbf{u}$, and $\mathbf{S}$. The resulting expression is:

$$\begin{aligned}
 g_i = w_i \Big[& p^* + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{\mathbf{H}^{[2]}(\mathbf{c}_i) : \mathbf{S}}{2c_s^4} \\
 & + \frac{1}{2c_s^6} \Big(\mathbf{H}_{ixxy}^{[3]}(S_{xx}u_y + 2S_{xy}u_x - 2u_xu_xu_y) \\
 & + \mathbf{H}_{ixyy}^{[3]}(S_{yy}u_x + 2S_{xy}u_y - 2u_xu_yu_y) \\
 & + \mathbf{H}_{ixxz}^{[3]}(S_{xx}u_z + 2S_{xz}u_x - 2u_xu_xu_z) \\
 & + \mathbf{H}_{ixzz}^{[3]}(S_{zz}u_x + 2S_{xz}u_z - 2u_xu_zu_z) \\
 & + \mathbf{H}_{iyzz}^{[3]}(S_{zz}u_y + 2S_{yz}u_z - 2u_yu_zu_z) \\
 & + \mathbf{H}_{iyyz}^{[3]}(S_{yy}u_z + 2S_{yz}u_y - 2u_yu_yu_z) \\
 & + \mathbf{H}_{ixyz}^{[3]}((S_{xz}u_y + S_{yz}u_x + S_{xy}u_z - 2u_xu_yu_z)) \Big] .
 \end{aligned} \tag{149}$$

6.6.2 HOME collision model. Once the three moments have been converted back into a full set of distribution functions as described above, we could now proceed as typically done in a LBM method: integrating Eq. (145), the evolution equation of $\mathbf{g}$, through Strang splitting via streaming, followed by collision evaluation, and finally external force injection, before saving the new three moments for all grid nodes after integration. However, Li et al. [68] proposed instead to convert the resulting streamed distribution back to its first three moments *right after streaming* using Eqs. (146); they then showed how to explicitly add high-order collision terms and external forces *directly* to these three moments to complete one integration timestep. We follow exactly their HOME-based approach, adapted to our new distribution $\mathbf{g}$. That is, we first perform the regular streaming process; the new three moments are then evaluated through Eqs. (146) and we denote them temporarily $p^{*'}, \mathbf{u}',$ and $\mathbf{S}'$ as they represent the moments *before* their final alteration by collision and external forces. Then, we perform the sixth-order Hermite expansion of the equilibrium central-moments and external forces recommended in [68] and compute the update of the first three moments due to the effect of the (multiple-relaxation-time) collision and forces in closed form. Using Mathematica [96], and for $\tau = 3\nu + 0.5$, the closed-form updates can be found to be expressed as:

$$p^*(\mathbf{x}, t + 1) = p^{*'} ; \tag{150}$$

$$u_\alpha(\mathbf{x}, t + 1) = u'_\alpha + \frac{1}{2\rho} F_\alpha ; \tag{151}$$

$$\begin{aligned}
 S_{xy}(\mathbf{x}, t + 1) &= (1 - \frac{1}{\tau})S'_{xy} + \frac{1}{\tau}u'_xu'_y + \frac{2\tau-1}{2\tau\rho}(F_xu'_y + F_yu'_x) ; \\
 S_{xx}(\mathbf{x}, t + 1) &= \frac{\tau-1}{3\tau}(2S'_{xx} - S'_{yy} - S'_{zz}) + \frac{1}{3}(u_x'^2 + u_y'^2 + u_z'^2) + \frac{1}{3\tau}(2u_x'^2 - u_y'^2 - u_z'^2) \\
 &\quad + \frac{1}{\rho}F_xu'_x + \frac{\tau-1}{3\tau\rho}(2F_xu'_x - F_yu'_y - F_zu'_z) ; \\
 S_{yy}(\mathbf{x}, t + 1) &= \frac{\tau-1}{3\tau}(2S'_{yy} - S'_{xx} - S'_{zz}) + \frac{1}{3}(u_x'^2 + u_y'^2 + u_z'^2) + \frac{1}{3\tau}(2u_y'^2 - u_x'^2 - u_z'^2) \\
 &\quad + \frac{1}{\rho}F_yu'_y + \frac{\tau-1}{3\tau\rho}(2F_yu'_y - F_xu'_x - F_zu'_z) ;
 \end{aligned} \tag{152}$$

$$S_{zz}(\mathbf{x}, t+1) = \frac{\tau-1}{3\tau}(2S'_{zz} - S'_{xx} - S'_{yy}) + \frac{1}{3} \left(u_x'^2 + u_y'^2 + u_z'^2 \right) + \frac{1}{3\tau} \left(2u_z'^2 - u_x'^2 - u_y'^2 \right) \\ + \frac{1}{\rho} F_z u_z' + \frac{\tau-1}{3\tau\rho} (2F_z u_z' - F_x u_x' - F_y u_y').$$

Note that these expressions do not change if a D3Q19 discretization is used instead.

6.6.3 Surface tension. While three of the four forces present in Eq. (142) are simple to evaluate (concretely, F_p is computed via Eq. (143), F_v via Eq. (144), and F_b depends on forces imposed on the fluid), the surface tension force F_s needs care in our multiphase flow context. One can base the evaluation on the work of He et al. [46] who proposed a unified total free energy $\mathcal{F}$ for the case of ternary fluids. We extend their formulation for the n -phase case through:

$$\mathcal{F}(\phi, \nabla\phi) = \int_{\Omega} \frac{3\xi}{4} \sum_{i=1}^n \beta_i |\nabla\phi_i|^2 + \frac{12}{\xi} \sum_{i=1}^n \beta_i \phi_i^2 \left(\sum_{j \neq i} \sigma_{ij} \phi_j \right)^2 d\mathbf{x}, \quad (153)$$

where β_i is the capillary coefficient for phase i . Using the virtual work method [52], we have

$$\delta\mathcal{F} = F(\phi + \delta\phi, \nabla\phi + \nabla\delta\phi) - F(\phi, \nabla\phi) = - \int_{\Omega} \mathbf{F}_s \cdot \delta\mathbf{x} d\mathbf{x}, \quad (154)$$

where $\delta\mathbf{x}$ is a virtual displacement. As a result, the surface tension force can be written as:

$$\mathbf{F}_s = \sum_{i=1}^n \left\{ \beta_i \nabla\phi_i \left[\frac{24}{\xi} \phi_i \left(\sum_{j \neq i} \sigma_{ij} \phi_j \right)^2 - \frac{3\xi}{2} \nabla^2 \phi_i \right] \right. \\ \left. + \frac{24}{\xi} \beta_i \phi_i^2 \left(\sum_{j \neq i} \sigma_{ij} \phi_j \right) \left(\sum_{j \neq i} \sigma_{ij} \nabla\phi_j \right) \right\}. \quad (155)$$

Note that for immiscible fluids, surface tension simplifies down to:

$$\mathbf{F}_s = \sum_{i=1}^n \beta_i \nabla\phi_i \left\{ \frac{24}{\xi} \phi_i (1 - \phi_i) (1 - 2\phi_i) - \frac{3\xi}{2} \nabla^2 \phi_i \right\}. \quad (156)$$

6.6.4 In/outlet and one-way coupling boundary conditions. To simulate inlet boundary conditions, we simply set the initial velocity moments of the flow based on the desired pressure p^0 and velocity $\mathbf{u}^0$ of the fluid, and set the second-order moment as

$$S_{\alpha\beta}^0 = \mathbf{u}_{\alpha}^0 \mathbf{u}_{\beta}^0, \quad (157)$$

where δ is the Kronecker delta function. To set the phase values ϕ_i^0 for the i^{th} phase, we initialize the distribution function $\mathbf{h}$ as

$$\bar{h}_{\alpha}^{i,0} = w_{\alpha}^d \phi_i^0 \left(1 + \frac{\mathbf{d}_{\alpha} \cdot \mathbf{u}^0}{d_s^2} \right) \quad (158)$$

based on Eq. (132). For outlet boundary conditions, we apply Neumann boundary conditions if the flux through an outlet wall (see inset) is going outwards, or simply set the velocity to zero, i.e.,

$$\mathbf{u}_x(\mathbf{x}_s, t) = \nabla\mathbf{u}(\mathbf{x}_p, t) \cdot \mathbf{n}_{\text{wall}} > 0 ? \mathbf{u}_x(\mathbf{x}_p, t) : 0, \\ \mathbf{u}_y(\mathbf{x}_s, t) = 0, \quad (159)$$

from which the second-order moment is again approximated as in Eq. (157). For the phase field equation, we implement an outlet boundary condition for the distribution function $\mathbf{h}$ on the wall based on [73], written as:

$$\mathbf{h}(\mathbf{x}_s, t) = \mathbf{h}(\mathbf{x}_p, t). \quad (160)$$

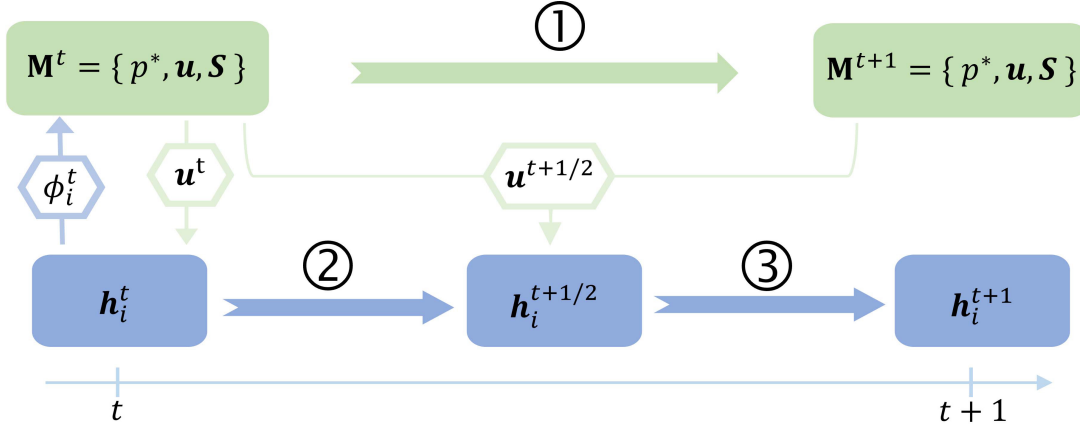


Fig. 23. **Time integration.** A timestep from t to $t+1$ of our solver is achieved through (1) first, advancing the moments $\mathbf{M}^t$ in time to $t+1$ accounting for the forces induced by the n phase fields ϕ_i^t known at time t (top); then (2) the phase fields are updated in two substeps (bottom), first advected by the macroscopic velocity $\mathbf{u}^t$, then by the average velocity $\mathbf{u}^{t+1/2}$. Hexagonal labels indicate dependencies between flow and phase-field equations.

Algorithm 3: Pseudocode of our kinetic multifluid solver.

```

 $t \leftarrow 0$ ;
Initialize  $\phi$ , as well as the velocity moments  $p$ ,  $\mathbf{u}$ , and  $\mathbf{S}$ ;
Init $\nabla\phi()$                                 ▶ Eq. (137) and (139)
InitForces()                              ▶ Eq. (143), (144) and (155)
Initialize distribution  $\mathbf{h}$                 ▶ Eq. (132)
while  $t < T$  do
  In/OutletBoundaryTreatmentForFlow()      ▶ Sec. 6.6.4
  HOME-LBM-FlowSolver()                    ▶ Sec. 6.6
   $k \leftarrow 0$ ;
  while  $k < 2$  do
    PerformPhaseCollision()                 ▶ Eq. (133)
    In/OutletBoundaryTreatmentForPhase()    ▶ Sec. 6.6.4
    PerformPhaseStream()                    ▶ Eq. (133)
    UpdatePhaseField()                      ▶ Eq. (136)
    WettingBoundaryConditionForPhase()      ▶ Sec. 6.6.5
    Compute $\nabla\phi()$                        ▶ Eq. (137) and (139)
     $k \leftarrow k + 1$ ;
  end
  CalculateForces()                         ▶ Eq. (143), (144) and (155)
   $t \leftarrow t + 1$ ;
end

```

6.6.5 Putting it all together. We can now complete the description of HOME-LBM based simulation of multiphase flows by discussing how the solvers for respectively the phases and the flow are coupled.

We have reviewed in Sec. 6.4.3 how to deal with the LBM-based n phase fields, indicating where the n fluids are located in the domain. Note that the only other field they require to be integrated in time is the macroscopic velocity field in which they are advected. Conversely, we have discussed in Sec. 6.4.6 how to use a lightweight and efficient

flow integration to compute a multiphase version of HOME-LBM using a velocity-based distribution to handle the velocity field through LBM. This time, forces based on the phase densities (hence, on all the phases) are needed to properly integrate the velocity field. We thus have all we need to couple the simulation of the n phases $\{\phi_i\}_n$ and the flow simulator to integrate multifluid flows in time.

However, we can add a few other components to this basic simulation framework. First, to improve the accuracy and stability of the phase field equations, one can follow the recommendation in [65] by employing an eight-times finer resolution for the phase fields compared to the grid for distribution g (i.e., each spatial dimension uses a twice finer discretization). This changes the time integration scheme slightly since the phase fields now need two substeps, see Fig. 23. Moreover, while Li and Desbrun [65] had the two grids staggered with respect to each other to filter spurious oscillations of the velocity field, the filtering brought by the use of a HOME-LBM solver (based only on the first three moments) is enough in practice, and no staggering is required. So we can keep the two grids aligned as this reduces the need for grid interpolation, hence accelerating the final solver.

Second, we can also incorporate wetting (i.e., the way fluids cling to or separate from obstacles) using the same treatment as in two-phase wetting [67], for all the phases individually — i.e., the boundary conditions to induce hydrophobic or hydrophilic wetting near solid boundaries are set phase by phase, ignoring the fact that phases are globally bound by Eq. (124). While this may be inaccurate, we could not find modeling papers discussing how to properly deal with wetting for multiple phases/fluids.

Finally, coupling with complex solids is handled through cut cells, exactly as proposed in [65], and so are wall boundary conditions. However, we no longer need their “hybrid” bounce-back approach, which was designed to filter the pressure near solids: the use of HOME-LBM removes this issue, and one can stick to the original (simpler) bounce-back approach, which results in non-oscillatory pressure behavior near boundaries in practice.

7 GPU-Oriented Optimization

LBM has become one of the most widely adopted frameworks for high-performance fluid simulation, particularly in large-scale or real-time applications, due to its inherent parallelism. Unlike traditional incompressible flow solvers based on the Navier–Stokes equations [11, 87], which often require the solution of global linear systems such as the pressure Poisson equation, LBM evolves the fluid states through strictly local operations: collision and streaming. This locality avoids a global linear solve and supports local, regular kernels during each time step, making LBM particularly well suited for massively parallel architectures, especially modern graphics processing units (GPUs) [6, 17, 69, 79, 90].

However, achieving high efficiency in GPU-based LBM simulations remains non-trivial. Performance is often limited by several factors, including memory bandwidth saturation, control flow divergence, and irregular memory access patterns. These issues become even more pronounced in high-fidelity kinetic solvers, where more sophisticated collision models, multiphysics couplings, and complex boundary treatments introduce additional computational complexity and memory overhead.

In this chapter, we present a systematic overview of GPU-oriented optimization techniques for large-scale LBM simulations, with a particular focus on fluid-only and fluid-solid coupling scenarios. We begin by analyzing the performance characteristics of LBM and identifying the primary bottlenecks of baseline GPU implementations. Building on this analysis, we then discuss practical design considerations, including data layout strategies, memory access patterns, and kernel design techniques that are critical for achieving high throughput on modern GPUs. Furthermore, we present a set of implementation-level optimization strategies. Together, these techniques demonstrate how careful

co-design of numerical methods and execution models can substantially improve the performance, scalability, and memory efficiency of LBM solvers.

7.1 Performance Characteristics on GPUs

7.1.1 Memory-Bound Nature of Conventional GPU LBM Kernels. A defining characteristic of LBM on modern GPU architectures is its relatively low arithmetic intensity, i.e., the ratio of floating-point operations (FLOPs) to memory traffic. During each timestep, every lattice node must access and update multiple particle distribution functions (f_i), resulting in substantial memory movement relative to the amount of computation performed. As a result, within the CUDA execution model [95], many conventional GPU LBM kernels implementations are limited by memory bandwidth rather than computational throughput. This behavior becomes more pronounced for higher-order lattice velocity discretizations such as D3Q19 (Fig. 4(b)) and D3Q27 (Fig. 4(c)), where dozens of distribution functions must be loaded, processed, and streamed every timestep. Although collision operators may involve a moderate amount of arithmetic, the overall performance is typically dominated by memory accesses associated with distribution storage and transport.

In addition, many LBM implementations employ double-buffering (ping-pong) schemes to avoid read–write hazards, ensuring correctness and implementation simplicity. While robust and widely adopted, this approach significantly increases memory footprint, particularly for large-scale simulations. Achieving high performance therefore requires maximizing effective memory throughput through coalesced memory accesses, minimizing redundant data transfers, and improving data locality across streaming multiprocessors (SMs).

7.1.2 Control Flow Divergence and Spatial Irregularity. While LBM exhibits highly regular and branch-free computation in homogeneous regions, the presence of complex solid boundaries introduces execution irregularity.

First, boundary treatments such as bounce-back and immersed boundary (IB) methods often require conditional branching to distinguish between fluid and boundary nodes. As a result, threads within the same warp may follow different execution paths, forcing the GPU to serialize branch execution and reducing SIMT efficiency. Second, many fluid–solid coupling methods involve irregular memory access patterns. For example, IB-based methods require interpolation and force spreading between Lagrangian surface samples and the Eulerian lattice nodes. These gather–scatter operations, as shown in Fig. 25, result in uncoalesced memory accesses and reduced memory efficiency. Furthermore, the computational workload associated with boundary processing is often highly non-uniform in space. Regions containing dense geometric features require substantially more computation than fluid regions, leading to uneven workload distribution across thread blocks. As some blocks complete much earlier than others, the remaining heavily loaded blocks dominate kernel execution time, producing a tail effect and reducing overall device utilization.

7.1.3 Register Pressure and Occupancy. Recent high-fidelity LBM solvers often employ advanced collision models (see Chapter 3 for more details) to improve numerical stability, accuracy, and physical fidelity. Compared with simple BGK formulations, these models introduce significant computational complexity and increased pressure on hardware resources. In particular, transformations between distribution space and central-moment space involve dense matrix-vector operations and high-order polynomial evaluations, requiring a large number of registers per thread. On modern CUDA architectures [95], the amount of registers per SM is fixed. Consequently, increasing register usage per thread reduces the number of active threads and warps, thereby lowering hardware occupancy. This further introduces a fundamental trade-off in kernel design. On one hand, kernel fusion can improve memory efficiency by reducing intermediate global memory accesses and increasing data locality. On the other hand, overly fused kernels result in

higher register usage and lower occupancy. Therefore, achieving high performance requires carefully balancing kernel fusion and kernel decomposition to jointly optimize memory efficiency, register usage, and hardware utilization.

7.2 Memory-Centric Optimization Strategies

Since LBM is typically memory-bandwidth limited on modern GPUs, optimizing an LBM solver is often less about increasing floating-point throughput and more about reducing and regularizing data movement. From a system perspective, this can be approached in two complementary ways. The first is to improve the efficiency of memory accesses so that the data already being transferred makes better use of the available bandwidth. The second is to reduce the amount of data that must be transferred in the first place, either by adopting a more compact simulation representation or by storing selected quantities at lower precision.

7.2.1 Data Layout: Maximizing Effective Memory Bandwidth. The efficiency of global memory access on GPUs is strongly affected by memory coalescing. Ideally, neighboring threads in a warp should access contiguous memory locations, allowing their requests to be served by 32-byte memory transactions. This consideration is particularly important for LBM because each lattice node stores multiple distribution components, and these components must be repeatedly loaded and written during collision and streaming.

As shown in Fig.24, a straightforward CPU-oriented implementation often adopts an array-of-structures (AoS) layout [94], where all distribution components associated with a lattice node are stored contiguously (e.g., $f_0, f_1, \dots, f_{26}$ per node). While this layout can be convenient and cache-friendly on CPUs, it leads to inefficient memory access patterns on GPUs, as threads processing neighboring nodes access strided memory locations. For GPU implementations, a structure-of-arrays (SoA) layout [89] is therefore commonly preferred, in which each distribution component is stored in a separate contiguous array (e.g., all f_0 values, followed by all f_1 values, $\dots$, all f_{26} values). When neighboring threads access the same distribution component for neighboring nodes, their memory requests become contiguous, enabling more efficient global memory transactions. This simple change is often one of the most important first steps in porting an LBM solver to the GPU.

Building upon this idea, more advanced layouts further improve memory locality by grouping multiple lattice nodes into blocks or tiles, as in collected SoA (CSoA) [13, 17] or block-structured (usually $8 \times 8 \times 8$ in 3D) layouts. These layouts combine the coalescing advantages of SoA with better cache locality within a local spatial region. They are especially useful for three-dimensional simulations and for scenes with complex boundaries, where pure global ordering may lead to less coherent access patterns near irregular geometry.

7.2.2 Reducing Memory Traffic through Compact Representations. While data-layout optimizations improve how memory is accessed, the total amount of data that must be stored and transferred is not reduced. To address the bandwidth bottleneck more fundamentally, recent approaches [18, 68] focus on compressing the representation itself and reducing the per-node memory footprint.

Moment-Space Compactness: As discussed in Chapter 6, instead of storing the full set of 27 distribution functions (f_i) per lattice node, one can store a compact set of velocity moments (e.g., $\rho, \rho u, \rho S$), where the per-node memory footprint is significantly reduced. In a typical D3Q27 configuration, this reduces the storage requirement by approximately a factor of three [68].

On-the-fly Distribution Reconstruction: Under the moment-encoded representation, distribution functions are no longer explicitly stored in global memory. Instead, they are reconstructed on-the-fly when needed. In practice, this reconstruction is performed at the block level prior to the streaming step. For each thread block, the required

box), most threads execute only the standard fluid update, while a small subset of threads located near solid boundaries must additionally perform interpolation, force evaluation, momentum exchange, or boundary reconstruction. This imbalance results in irregular workloads and increased warp divergence.

To mitigate this issue, many high-performance implementations adopt a boundary-centric parallelization strategy (see Fig. 25 green and red boxes) [17], in which the unit of parallelism is shifted from Eulerian grid nodes to Lagrangian boundary samples. In this formulation, each thread processes a solid sample and computes its contribution to the surrounding fluid nodes. This transformation converts an irregular per-node workload into a more uniform per-sample workload, improving load balance and reducing execution variance across threads. Since threads within a warp now perform similar operations, warp divergence is also reduced.

To further reduce computational overhead, active boundary lists are often constructed to compactly store only the subset of lattice nodes or boundary samples participating in fluid–solid interactions. During each simulation step, GPU kernels operate only on these active elements rather than the entire computational domain. This compaction strategy avoids unnecessary computation on inactive regions and significantly improves efficiency when the solid boundary occupies only a small portion of the simulation domain.

Finally, contributions from multiple boundary samples to the same fluid node must be accumulated consistently. This is commonly handled using hardware-supported atomic operations (e.g., `atomicAdd`). Atomics provide a simple and correct accumulation mechanism, but can introduce serialization and may become a bottleneck when many boundary samples target the same lattice node.

7.3.2 Geometry-Decoupled Scheme. Beyond improving boundary processing itself, recent work has explored restructuring the overall simulation pipeline to reduce the impact of geometric complexity on GPU performance. A key limitation of conventional implementations is the reliance on a merged “mega-kernel” that combines streaming, collision, and boundary handling within a single execution kernel. While such designs minimize kernel launch overhead, they tightly couple regular fluid computation with irregular boundary processing, resulting in significant control-flow divergence and reduced SIMT efficiency.

To address this issue, high-performance implementations adopt a geometry-decoupled algorithm design [18]. The central idea is to separate regular fluid computation from irregular boundary processing. More specifically, the simulation is decomposed into two stages:

Dense Fluid Kernel: A highly optimized, branch-free kernel that performs collision and streaming over the entire lattice domain. By eliminating boundary-related conditionals, this kernel maintains uniform control flow within each warp, enabling high instruction throughput and efficient utilization of the GPU’s SIMT execution model.

Sparse Boundary Correction Kernel: A secondary kernel processes only lattice nodes affected by solid boundaries. Rather than embedding geometry-dependent logic into the main fluid update, boundary effects are computed separately and applied as fluid state corrections. Since the number of boundary nodes is typically much smaller than the total number of fluid nodes, the performance impact of irregular computation remains limited. Besides improving warp execution efficiency, the computational cost of boundary processing becomes proportional to the number of boundary nodes rather than the total simulation size, improving scalability for large domains containing complex geometries.

7.4 Advanced Optimization Directions

The topics discussed above by no means exhaust the space of GPU optimization techniques for LBM. We conclude this chapter by briefly mentioning several additional optimization directions that are beyond the scope of these course notes.

Grid Refinement and Adaptive Resolution. Multi-resolution and adaptive LBM methods have been developed to concentrate computational effort in regions requiring higher accuracy while maintaining a coarser discretization elsewhere. Such approaches can substantially reduce memory consumption and computational cost for large-scale simulations [76, 84].

Mixed-Precision and Quantized Representation. Unlike generic compression schemes, quantization in LBM must be carefully designed to preserve numerical stability, especially in turbulent or high-Reynolds-number regimes. Recent work has explored storing selected fluid variables using reduced-precision representations. By reducing memory footprint and bandwidth requirements, these methods can improve performance while maintaining acceptable numerical accuracy [18].

Multi-GPU and Distributed LBM. For extremely large simulation domains, LBM has also been extended to multi-GPU and distributed-memory systems, which is particularly well suited for large-scale parallel deployment [12, 17, 49, 80].

8 Applications

Fluid simulation plays a central role in computer graphics, enabling realistic modeling of natural phenomena such as wind, smoke, and fire. Beyond visual fidelity, high-performance fluid simulation is increasingly important for downstream applications in games and visual effects, as well as in broader domains such as climate and environmental science, including weather forecasting and air pollution modeling. It is also critical in engineering and industrial design, where it supports tasks such as fluid shape optimization, unmanned aerial vehicle design and evaluation, underwater robot control, and reinforcement learning. Compared with macroscopic simulation methods such as Stable Fluids, FLIP, and SPH, LBM offers its own advantages in supporting a wide range of fluid phenomena, from laminar to turbulent flows. More importantly, for many downstream applications, such as CAD, industrial design, and reinforcement learning, computational performance is a key factor for rapid iteration. LBM is particularly well suited to efficient GPU implementation, since its streaming and collision operations are highly local, making it possible to fully exploit parallel hardware and significantly accelerate these applications. In this section, we present several representative applications, including aerodynamic analysis and industrial design, swimmer control and learning, and real-time effects in games.

Industrial design. A key application of LBM is the construction of high-performance virtual wind-tunnel facilities for aerodynamic analysis and industrial design. By describing fluid motion through local streaming and collision operations on structured lattices, LBM provides a highly parallel computational framework for simulating weakly compressible turbulent flows around complex three-dimensional geometries, such as racing cars, aircraft components, and urban structures. Its intrinsic locality makes it particularly suitable for GPU acceleration and adaptive or multiresolution grid strategies, enabling efficient resolution of fine-scale vortical structures, wake dynamics, surface pressure distributions, and aerodynamic loads. In this context, LBM-based virtual wind tunnels can predict key aerodynamic quantities, including drag, lift, pressure coefficients, and flow-separation patterns, prior to physical prototyping or experimental wind-tunnel testing. This significantly reduces design cost and turnaround time while supporting rapid design iteration.

and performance optimization. Therefore, LBM can serve as a core simulation engine for industrial-scale aerodynamic testing platforms that require both physical fidelity and high computational efficiency.

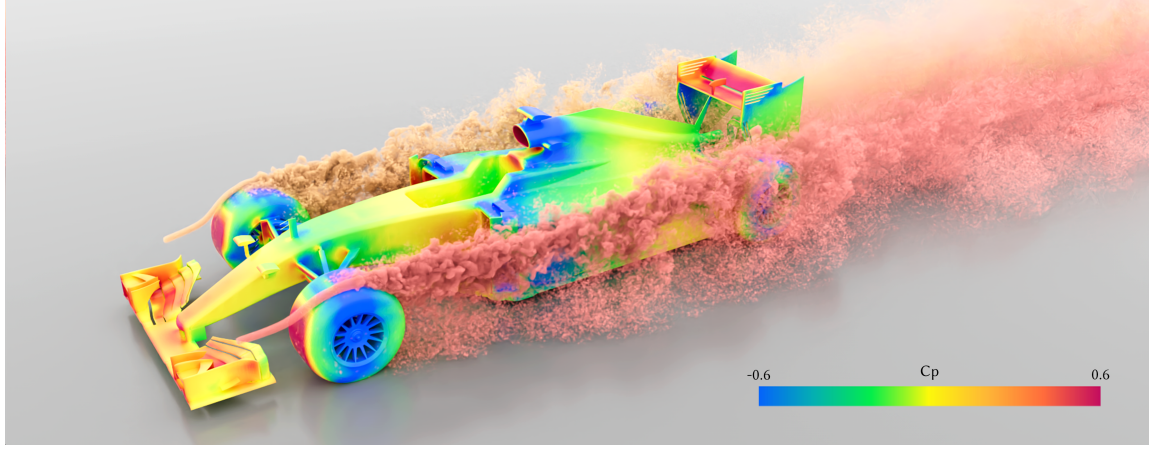


Fig. 26. Aerodynamics of an F1 racing car [74].

Virtual environments for embodied robot learning. Physics simulation engines are becoming fundamental infrastructure for embodied intelligence, as they provide scalable, controllable, and physically grounded environments in which agents can learn perception–action policies before deployment in the real world. Compared with purely kinematic or data-driven environments, physics-based simulators explicitly model the dynamic coupling among the agent, its actuation, the surrounding medium, and environmental constraints. This capability is particularly important for robots whose locomotion and control performance strongly depend on complex fluid interactions, such as aerial vehicles, aquatic robots, and torus-shaped swimmers. In these scenarios, the simulator is not merely a visualization tool but a closed-loop training platform that provides physically meaningful feedback, including aerodynamic or hydrodynamic forces, vortex structures, wake evolution, pressure distributions, and near-body flow effects.

One representative application is the virtual design and evaluation of flight controllers for unmanned aerial vehicles operating in complex airflow environments, as shown in Fig. 27. A high-performance hybrid simulator can capture the interaction among UAV motion, rotor-induced airflow, and nearby obstacles, such as walls, tubes, or confined passages. Such a virtual testbed enables controller evaluation under challenging conditions where real-world experiments are costly, risky, or difficult to reproduce. It also supports systematic testing of trajectory tracking, stability, disturbance rejection, and robustness under strong fluid–structure interactions. By combining efficient fluid simulation with empirical near-rotor modeling, the simulator can support practical controller development while maintaining computational efficiency.

Another important application lies in the learning-based control of soft aquatic robots, as shown in Fig. 28. For free-form soft swimmers, locomotion emerges from the coupled dynamics of body deformation, vortex shedding, wake interaction, and hydrodynamic force generation. When integrated with reinforcement learning or morphology–control co-design frameworks, an LBM-based simulator can provide high-fidelity physical feedback for different body shapes

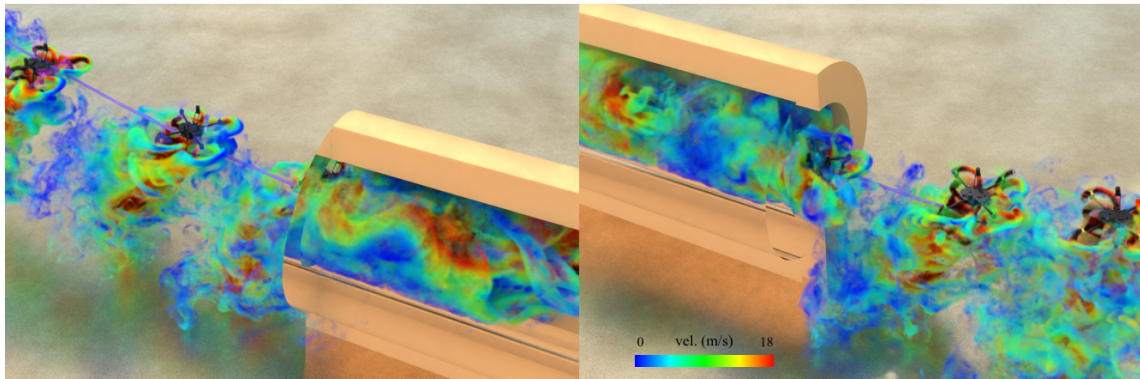


Fig. 27. **A quadcopter swarm executing sequential flight through a cylindrical tube [93].**

and actuation patterns, enabling agents to discover effective swimming gaits without relying on manually designed control rules or overly simplified fluid models.

The LBM-based simulation framework can also be used to train fluid-interacting virtual agents, such as flapping-wing robots or ornithopters. In such systems, effective locomotion depends on exploiting unsteady aerodynamic effects induced by wing motion, including vortex generation, wake capture, and force feedback from the surrounding flow. LBM provides a computationally efficient kinetic framework for resolving these phenomena while retaining a highly local computational structure, making it especially well suited for GPU acceleration. Moreover, a localized moving-domain strategy allows the fluid domain to follow the agent, avoiding the prohibitive cost of simulating a large global environment while preserving the local flow dynamics essential for control learning. This makes LBM particularly attractive for training embodied agents that must actively interact with surrounding fluids to navigate, maneuver, and exploit environmental forces.

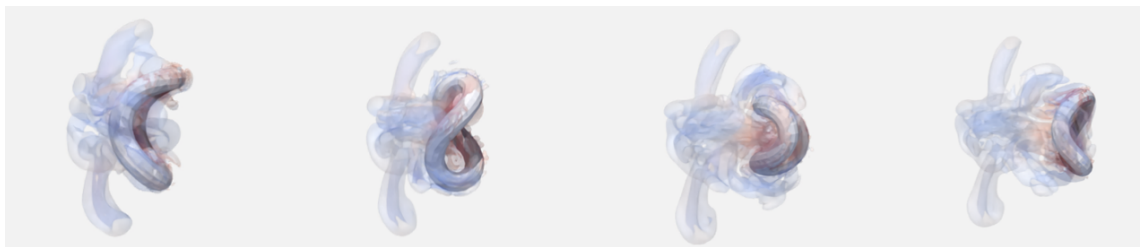


Fig. 28. **The torus-shaped swimmer achieves propulsion through a periodic deformation cycle, dynamically balancing body-fluid momentum exchange [50].**

Together, these applications demonstrate that high-performance physics engines can serve as core simulation backbones for embodied robot learning. They enable large-scale policy training, safe evaluation of failure cases, rapid testing of robot designs, and systematic analysis of complex fluid-mediated behaviors. In particular, LBM-based fluid simulation offers a promising foundation for constructing virtual training environments in which embodied agents can learn physically plausible control strategies for aerial and aquatic locomotion tasks.

Visual effects in games. Wind fields, fog, and smoke are critical features in games for creating dynamic weather, environmental interaction, and gameplay mechanics. They enrich the player's experience, help drive narrative atmosphere, and provide additional opportunities for interaction and strategy. For decades, the game industry has relied heavily on Stable Fluids because of its conceptual simplicity and ease of implementation. However, as discussed earlier, enforcing a divergence-free velocity field requires solving a large linear system during the projection step. Even with matrix-free solvers, the iterative solution process remains a major performance bottleneck due to kernel launch overhead and frequent memory reads and writes. In contrast, recent advances in HOME-LBM make it possible to combine collision and streaming within a single kernel invocation, while storing and loading only moment variables in memory and decoding the distribution functions on the fly. This design significantly reduces memory traffic and improves computational efficiency. As a result, HOME-LBM is particularly well suited for real-time applications, especially games, where both minimal data transfer and visually rich, turbulent fluid dynamics are essential, as shown in Fig. 29.



Fig. 29. The real-time LBM smoke driven by interaction with character motion.

9 Conclusion

In these course notes, we tried to offer a coherent view of the Lattice Boltzmann Method (LBM) as a practical kinetic framework for fluid simulation, spanning fundamentals of kinetic theory, collision modeling, boundary conditions, refinement strategies, moment representations, and GPU-oriented implementations. LBM has progressed from a robust tool for single-phase flows to a versatile method for two-phase and more general n -phase fluids, turbulent regimes, and tightly coupled fluid–solid interaction. These advances increasingly position LBM not merely as an alternative discretization of continuum equations, but as a scalable mesoscopic modeling method enabling both higher-fidelity physics and production-ready performance, supporting deployment in industrial simulation software and powering virtual dynamic environments for interactive applications such as robotics training and embodied AI.

Looking ahead, kinetic theory offers a broader, multi-scale opportunity: by linking molecular, mesoscopic, and macroscopic descriptions in a unified framework, it may provide a powerful computational foundation for next-generation simulations. Beyond visual effects and engineering analysis, this capability can facilitate mechanism-level investigations in foundational areas such as materials science — covering transport processes, phase change, and interfacial dynamics — while also enabling data-rich, interactive dynamic environments for applied domains including embodied intelligence, robotics training, and world-model construction, where scalable and physically grounded simulators are essential.

References

- [1] Saad Adam, Farzaneh Hajabdollahi, and Kannan N. Premnath. 2021. Cascaded lattice Boltzmann modeling and simulations of three-dimensional non-Newtonian fluid flows. *Computer Physics Communications* 262 (2021), 107858. doi:10.1016/j.cpc.2021.107858
- [2] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.* 31, 4, Article 62 (July 2012), 8 pages.
- [3] Stefano Allegretti, Federico Bolelli, and Costantino Grana. 2019. Optimized block-based algorithms to label connected components on GPUs. *IEEE Trans. Parallel and Distributed Systems* 31, 2 (2019), 423–438.
- [4] Thomas Astoul, Gauthier Wissocq, Jean-François Boussuge, Alois Sengissen, and Pierre Sagaut. 2021. Lattice Boltzmann Method for Computational Aeroacoustics on Non-uniform Meshes: a direct grid coupling approach. *J. Comput. Phys.* 447 (2021), 110667.
- [5] Mohammadmehdi Ataei, Vahid Shaayegan, Franco Costa, Sejin Han, Chul B Park, and Markus Bussmann. 2021. LBfoam. *Comp. Phys. Comm.* 259 (2021), 107698.
- [6] Peter Bailey, Joe Myre, Stuart DC Walsh, David J Lilja, and Martin O Saar. 2009. Accelerating lattice Boltzmann fluid flow simulations using graphics processors. In *2009 international conference on parallel processing*. IEEE, 550–557.
- [7] Roberto Benzi, Sauro Succi, and Massimo Vergassola. 1992. The lattice Boltzmann equation: theory and applications. *Physics Reports* 222, 3 (1992), 145–197.
- [8] Prabhu Lal Bhatnagar, Eugene P Gross, and Max Krook. 1954. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. *Physical review* 94, 3 (1954), 511.
- [9] Simon Bogner. 2017. *Direct numerical simulation of liquid-gas-solid flows based on the lattice Boltzmann method*. Ph. D. Dissertation. Erlangen-Nürnberg University.
- [10] M’hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. 2001. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids* 13, 11 (2001), 3452–3459. doi:10.1063/1.1399290
- [11] Robert Bridson. 2015. *Fluid simulation for computer graphics*. AK Peters/CRC Press.
- [12] Enrico Calore, Alessandro Gabbana, Jiri Kraus, Elisa Pellegrini, Sebastiano Fabio Schifano, and Raffaele Tripiccone. 2016. Massively parallel lattice-Boltzmann codes on large GPU clusters. *Parallel Comput.* 58 (2016), 1–24.
- [13] Enrico Calore, Alessandro Gabbana, Sebastiano Fabio Schifano, and Raffaele Tripiccone. 2019. Optimization of lattice Boltzmann simulations on heterogeneous computers. *The International Journal of High Performance Computing Applications* 33, 1 (2019), 124–139.
- [14] Carlo Cercignani. 1975. Theory and application of the Boltzmann equation. *Elsevier* (1975).
- [15] Sydney Chapman and Thomas George Cowling. 1990. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge university press.
- [16] Hudong Chen, Olga Filippova, James Hoch, Kim Molvig, Richard Shock, Christopher Teixeira, and Raoyang Zhang. 2006. Grid Refinement in Lattice Boltzmann Methods Based on Volumetric Formulation. *Physica A: Statistical Mechanics and its Applications* 362, 1 (2006), 158–167. doi:10.1016/j.physa.2005.09.036
- [17] Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. 2022. GPU Optimization for High-Quality Kinetic Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* 28, 9 (2022), 3235–3251.
- [18] Yixin Chen, Wei Li, David IW Levin, and Kui Wu. 2026. High-Performance Moment-Encoded Lattice Boltzmann Method with Stability-Guided Quantization. *arXiv preprint arXiv:2602.05295* (2026).
- [19] Christophe Coreixas, Gauthier Wissocq, Guillaume Puigt, Jean-François Boussuge, and Pierre Sagaut. 2017. Recursive regularization step for high-order lattice Boltzmann methods. *Phys. Rev. E* 96 (2017), 033306. Issue 3. doi:10.1103/PhysRevE.96.033306
- [20] Alessandro De Rosi. 2017. Nonorthogonal central-moments-based lattice Boltzmann scheme in three dimensions. *Phys. Rev. E* 95, 1 (2017), 013310.
- [21] Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023. Fluid Simulation on Neural Flow Maps. *ACM Trans. Graph.* 42, 6, Article 248 (Dec. 2023), 21 pages.
- [22] Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*. 61–76.
- [23] Dominique d’Humières. 2002. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Phil. Trans. R. Soc. A* 360 (2002), 437–451. Issue 1792.

- [24] Dominique d'Humières and Irina Ginzburg. 2009. Viscosity independent numerical errors for Lattice Boltzmann models: From recurrence equations to “magic” collision numbers. *Computers & Mathematics with Applications* 58, 5 (2009), 823–840. doi:10.1016/j.camwa.2009.02.008 Mesoscopic Methods in Engineering and Science.
- [25] Lin Ding and Xiaopei Liu. 2025. A Hybrid LBM-FVM Solver for Simulating Multiphase Multi-Component Fluids. *IEEE transactions on visualization and computer graphics* (2025).
- [26] Suchuan Dong. 2017. Wall-bounded multiphase flows of N immiscible incompressible fluids: Consistency and contact-angle boundary condition. *J. Comput. Phys.* 338 (2017), 21–67.
- [27] François Dubois, Tony Février, and Benjamin Graille. 2015. On the stability of a relative velocity lattice Boltzmann scheme for compressible Navier–Stokes equations. *Comptes Rendus. Mécanique* 343, 10–11 (2015), 599–610. doi:10.1016/j.crme.2015.07.010
- [28] Abbas Fakhari, Travis Mitchell, Christopher Leonardi, and Diogo Bolster. 2017. Improved locality of the phase-field lattice-Boltzmann model for immiscible fluids at high density ratios. *Physical Review E* 96, 5 (2017), 053301.
- [29] Marco A. Ferrari, Waine B. de Oliveira Jr., Alan Lugarini, Admilson T. Franco, and Luiz A. Hegele Jr. 2023. A graphic processing unit implementation for the moment representation of the lattice Boltzmann method. *International Journal for Numerical Methods in Fluids* 95, 7 (2023), 1076–1089.
- [30] Olga Filippova and Dieter Hänel. 1998. Grid Refinement for Lattice-BGK Models. 147, 1 (1998), 219–228. doi:10.1006/jcph.1998.6089
- [31] James D. Foley. 1996. *Computer graphics: principles and practice*. Addison-Wesley.
- [32] Nicolò Frapolli. 2017. *Entropic lattice Boltzmann models for thermal and compressible flows*. Ph.D. Dissertation. ETH Zurich.
- [33] Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Trans. Graph.* 36, 6, Article 222 (Nov. 2017), 12 pages.
- [34] Martin Geier, Andreas Greiner, and Jan G. Korvink. 2006. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Phys. Rev. E* 73 (Jun 2006), 066705. Issue 6. doi:10.1103/PhysRevE.73.066705
- [35] M. Geier, A. Greiner, and J. G. Korvink. 2009. A factorized central moment lattice Boltzmann method. *The European Physical Journal Special Topics* 171, 1 (01 Apr 2009), 55–61. doi:10.1140/epjst/e2009-01011-1
- [36] Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. 2015. The Cumulant Lattice Boltzmann Equation in Three Dimensions: Theory and Validation. *Computers & Mathematics with Applications* 70, 4 (2015), 507–547.
- [37] Félix Gendre, Denis Ricot, Guillaume Fritz, and Pierre Sagaut. 2017. Grid refinement for aeroacoustics in the lattice Boltzmann method: A directional splitting approach. *Phys. Rev. E* 96 (2017), 023311. Issue 2.
- [38] Irina Ginzburg. 2008. Consistent lattice Boltzmann schemes for the Brinkman model of porous flow and infinite Chapman-Enskog expansion. *Phys. Rev. E* 77 (Jun 2008), 066704. Issue 6. doi:10.1103/PhysRevE.77.066704
- [39] Irina Ginzburg. 2012. Truncation Errors, Exact And Heuristic Stability Analysis Of Two-Relaxation-Times Lattice Boltzmann Schemes For Anisotropic Advection-Diffusion Equation. *Communications in Computational Physics* 11, 5 (2012), 1439–1502. doi:10.4208/cicp.211210.280611a
- [40] Irina Ginzburg, Dominique d'Humières, and Alexander Kuzmin. 2010. Optimal Stability of Advection-Diffusion Lattice Boltzmann Models with Two Relaxation Times for Positive/Negative Equilibrium. *Journal of Statistical Physics* 139, 6 (01 Jun 2010), 1090–1143. doi:10.1007/s10955-010-9969-9
- [41] Ehsan Goraki Fard. 2015. *Cumulant LBM approach for Large Eddy Simulation of Dispersion Microsystems*. Ph.D. Dissertation. Technische Universität Braunschweig.
- [42] John Gounley, Madhurima Vardhan, Erik W. Draeger, Pedro Valero-Lara, Shirley V. Moore, and Amanda Randles. 2022. Propagation Pattern for Moment Representation of the Lattice Boltzmann Method. *IEEE Trans. Parallel Distrib. Syst.* 33, 3 (2022), 642–653.
- [43] Philip M Gresho and Robert L Sani. 1987. On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids* 7, 10 (1987), 1111–1145.
- [44] Yulong Guo, Xiaopei Liu, and Xuemiao Xu. 2017. A Unified Detail-Preserving Liquid Simulation by Two-Phase Lattice Boltzmann Modeling. *IEEE Trans. Vis. Comp. Graph.* 23, 5 (2017), 1479–1491.
- [45] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. 2002. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Physical Review E* 65, 4 (Apr 2002), 046308. doi:10.1103/PhysRevE.65.046308
- [46] Qiang He, Yongjian Li, Weifeng Huang, Yang Hu, Decai Li, and Yuming Wang. 2020. A unified lattice Boltzmann model for immiscible and miscible ternary fluids. *Computers & Mathematics with Applications* 80, 12 (2020), 2830–2859.
- [47] William Henry. 1803. Experiments on the quantity of gases absorbed by water at different temperatures and under different pressures. In *Phil. Trans. R. Soc.* 28–274.
- [48] Cyril W Hirt and Billy D Nichols. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comput. Phys.* 39, 1 (1981), 201–225.
- [49] Pei-Yao Hong, Li-Min Huang, Li-Song Lin, and Chao-An Lin. 2015. Scalable multi-relaxation-time lattice Boltzmann simulations on multi-GPU cluster. *Computers & Fluids* 110 (2015), 1–8.
- [50] Changyu Hu, Yanke Qu, Qiuhan Yang, Xiaoyu Xiong, Kui Wu, Wei Li, and Tao Du. 2025. Learning to Control Free-Form Soft Swimmers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=Z01gNsO9SW>
- [51] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.* 37, 4, Article 150 (2018).
- [52] Yang Hu, Decai Li, and Qiang He. 2020. Generalized conservative phase field model and its lattice Boltzmann scheme for multicomponent multiphase flows. *International Journal of Multiphase Flow* 132 (2020), 103432.

- [53] C Huber, Y Su, CT Nguyen, Andrea Parmigiani, Helge M Gonnermann, and J Dufek. 2014. A new bubble dynamics model to study bubble growth, deformation, and coalescence. *Journal of Geophysical Research: Solid Earth* 119, 1 (2014), 216–239.
- [54] iRMB at Technische Universität Braunschweig. 2023. VirtualFluids. Git repository. <https://git.rz.tu-bs.de/irmb/virtualfluids>
- [55] Jérôme Jacob, Orestis Malaspinas, and Pierre Sagaut. 2018. A new hybrid recursive regularised Bhatnagar-Gross-Krook collision model for Lattice Boltzmann method-based large eddy simulation. *Journal of Turbulence* 19, 11-12 (2018), 1051–1076. doi:10.1080/14685248.2018.1540879
- [56] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Trans. Graph.* 34, 4, Article 51 (July 2015), 10 pages.
- [57] Carolin Körner, Michael Thies, Torsten Hofmann, Nils Thürey, and Ulrich Rüde. 2005. Lattice Boltzmann model for free surface flow for modeling foaming. *Journal of Statistical Physics* 121 (2005), 179–196.
- [58] Anthony J. C. Ladd. 1994. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics* 271 (1994), 285–309. doi:10.1017/S0022112094001771
- [59] Pierre Lallemand and Li-Shi Luo. 2000. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. *Phys. Rev. E* 61 (Jun 2000), 6546–6562. Issue 6. doi:10.1103/PhysRevE.61.6546
- [60] Jonas Latt. 2007. *Hydrodynamic limit of lattice Boltzmann equations*. Ph.D. Dissertation. University of Geneva.
- [61] Jonas Lätt and Bastien Chopard. 2006. Lattice Boltzmann Method with Regularized Pre-collision Distribution Functions. *Mathematics and Computers in Simulation* 72, 2 (2006), 165–168.
- [62] Moritz Lehmann. 2019. *High Performance Free Surface LBM on GPUs*. Ph.D. Dissertation. University of Bayreuth.
- [63] Wei Li, Kai Bai, and Xiaopei Liu. 2019. Continuous-scale kinetic fluid simulation. *IEEE Trans. Vis. Comp. Graph.* 25, 9 (2019), 2694–2709. doi:10.1109/TVCG.2018.2859931
- [64] Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and scalable turbulent flow simulation with two-way coupling. *ACM Trans. Graph.* 39, 4 (2020), 47.
- [65] Wei Li and Mathieu Desbrun. 2023. Fluid-Solid Coupling in Kinetic Two-Phase Flow Simulation. *ACM Trans. Graph.* 42, 4, Article 123 (2023).
- [66] Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. 2021. Kinetic-based multiphase flow simulation. *IEEE Trans. Vis. & Comp. Graph.* 27, 7 (2021), 3318–3334.
- [67] Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient kinetic simulation of two-phase flows. *ACM Transactions on Graphics* 41, 4 (2022), 114.
- [68] Wei Li, Tongtong Wang, Zherong Pan, Xifeng Gao, Kui Wu, and Mathieu Desbrun. 2023. High-order moment-encoded kinetic simulation of turbulent flows. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–13.
- [69] Wei Li, Xiaoming Wei, and Arie Kaufman. 2003. Implementing lattice Boltzmann computation on graphics hardware. *The Visual Computer* 19, 7 (2003), 444–456.
- [70] Wei Li, Kui Wu, and Mathieu Desbrun. 2024. Kinetic Simulation of Turbulent Multifluid Flows. *ACM Trans. Graph.* 43, 4, Article 55 (2024).
- [71] Mengyun Liu, Kai Bai, and Xiaopei Liu. 2025. A Hybrid Near-wall Model for Kinetic Simulation of Turbulent Boundary Layer Flows. *ACM Trans. Graph.* 44, 4, Article 148 (July 2025), 24 pages. doi:10.1145/3730829
- [72] Mengyun Liu and Xiaopei Liu. 2023. A Parametric Kinetic Solver for Simulating Boundary-Dominated Turbulent Flow Phenomena. *ACM Trans. Graph.* 42, 6, Article 189 (Dec. 2023), 20 pages. doi:10.1145/3618313
- [73] Qin Lou, Zhaoli Guo, and Baochang Shi. 2013. Evaluation of outflow boundary conditions for two-phase lattice Boltzmann equation. *Physical review E* 87, 6 (2013), 063301.
- [74] Chaoyang Lyu, Kai Bai, Yiheng Wu, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2023. Building a Virtual Weakly-Compressible Wind Tunnel Testing Facility. *ACM Trans. Graph.* 42, 4 (2023).
- [75] Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Trans. Graph.* 40, 6 (2021), 1–18.
- [76] Ahmed H Mahmoud, Hesam Salehipour, and Massimiliano Meneghin. 2024. Optimized GPU implementation of grid refinement in lattice Boltzmann method. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 398–407.
- [77] Orestis Malaspinas. 2015. Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization. arXiv:1505.06900 [physics.flu-dyn] <https://arxiv.org/abs/1505.06900>
- [78] Yang Ning, Kannan N. Premnath, and Dhiraj V. Patil. 2016. Numerical study of the properties of the central moment lattice Boltzmann method. *International Journal for Numerical Methods in Fluids* 82, 2 (2016), 59–90. doi:10.1002/ld.4208
- [79] Christian Obrecht, Frédéric Kuznik, Bernard Tourancheau, and Jean-Jacques Roux. 2013. Multi-GPU implementation of the lattice Boltzmann method. *Computers & Mathematics with Applications* 65, 2 (2013), 252–261.
- [80] Christian Obrecht, Frédéric Kuznik, Bernard Tourancheau, and Jean-Jacques Roux. 2013. Scalable lattice Boltzmann solvers for CUDA GPU clusters. *Parallel Comput.* 39, 6-7 (2013), 259–270.
- [81] Cheng Peng, Yihua Teng, Brian Hwang, Zhaoli Guo, and Lian-Ping Wang. 2016. Implementation issues and benchmarking of lattice Boltzmann method for moving rigid particle simulations in a viscous flow. *Math. Comput. Simul.* 72, 2 (2016), 349–374.
- [82] Charles S. Peskin. 1972. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.* 10, 2 (1972), 252–271. doi:10.1016/0021-9991(72)90065-4

- [83] Martin Rohde, Drona Kandhai, Jos J. Derksen, and Harry E. A. van den Akker. 2006. A Generic, Mass Conservative Local Grid Refinement Technique for Lattice-Boltzmann Schemes. *International Journal for Numerical Methods in Fluids* 51, 4 (2006), 439–468.
- [84] Florian Schornbaum and Ulrich Rude. 2016. Massively parallel algorithms for the lattice Boltzmann method on nonuniform grids. *SIAM Journal on Scientific Computing* 38, 2 (2016), C96–C126.
- [85] Xiaowen Shan. 2016. The mathematical structure of the lattices of the lattice Boltzmann method. *Journal of Computational Science* 17 (2016), 475–481.
- [86] Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. 2006. Kinetic theory representation of hydrodynamics: a way beyond the Navier–Stokes equation. *Journal of Fluid Mechanics* 550 (2006), 413–441.
- [87] Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128.
- [88] Shi Tao, Qing He, Baiman Chen, Xiaoping Yang, and Simin Huang. 2018. One-point Second-order Curved Boundary Condition for Lattice Boltzmann Simulation of Suspended Particles. *Computers & Mathematics with Applications* 76, 7 (2018), 1593–1607.
- [89] Jonas Tölke and Manfred Krafczyk. 2008. TeraFLOP computing on a desktop PC with GPUs for 3D CFD. *International Journal of Computational Fluid Dynamics* 22, 7 (2008), 443–456.
- [90] Nhat-Phuong Tran, Myungho Lee, and Sugwon Hong. 2017. Performance optimization of 3D lattice Boltzmann flow solver on a GPU. *Scientific Programming* 2017, 1 (2017), 1205892.
- [91] Madhurima Vardhan, John Gounley, Luiz Hegele, Erik W. Draeger, and Amanda Randles. 2019. Moment Representation in the Lattice Boltzmann Method on Massively Parallel Hardware. In *Supercomputing*. Article 34.
- [92] Haoxiang Wang, Kui Wu, Hui Qiao, Mathieu Desbrun, and Wei Li. 2025. Kinetic Free-Surface Flows and Foams with Sharp Interfaces. *ACM Transactions on Graphics* (2025).
- [93] Jiwei Wang, Wenbin Song, Yicheng Fan, Yang Wang, and Xiaopei Liu. 2025. A Highly-Efficient Hybrid Simulation System for Flight Controller Design and Evaluation of Unmanned Aerial Vehicles. *ACM Trans. Graph.* 44, 6, Article 198 (Dec. 2025), 23 pages.
- [94] Gerhard Wellein, Thomas Zeiser, Georg Hager, and Stefan Donath. 2006. On the single processor performance of simple lattice Boltzmann kernels. *Computers & Fluids* 35, 8-9 (2006), 910–919.
- [95] Nicholas Wilt. 2013. *The cuda handbook: A comprehensive guide to gpu programming*. Pearson Education.
- [96] Wolfram Research Inc. 2023. Mathematica, Version 13.2. <https://www.wolfram.com/mathematica>
- [97] Shuonan Wu and Jinchao Xu. 2017. Multiphase Allen–Cahn and Cahn–Hilliard models and their discretizations with the effect of pairwise surface tensions. *J. Comput. Phys.* 343 (2017), 10–32.
- [98] Xiaoyu Xiao, Ding Lin, Yiheng Wu, Kai Bai, and Xiaopei Liu. 2025. Simulating Two-phase Fluid-rigid Interactions with an Overset-Grid Kinetic Solver. *IEEE Transactions on Visualization and Computer Graphics* (2025).
- [99] David L Youngs. 1984. An interface tracking method for a 3D Eulerian hydrodynamics code. *Atomic Weapons Research Establishment* 44-92 (1984).
- [100] Raoyang Zhang, Xiaowen Shan, and Hudong Chen. 2006. Efficient Kinetic Method for Fluid Simulation beyond the Navier-Stokes Equation. *Phys. Rev. E* 74 (2006), 046703. Issue 4.
- [101] Xinxin Zhang, Minchen Li, and Robert Bridson. 2016. Resolving fluid boundary layers with particle strength exchange and weak adaptivity. *ACM Trans. Graph.* 35, 4, Article 76 (July 2016), 8 pages.
- [102] Weifeng Zhao and Wen-An Yong. 2017. Single-node second-order boundary schemes for the lattice Boltzmann method. *J. Comput. Phys.* 329 (2017), 1–15.

APPENDICES

A Equilibrium Formulations

From Detailed Balance to Collision Invariants (Eq. (11) $\rightarrow$ Eq. (12)). When the system reaches statistical equilibrium, the collision operator is zero, i.e., $\Omega(f, f) = 0$. According to the Principle of Detailed Balance, this requires that in any elastic collision (with pre-collision velocities c, c_1 and post-collision velocities c', c'_1), the probabilities of forward and reverse collisions are equal, yielding the identity:

$$f(c)f(c_1) = f(c')f(c'_1)$$

Taking the natural logarithm of both sides converts the multiplicative relationship into an additive one:

$$\ln f(c) + \ln f(c_1) = \ln f(c') + \ln f(c'_1).$$

This equation holds profound physical significance: it indicates that $\ln f$ is a quantity whose sum remains constant before and after a collision, known as a Collision Invariant. In a monatomic gas with no internal degrees of freedom, an elastic collision process has only three fundamental conserved quantities: Mass (proportional to the constant 1), Momentum (proportional to the velocity vector $\mathbf{c}$) and Kinetic Energy (proportional to the square of velocity $\mathbf{c}^2$). Since $\ln f$ is a collision invariant, it must be a linear combination of these three fundamental conserved quantities. Therefore, we can express it as:

$$\ln f = \chi + \boldsymbol{\zeta} \cdot \mathbf{c} + \lambda c^2.$$

Here, χ is a scalar constant, $\boldsymbol{\zeta}$ is a vector constant, and λ is a scalar constant. Taking the exponential of both sides yields the exponential family distribution form belonging to Eq. (12):

$$f = \exp(\chi + \boldsymbol{\zeta} \cdot \mathbf{c} + \lambda c^2).$$

Completing the Square and Introducing Macroscopic Physical Quantities. To clarify the relationship between the constants $\chi, \boldsymbol{\zeta}, \lambda$ in Eq. (12) and macroscopic fluid dynamic variables, we first complete the square for the exponential part, rewriting it into a quadratic form based on a certain center velocity:

$$f^{eq}(\mathbf{c}) = A \exp(-\beta(\mathbf{c} - \mathbf{u})^2).$$

Here, $A, \beta, \mathbf{u}$ are the undetermined coefficients after re-parameterization. According to the definition in the text, this distribution function f^{eq} is directly linked to three locally conserved macroscopic variables: mass density ρ , momentum density $\rho \mathbf{u}$, and total energy (or thermodynamic temperature T). We will determine these constants one by one by calculating the zeroth, first, and second moments of the distribution function. The derivation will repeatedly use standard Gaussian integral formulas in D -dimensional space:

$$\int_{-\infty}^{\infty} \exp(-ax^2) dx = \sqrt{\frac{\pi}{a}} \quad \text{and} \quad \int_{-\infty}^{\infty} x^2 \exp(-ax^2) dx = \frac{1}{2a} \sqrt{\frac{\pi}{a}}$$

Step 1: Determining Coefficient A (involving β) via the Zeroth Moment. The macroscopic mass density ρ is the integral of the distribution function over velocity space (Note: the definition of f^{eq} makes its integral the mass density rather than the number density):

$$\rho = \int f^{eq}(\mathbf{c}) d\mathbf{c} = \int A \exp(-\beta(\mathbf{c} - \mathbf{u})^2) d\mathbf{c}$$

Let the relative velocity be $\mathbf{v} = \mathbf{c} - \mathbf{u}$, changing the integration domain to $\mathbf{v} \in \mathbb{R}^D$:

$$\rho = A \int \exp(-\beta v^2) d\mathbf{v} = A \left(\int_{-\infty}^{\infty} \exp(-\beta v_x^2) dv_x \right)^D = A \left(\frac{\pi}{\beta} \right)^{D/2}$$

Solving for coefficient A gives:

$$A = \rho \left(\frac{\beta}{\pi} \right)^{D/2}$$

Step 2: Verifying Macroscopic Velocity $\mathbf{u}$ via the First Moment. The momentum density is defined by the first moment:

$$\rho \mathbf{u}_{\text{macro}} = \int \mathbf{c} f^{eq}(\mathbf{c}) d\mathbf{c} = \int (\mathbf{v} + \mathbf{u}) f^{eq}(\mathbf{v} + \mathbf{u}) d\mathbf{v} = \int \mathbf{v} A \exp(-\beta v^2) d\mathbf{v} + \mathbf{u} \int A \exp(-\beta v^2) d\mathbf{v}$$

The first term is the integral of an odd integrand over a symmetric interval, which results in 0. The integral of the second term is exactly ρ from earlier. Therefore, $\rho \mathbf{u}_{\text{macro}} = 0 + \rho \mathbf{u}$. This proves that the parameter $\mathbf{u}$ in the completed square is indeed the local macroscopic flow velocity.

Step 3: Determining Parameter β (Temperature T) via the Second Central Moment. According to the equipartition theorem in statistical mechanics, the average translational kinetic energy per degree of freedom is $\frac{1}{2}k_B T$. In D -dimensional space, the local thermodynamic internal energy density of a monatomic gas (excluding macroscopic translational kinetic energy) can be expressed as $\frac{D}{2}nk_B T$, where the number density is $n = \rho/m$. Using the second central moment to calculate the internal energy density:

$$\frac{D}{2} \frac{\rho}{m} k_B T = \int \frac{1}{2} (\mathbf{c} - \mathbf{u})^2 f^{eq}(\mathbf{c}) d\mathbf{c} = \frac{1}{2} A \int \mathbf{v}^2 \exp(-\beta \mathbf{v}^2) d\mathbf{v}$$

Utilizing Gaussian integrals, since $\mathbf{v}^2 = v_x^2 + v_y^2 + \dots + v_D^2$, the integral can be decomposed into the sum of D integrals of the same dimension:

$$\int \mathbf{v}^2 \exp(-\beta \mathbf{v}^2) d\mathbf{v} = D \left(\int_{-\infty}^{\infty} v_x^2 \exp(-\beta v_x^2) dv_x \right) \left(\int_{-\infty}^{\infty} \exp(-\beta v_y^2) dv_y \right)^{D-1}$$

Substituting the standard integral results:

$$= D \left(\frac{1}{2\beta} \sqrt{\frac{\pi}{\beta}} \right) \left(\sqrt{\frac{\pi}{\beta}} \right)^{D-1} = \frac{D}{2\beta} \left(\frac{\pi}{\beta} \right)^{D/2}$$

Substituting this result back into the internal energy equation, and utilizing the previously derived $\rho = A(\pi/\beta)^{D/2}$:

$$\frac{D}{2} \frac{\rho}{m} k_B T = \frac{1}{2} A \left[\frac{D}{2\beta} \left(\frac{\pi}{\beta} \right)^{D/2} \right] = \frac{D}{4\beta} \rho$$

Simplifying both sides and canceling D and ρ :

$$\frac{k_B T}{m} = \frac{1}{2\beta} \implies \beta = \frac{m}{2k_B T}$$

Assembling the Final Form. Substituting the derived $\beta = \frac{m}{2k_B T}$ back into the coefficient A obtained in the first step:

$$A = \rho \left(\frac{m/2k_B T}{\pi} \right)^{D/2} = \rho \left(\frac{m}{2\pi k_B T} \right)^{D/2}$$

Finally, substituting A and β , which now have explicit physical meanings, back into the completed square exponential form $f^{eq}(\mathbf{c}) = A \exp(-\beta(\mathbf{c} - \mathbf{u})^2)$, we obtain the complete local Maxwellian equilibrium distribution:

$$f^{eq}(\mathbf{c}, \rho, \mathbf{u}, T) = \rho \left(\frac{m}{2\pi k_B T} \right)^{D/2} \exp \left(-\frac{m(\mathbf{c} - \mathbf{u})^2}{2k_B T} \right)$$

B Standard Maxwell-Boltzmann equilibrium weighting function

The derivation using the microscopic particle properties (mass m and Boltzmann constant k_B) is mathematically isomorphic to the macroscopic formulation using the specific gas constant R . This equivalence is bridged by the fundamental thermodynamic identity:

$$R = \frac{k_B}{m}$$

By substituting this relation, the non-dimensionalization process unfolds with perfect symmetry. In the microscopic framework, the isothermal speed of sound is $c_s = \sqrt{k_B T/m}$. The nondimensional relative velocity $\mathbf{v}$ is defined by scaling

the physical relative velocity by this thermal fluctuation speed:

$$v = \frac{\mathbf{c} - \mathbf{u}}{\sqrt{\frac{k_B T}{m}}} = \sqrt{\frac{m}{k_B T}} (\mathbf{c} - \mathbf{u})$$

Conversely, the physical velocity $\mathbf{c}$ is expressed as: $\mathbf{c} = \mathbf{u} + \sqrt{\frac{k_B T}{m}} \mathbf{v}$. Substituting the expression for $\mathbf{c}$ into the exponential argument of the Maxwellian distribution $f^{eq}(\mathbf{c})$:

$$-\frac{m(\mathbf{c} - \mathbf{u})^2}{2k_B T} = -\frac{m \left(\sqrt{\frac{k_B T}{m}} \mathbf{v} \right)^2}{2k_B T} = -\frac{m \left(\frac{k_B T}{m} \right) \mathbf{v}^2}{2k_B T}.$$

The physical constants m and $k_B T$ perfectly cancel, leaving a pure quadratic form: $-\frac{\mathbf{v}^2}{2}$.

Under the change of variables, the volume element of the velocity space scales according to the Jacobian determinant. In a D -dimensional space, each velocity component transforms as $d\mathbf{c}_\alpha = \sqrt{k_B T / m} d\mathbf{v}_\alpha$. Thus, the D -dimensional measure transforms as:

$$d\mathbf{c} = \left(\sqrt{\frac{k_B T}{m}} \right)^D d\mathbf{v} = \left(\frac{k_B T}{m} \right)^{D/2} d\mathbf{v}$$

Substituting both the simplified exponential and the transformed integration measure $d\mathbf{c}$ back into the density integral $\int f^{eq}(\mathbf{c}) d\mathbf{c}$:

$$\int f^{eq}(\mathbf{c}) d\mathbf{c} = \int \underbrace{\rho \left(\frac{m}{2\pi k_B T} \right)^{D/2}}_{\text{Transformed } f^{eq}(\mathbf{c})} \underbrace{\exp\left(-\frac{\mathbf{v}^2}{2}\right) \left(\frac{k_B T}{m} \right)^{D/2} d\mathbf{v}}_{\text{New measure } d\mathbf{c}}$$

Rearranging the coefficients reveals the underlying mathematical structure:

$$\int f^{eq}(\mathbf{c}) d\mathbf{c} = \int \rho \left[\frac{1}{(2\pi)^{D/2}} \left(\frac{m}{k_B T} \right)^{D/2} \left(\frac{k_B T}{m} \right)^{D/2} \right] \exp\left(-\frac{\mathbf{v}^2}{2}\right) d\mathbf{v}$$

The normalization coefficient intrinsic to the distribution $\left(\frac{m}{k_B T} \right)^{D/2}$ and the spatial scaling factor generated by the Jacobian $\left(\frac{k_B T}{m} \right)^{D/2}$ are exact reciprocals. They annihilate each other to equal 1:

$$\int f^{eq}(\mathbf{c}) d\mathbf{c} = \int \rho \left[\frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{\mathbf{v}^2}{2}\right) \right] d\mathbf{v}$$

The bracketed term is strictly the standard normal distribution base $\omega(\mathbf{v})$.