Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling: *Supplementary Material*

WEI LI, ShanghaiTech University/SIMIT/UCAS
YIXIN CHEN, ShanghaiTech University/DGene
MATHIEU DESBRUN, ShanghaiTech/Caltech
CHANGXI ZHENG, Columbia University
XIAOPEI LIU, ShanghaiTech University

This supplementary document is meant to complement the main paper published at SIGGRAPH 2020 (ACM Trans. Graph, 39(4), 2020). We herein provide readers with some background on lattice Boltzmann methods to better understand our proposed technique. In particular, we present an introduction to lattice Boltzmann methods, starting from a derivation of the classical Boltzmann equations to the state-of-the-art lattice Boltzmann techniques. We also describe the connections between Boltzmann equations and Navier-Stokes equations, and the advantages of using a Boltzmann solver over a Navier-Stokes solver. Further discussed is the solid boundary treatment using the immersed boundary method that supports both static and dynamic solids in the fluid domain. In addition, to enable stable and adaptive simulations, we present more details on the dimension rescaling used in our fluid simulation and in our fluid-solid coupling, with an analysis of the stability range of the parameters.

1 NAVIER-STOKES EQUATIONS FOR FLUID FLOWS

Compressible Navier-Stokes equations. Fluid dynamics (under isothermal conditions) are governed by the classical compressible Navier-Stokes equations, namely,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \,, \tag{1}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{F} \,, \tag{2}$$

where $\otimes$ indicates the outer product; t is the time; $\boldsymbol{F}$ is the external force density (such as gravity); ρ, $\boldsymbol{u}$, and p are density, velocity, and pressure fields of the fluid domain, respectively. Pressure under a temperature T is determined by the equation of state (EoS): $p = p(\rho, T)$ — for example, $p = \rho RT$ for ideal gas, where R is the ideal gas constant. In Eq. (2), the shear stress tensor $\boldsymbol{\sigma}$ is defined as:

$$\boldsymbol{\sigma} = 2\rho\nu \boldsymbol{S} - \rho\nu' (\nabla \cdot \boldsymbol{u}) \boldsymbol{I} \,, \tag{3}$$

where $\boldsymbol{I}$ is the second-order identity tensor (i.e., an identity matrix); ν and ν' are the shear and bulk viscosities, respectively (for an ideal gas, $\nu' = 2\nu/3$); and $\boldsymbol{S}$ is the strain rate defined as

$$\boldsymbol{S} = \frac{1}{2} \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) . \tag{4}$$

Compressible Navier-Stokes equations are difficult to solve. The compressibility allows the sound wave to propagate in a finite speed in the fluid, and thus requires the use of extremely small timestep size to resolve sound propagation in the full frequency spectrum. To ensure numerical stability at larger timestep sizes, implicit time integration is often used, but it requires solving a large global linear system at each timestep, a rather slow solving process.

Incompressible Navier-Stokes equations. A relaxation of compressible Navier-Stokes equations is to disable sound wave propagation by assuming a constant density ρ everywhere. Under this assumption, the fluid becomes incompressible. This effectively leads to an infinite sound speed, and results in the following incompressible Navier-Stokes equations:

$$\nabla \cdot \boldsymbol{u} = 0 \,, \tag{5}$$

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

Incompressible Navier-Stokes equations have been widely used for generating fluid animations in computer graphics, and numerous research efforts have been invested towards developing efficient numerical solvers (see discussion in §1.1 of the main text). In these solvers, a key step at each timestep is to compute the pressure p by solving the Poisson equation:

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

Numerical discretization of this equation leads to a large linear system that involves all (discretized) pressure values in the fluid. Solving this linear system is expensive and hard to be fully parallelized. As a result, it is often the performance bottleneck in a fluid simulation. In addition, because the incompressibility condition (5) is merely a mathematical simplification (which is not physical), finding a rigorous physical justification for pressure boundary conditions of (7) is difficult: while the no-penetration or no-slip pressure boundary condition has been commonly used in computer graphics applications, it remains controversial in the fluid dynamics community what precise pressure boundary conditions should be used to achieve high simulation accuracy [Gresho and Sani 1987].

2 BOLTZMANN EQUATION

An alternative route for modeling fluid dynamics is to consider the evolution of a large number of microscopic fluid particles from a statistical perspective. Central in this approach is the concept of distribution function, $f(\boldsymbol{x}, \boldsymbol{v}, t)$, that indicates the probability for a microscopic fluid particle to be at position $\boldsymbol{x}$ at time t with a velocity $\boldsymbol{v}$. The microscopic fluid particles move around and collide with each other. As a result, the evolution of $f(\boldsymbol{v}, \boldsymbol{x}, t)$ is governed by the

Authors' addresses: Wei Li, Yixin Chen, Xiaopei Liu, School of Information Science and Technology (Shanghai Engineering Research Center of Intelligent Vision and Imaging) of ShanghaiTech University, Shanghai, China; Wei Li is also affiliated with the Shanghai Institute of Microsystem and Information Technology (SIMIT) and the University of the Chinese Academy of Sciences (UCAS); Mathieu Desbrun, California Institute of Technology, Pasadena, CA, USA, on sabbatical at SIST in ShanghaiTech University, Shanghai, China; Changxi Zheng, Columbia University, New York, NY, USA.

continuous Boltzmann transport equation,

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla f = \Omega(f) + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f \,, \tag{8}$$

where $\boldsymbol{F}$ is the external force density (as in (2)), and $\Omega(f)$ is the so-called collision (or relaxation) operator, modeling the change of distribution function $f(\boldsymbol{x}, \boldsymbol{v}, t)$ due to particle collisions. Effectively, this operator relaxes f towards a local equilibrium state $\bar{f}$.

The distribution function f, while accounting for microscopic particles, naturally connects to macroscopic physical quantities. For example, taking the zero-th, first, and second-order moments of f with respect to $\boldsymbol{v}$ yields the macroscopic density ρ, momentum $\rho\boldsymbol{u}$, and momentum flux tensor $\boldsymbol{\Pi}$:

$$\rho = \int f \mathrm{d}\boldsymbol{v}, \quad \rho\boldsymbol{u} = \int \boldsymbol{v} f \mathrm{d}\boldsymbol{v}, \quad \text{and } \boldsymbol{\Pi} = \int \boldsymbol{v}\boldsymbol{v}^T f \mathrm{d}\boldsymbol{v} \,. \tag{9}$$

The latter further relates to the pressure p and shear stress tensor through the relation $\boldsymbol{\Pi} = \rho\boldsymbol{u}\otimes\boldsymbol{u} + p\boldsymbol{I} - \boldsymbol{\sigma}$, where the pressure p can be obtained from f as well, namely,

$$p = \frac{1}{3} \int \|\boldsymbol{v} - \boldsymbol{u}\|_2^2 f \mathrm{d}\boldsymbol{v} \,. \tag{10}$$

To complete the Boltzmann equation (8), we need a model for the collision operator Ω. First, Ω must satisfy certain constraints, which can be seen by taking the zero-th and first order moment integrals (defined in Eq. (9)) on both sides of Eq. (8): by taking the integrals, we expect to recover the macroscopic compressible Navier-Stokes equations in Eqs. (1) and (2). This expectation is met only when Ω satisfies the constraints,

$$\int \Omega \mathrm{d}\boldsymbol{v} = 0 \quad \text{and} \quad \int \boldsymbol{v}\Omega \mathrm{d}\boldsymbol{v} = 0 \,. \tag{11}$$

The most widely used collision model is the iso-thermal single-relaxation-time Bhatnagar-Gross-Krook (BGK) model [Chen and Doolen 1998]:

$$\Omega(f) = -\frac{f - \bar{f}(\rho, \boldsymbol{u})}{\tau} \,, \tag{12}$$

where the density ρ and macroscopic velocity $\boldsymbol{u}$ are obtained through the zero-th and first order moments of f, as defined in Eq. (9); τ is the relaxation time related to the fluid's kinematic viscosity ν; and $\bar{f}$ is the local equilibrium distribution defined by the Maxwell-Boltzmann distribution under a constant temperature, namely,

$$\bar{f}(\rho, \boldsymbol{u}) = \frac{\rho}{(2\pi)^{D/2}} \exp\left(-\frac{\|\boldsymbol{v} - \boldsymbol{u}\|_2^2}{2}\right) \,. \tag{13}$$

Here D is the number of dimensions (i.e., $D=2$ and 3 for 2D and 3D cases, respectively). It is simple to check that the BGK model satisfies the constraints in Eq. (11) and thus recovers the compressible Navier-Stokes equations (1) and (2).

3 LATTICE BOLTZMANN EQUATIONS

Unlike the Navier-Stokes equations that involve nonlinear differential operators (such as the advection term $\nabla \cdot (\rho\boldsymbol{u}\otimes\boldsymbol{u})$), Boltzmann equation (8) has only linear differential operators. This simplicity enables the numerical solver to easily conserve momentum and energy — a feature desired for simulating turbulent flows. Yet, a challenge arising from (8) is how to effectively discretize f due to its high dimension: f has 5 dimensions in 2D and 7 dimensions in 3D, as it depends on both position and velocity.

Lattice Boltzmann methods are developed to address this challenge and solve the Boltzmann equation (8). The starting point is to apply a separation of variables on f using Hermite series expansions [Shan et al. 2006] in velocity space,

$$f(\boldsymbol{v}, \boldsymbol{x}, t) = \omega(\boldsymbol{v}) \sum_{n=0}^{\infty} \frac{1}{n!} \mathbf{a}^{(n)}(\boldsymbol{x}, t) : \mathbf{H}^{(n)}(\boldsymbol{v}) \,. \tag{14}$$

Here the superscript "(n)" indicates a rank-n tensor; the operator ":" denotes full tensor contraction, i.e.,

$$\mathbf{A}^{(n)} : \mathbf{B}^{(n)} = \sum_{i_1,\ldots,i_n} \mathbf{A}_{i_1,i_2,\ldots,i_n} \mathbf{B}_{i_1,i_2,\ldots,i_n};$$

and the weighting function $\omega(\boldsymbol{v})$ is defined as:

$$\omega(\boldsymbol{v}) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{\|\boldsymbol{v}\|_2^2}{2}\right). \tag{15}$$

Lastly, the Hermite polynomial basis of order n is an n-th order tensor defined as

$$\mathbf{H}^{(n)}(\boldsymbol{v}) = \frac{(-1)^n}{\omega(\boldsymbol{v})} \nabla^n \omega(\boldsymbol{v}). \tag{16}$$

Hermite polynomials form an orthonormal basis with respect to the weighting function in Eq. (15), thus the corresponding coefficient $\mathbf{a}^{(n)}(\boldsymbol{x}, t)$ can be computed via a weighted inner product,

$$\mathbf{a}^{(n)}(\boldsymbol{x}, t) = \int \frac{f(\boldsymbol{v}, \boldsymbol{x}, t)}{\omega(\boldsymbol{v})} \mathbf{H}^{(n)}(\boldsymbol{v}) \mathrm{d}\boldsymbol{v} \,. \tag{17}$$

Hermite polynomial expansions are particularly appropriate to our goals as they allow us to easily recover macroscopic quantities: the first three orders of coefficients have straightforward physical interpretations since

$$\mathbf{a}^{(0)} = \rho \,, \quad \mathbf{a}^{(1)} = \rho\boldsymbol{u} \,, \quad \text{and } \mathbf{a}^{(2)} = \boldsymbol{\Pi} - \rho\boldsymbol{I} \,. \tag{18}$$

To evaluate the coefficients numerically, the integral (17) can be approximated through Gauss-Hermite quadrature,

$$\mathbf{a}^{(n)}(\boldsymbol{x}, t) = \sum_{i=0}^{q-1} \frac{w_i}{\omega(\boldsymbol{c}_i)} f(\boldsymbol{c}_i, \boldsymbol{x}, t) \mathbf{H}^{(n)}(\boldsymbol{c}_i) \,, \tag{19}$$

where q is the number of discrete microscopic velocities used in the approximation; w_i are the quadrature weights (also called lattice weights). Different ways of choosing the discretized velocities $\{\boldsymbol{c}_i\}_{i=0}^{q-1}$ exist. The commonly used schemes in 2D and 3D are D2Q9 and D3Q27 structures as illustrated in Fig. 2 of the main text. For example, in 2D, the discretized velocities are:

$$\boldsymbol{c}_i = \begin{cases} (0,0), & i=0 \,, \\ (\pm1, 0)c, (0, \pm1)c, & i=1,2,3,4 \,, \\ (\pm1, \pm1)c, & i=5,6,7,8 \,, \end{cases} \tag{20}$$

and in 3D, they are:

$$\boldsymbol{c}_i = \begin{cases} (0,0,0), & i=0 \,, \\ (\pm1,0,0)c, (0,\pm1,0)c, (0,0,\pm1)c, & i=1,\ldots,6 \,, \\ (\pm1,\pm1,0)c, (\pm1,0,\pm1)c, (0,\pm1,\pm1)c, & i=7,\ldots,18 \,, \\ (\pm1,\pm1,\pm1)c, & i=19,\ldots,26 \,, \end{cases} \tag{21}$$

where c is the lattice speed constant; their corresponding lattice weights w_i in 2D are:

$$w_i = \begin{cases} 4/9, & i = 0\ , \\ 1/9, & i = 1, 2, 3, 4\ , \\ 1/36, & i = 5, 6, 7, 8\ , \end{cases} \tag{22}$$

and in 3D are:

$$w_i = \begin{cases} 8/27, & i = 0\ , \\ 2/27, & i = 1, 2, ..., 6\ , \\ 1/54, & i = 7, 8, ..., 18\ , \\ 1/216, & i = 19, 20, ..., 26\ . \end{cases} \tag{23}$$

With this choice of discretized velocities, we define discrete distribution functions as

$$f_i(\boldsymbol{x}, t) = \frac{w_i}{\omega(\boldsymbol{c}_i)} f(\boldsymbol{c}_i, \boldsymbol{x}, t), \tag{24}$$

which allows us to compute the macroscopic physical quantities through Gauss-Hermite quadrature as:

$$\rho = \sum_{i=0}^{q-1} f_i, \quad \rho \boldsymbol{u} = \sum_{i=0}^{q-1} \boldsymbol{c}_i f_i, \quad \boldsymbol{\Pi} = \sum_{i=0}^{q-1} \boldsymbol{c}_i^2 f_i, \quad p = \frac{1}{3} \sum_{i=0}^{q-1} \|\boldsymbol{c}_i - \boldsymbol{u}\|_2^2 f_i\ . \tag{25}$$

We can now use the Hermite expansion from Eq. (14) together with the Gauss-Hermite quadrature scheme (19) to discretize the velocity space of the Boltzmann equation (8). For the BGK model as the collision operator, we obtain the following equation for each f_i:

$$\frac{\partial f_i}{\partial t} + \boldsymbol{c}_i \cdot \nabla f_i = -\frac{f_i - \bar{f}_i}{\tau} + \boldsymbol{F} \cdot \nabla_{\boldsymbol{c}_i} f_i\ , \tag{26}$$

where $\bar{f}_i$ is obtained by re-expressing the local equilibrium distribution (13) using Hermite expansion. In most existing Lattice Boltzmann methods, only the expansion up to the second order is used, resulting in the following discrete BGK equilibrium distribution:

$$\bar{f}_i(\rho, \boldsymbol{u}) \approx w_i \rho \left(1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2}\right), \tag{27}$$

where c_s is the speed of sound.

Equation (26) shows that the evolution of all f_i are mostly independent, only coupled when calculating $\bar{f}_i$, which depends on ρ and $\boldsymbol{u}$. To numerically solve (26), we further discretize it using standard finite difference technique with second order accuracy in both space and time [Latt 2007]. By choosing a proper characteristic length and velocity (we will discuss this point at length in Sec. 7), one can rescale time and space, yielding the normalized *lattice Boltzmann equation* (LBE),

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i, t + 1) - f_i(\boldsymbol{x}, t) = \Omega_i + F_i\ , \tag{28}$$

which can be solved by a standard (backward) streaming process,

$$f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x} - \boldsymbol{c}_i, t)\ , \tag{29}$$

followed by a collision process,

$$f_i(\boldsymbol{x}, t + 1) = f_i^*(\boldsymbol{x}, t) + \Omega_i + F_i\ . \tag{30}$$

Both processes are conducted at each lattice node. Because of the choice of $\boldsymbol{c}_i$ (recall Fig. 2 in the main paper), they need to use data only on the current lattice node and its immediate neighboring nodes, not globally coupled together (in contrast to the Poisson equation solve in Navier-Stokes solvers). Therefore, these two steps are embarrassingly parallelizable over all lattice nodes, offering high performance integration.

4 RELAXATION MODELS

The collision term Ω_i in the lattice Boltzmann equation (28) was computed from the BGK model as

$$\Omega_i = \frac{f_i - \bar{f}_i}{\tau} = (f_i - \bar{f}_i)/\tau\ , \tag{31}$$

which used a fixed relaxation time τ. The relaxation time determines how quickly the microscopic fluid particles restore to their equilibrium states. It is therefore related to the fluid's kinetic viscosity through the relation,

$$\frac{1}{\tau} = 3\nu + \frac{1}{2}. \tag{32}$$

It has been shown that such a fixed relaxation time leads to a numerically unstable simulation, especially for turbulent flows [Nathen et al. 2017] (i.e., for high Reynolds numbers). This is because a fixed τ relaxes f, and thus all the moments, at the *same* rate, contradicting the fact that physical quantities such as momentum and energy — which directly relates to the moments — approach their equilibrium states at different rates. Therefore, using a single relaxation rate causes severe dispersion errors. Since dispersion errors create sharp velocity gradients, strong numerical instability ensues.

To reduce dispersion errors, we can express the collision operator in a more general form,

$$\boldsymbol{\Omega} := \mathbf{C}(\boldsymbol{f} - \bar{\boldsymbol{f}})\ , \tag{33}$$

where $\boldsymbol{f}$ and $\bar{\boldsymbol{f}}$ are the vectors of all discretized distribution functions f_i and $\bar{f}_i$ at each lattice node. For the D2Q9 and D3Q27 lattice structures (recall Fig. 2 in the main paper), they have length 3^D, with D being the number of dimensions ($D = 2$ or 3). $\mathbf{C}$ is a general relaxation matrix; for the single-relaxation-time BGK model, $\mathbf{C}$ reduces to a simple diagonal matrix, i.e., $\mathbf{C} = \frac{1}{\tau}\mathbf{I}$.

We can define a more complex relaxation model by picking a different expression of this general matrix. An effective way of reducing the dispersion error is the multiple-relaxation-time (MRT) model [d'Humières 2002], motivated by the observation that different moments relax at different rates. In MRT model, $\mathbf{C}$ is constructed through a factorization, $\mathbf{C} = -\mathbf{M}^{-1}\mathbf{R}\mathbf{M}$, where $\mathbf{M}$ is a projection matrix that expresses $\boldsymbol{f}$ in a different basis, and $\mathbf{R}$ is a diagonal matrix whose diagonal elements r_i indicate relaxation rates for individual modes in this modified basis, as we explain next.

4.1 Moment Projection Matrix

There exist multiple ways of constructing $\mathbf{M}$. One of the earliest methods proposes to construct $\mathbf{M}$ that projects $\boldsymbol{f}$ in a "Raw Moment" space, and it is thus often referred as the RM-MRT model [Premnath and Banerjee 2009]. Each element of $\mathbf{M}$ is a combination of monomials of $\boldsymbol{c}_{j,\alpha}$, the α-th component of the j-th lattice velocity $\boldsymbol{c}_j$. In particular, if we express $\mathbf{M}$ in terms of its row vectors $\boldsymbol{m}_i$ (i.e., $\mathbf{M} = [\boldsymbol{m}_0, \boldsymbol{m}_1, ..., \boldsymbol{m}_{q-1}]^T$ with $q = 3^D$), as shown in [De Rosis 2017c,b], the different raw moments $\boldsymbol{m}_i$ in 2D are defined as:

$$\begin{aligned} &\boldsymbol{m}_0 = \{\|\boldsymbol{c}_j\|^0\}, \\ &\boldsymbol{m}_1 = \{\boldsymbol{c}_{j,x}\}, \boldsymbol{m}_2 = \{\boldsymbol{c}_{j,y}\}, \\ &\boldsymbol{m}_3 = \{\boldsymbol{c}_{j,x}^2 + \boldsymbol{c}_{j,y}^2\}, \boldsymbol{m}_4 = \{\boldsymbol{c}_{j,x}^2 - \boldsymbol{c}_{j,y}^2\}, \boldsymbol{m}_5 = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}\}, \\ &\boldsymbol{m}_6 = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}\}, \boldsymbol{m}_7 = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}^2\}, \\ &\boldsymbol{m}_8 = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}^2\}, \end{aligned} \tag{34}$$

where in each $\boldsymbol{m}_i$ the subscript j goes from 0 to $q-1$ to define its individual vector coordinates; and $\boldsymbol{c}_{j,x}$ and $\boldsymbol{c}_{j,y}$ are the x- and y-components of the j-th lattice velocity $\boldsymbol{c}_j$, respectively.

In 3D, the definitions of all $\boldsymbol{m}_i$ are:

$$
\begin{aligned}
&\boldsymbol{m}_0 = \{\|\boldsymbol{c}_j\|^0\},\\
&\boldsymbol{m}_1 = \{\boldsymbol{c}_{j,x}\}, \boldsymbol{m}_2 = \{\boldsymbol{c}_{j,y}\}, \boldsymbol{m}_3 = \{\boldsymbol{c}_{j,z}\},\\
&\boldsymbol{m}_4 = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}\}, \boldsymbol{m}_5 = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,z}\}, \boldsymbol{m}_6 = \{\boldsymbol{c}_{j,y}\boldsymbol{c}_{j,z}\},\\
&\boldsymbol{m}_7 = \{\boldsymbol{c}_{j,x}^2 - \boldsymbol{c}_{j,y}^2\}, \boldsymbol{m}_8 = \{\boldsymbol{c}_{j,x}^2 - \boldsymbol{c}_{j,z}^2\}, \boldsymbol{m}_9 = \{\boldsymbol{c}_{j,x}^2 + \boldsymbol{c}_{j,y}^2 + \boldsymbol{c}_{j,z}^2\},\\
&\boldsymbol{m}_{10} = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}^2 + \boldsymbol{c}_{j,x}\boldsymbol{c}_{j,z}^2\}, \boldsymbol{m}_{11} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y} + \boldsymbol{c}_{j,y}\boldsymbol{c}_{j,z}^2\},\\
&\boldsymbol{m}_{12} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,z} + \boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}\}, \boldsymbol{m}_{13} = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}^2 - \boldsymbol{c}_{j,x}\boldsymbol{c}_{j,z}^2\},\\
&\boldsymbol{m}_{14} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y} - \boldsymbol{c}_{j,y}\boldsymbol{c}_{j,z}^2\}, \boldsymbol{m}_{15} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,z} - \boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}\},\\
&\boldsymbol{m}_{16} = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}\boldsymbol{c}_{j,z}\},\\
&\boldsymbol{m}_{17} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}^2 + \boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,z}^2 + \boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}^2\},\\
&\boldsymbol{m}_{18} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}^2 + \boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,z}^2 - \boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}^2\}, \boldsymbol{m}_{19} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}^2 - \boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,z}^2\},\\
&\boldsymbol{m}_{20} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}\boldsymbol{c}_{j,z}\}, \boldsymbol{m}_{21} = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}\}, \boldsymbol{m}_{22} = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}\boldsymbol{c}_{j,z}^2\},\\
&\boldsymbol{m}_{23} = \{\boldsymbol{c}_{j,x}\boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}^2\}, \boldsymbol{m}_{24} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}\boldsymbol{c}_{j,z}\}^2, \boldsymbol{m}_{25} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}\},\\
&\boldsymbol{m}_{26} = \{\boldsymbol{c}_{j,x}^2\boldsymbol{c}_{j,y}^2\boldsymbol{c}_{j,z}^2\}.
\end{aligned}
\tag{35}
$$

This construction of $\mathbf{M}$ is a *non-orthogonal* version of the RM-MRT model, since $\boldsymbol{m}_i$ are not orthogonal to each other. But the intuition behind $\mathbf{M}$ is simple: it consists in mapping the distribution functions $\boldsymbol{f}$ into its individual moments. For example, according to Eq. (25), $\boldsymbol{m}_0^T\boldsymbol{f}$ is the first-order moment (the density ρ), $\boldsymbol{m}_1^T\boldsymbol{f}$ is the x-component of the macroscopic momentum $\rho\boldsymbol{u}_x$, etc. Note that the relaxation matrix $\mathbf{M}$ can be orthogonalized via Gram-Schmidt orthogonalization, and it has been empirically observed that the orthogonalized RT-MRT model produces more accurate and stable results than its non-orthogonal counterpart.

Limitations. A major limitation of the RM-MRT model is that it violates Galilean invariance. For two inertial frames that differ by a constant velocity $\boldsymbol{u}$, the fluid dynamics should be the same. However, under the two inertial frames, the fluid particle velocities are different, and thus in the RM-MRT model, they are relaxed at different time rates, resulting in different fluid dynamics. Violation of Galilean invariance is the main reason causing simulation instability in the RM-MRT model [De Rosis 2017c].

4.2 Central-Moment Relaxation Model

The central-moment multiple-relaxation-time (CM-MRT) model was introduced as a remedy of the unstable RM-MRT model [Geier et al. 2006]. The idea is simple: to ensure Galilean invariance, it replaces all $\boldsymbol{c}_j$ in the construction of $\mathbf{M}$ and its raw moments in Eqs. (34) and (35) with $\bar{\boldsymbol{c}}_j \coloneqq \boldsymbol{c}_j - \boldsymbol{u}$, where $\boldsymbol{u}$ is the macroscopic velocity at the current lattice node (computed using Eq. (19)). The resulting moments are thus called central moments.

The construction of the relaxation matrix $\mathbf{C} = -\mathbf{M}^{-1}\mathbf{R}\mathbf{M}$ requires the computation of $\mathbf{M}^{-1}$. In RM-MRT model, all $\boldsymbol{c}_j$ only depend on the pre-specified lattice structure, and thus are fixed throughout the simulation. Therefore, $\mathbf{M}^{-1}$ can be pre-computed and stays constant in the simulation. In CM-MRT model, however, $\mathbf{M}$ depends on $\boldsymbol{u}$ which evolves over time in the simulation. Therefore, $\mathbf{M}$ and $\mathbf{M}^{-1}$ need to be updated at each timestep. In practice, note that $\boldsymbol{M}^{-1}$ can be analytically computed, making the update not overly computational expensive.

High-order model. CM-MRT model may still suffer from instability issues, especially in the presence of large velocity variations in the fluid. This is due to the low-order (second-order) Hermite expansion of the Maxwell-Boltzmann equilibrium distribution function $\bar{f}$ (recall Eq. (27)). The resulting $\bar{f}_i$ is used in the CM-MRT model (recall Eq. (33)). To achieve better simulation accuracy, a higher-order expansion is needed. As proposed in [Shan et al. 2006], we employ the highest order Hermite series expansion of $\bar{f}$. 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 $\bar{f}_i$ are

$$
\begin{aligned}
\bar{f}_i \approx & w_i\rho\left[1 + \frac{\boldsymbol{c}_i\cdot\boldsymbol{u}}{c_s^2} + \frac{1}{2c_s^4}\mathbf{H}^{(2)}(\boldsymbol{c}_i) : \boldsymbol{u}\otimes\boldsymbol{u}\right.\\
&+ \frac{1}{2c_s^6}\left(\mathbf{H}^{(3)}_{ixxy}\boldsymbol{u}_x^2\boldsymbol{u}_y + \mathbf{H}^{(3)}_{ixxy}\boldsymbol{u}_x\boldsymbol{u}_y^2\right)\\
&\left.+\frac{1}{4c_s^8}\mathbf{H}^{(4)}_{ixxyy}\boldsymbol{u}_x^2\boldsymbol{u}_y^2\right],
\end{aligned}
\tag{36}
$$

where $\mathbf{H}^{(n)}_{i\alpha_1\ldots\alpha_n}$ is a shorthand notation of the tensor element of $\mathbf{H}^{(n)}(\boldsymbol{c}_i)$ corresponding to the $\alpha_1\ldots\alpha_n$ component. For example, $\mathbf{H}^{(3)}_{ixxy}$ denotes $\mathbf{H}^{(3)}_{xxy}(\boldsymbol{c}_i)$ which is $\frac{-1}{\omega(\boldsymbol{c}_i)}\frac{\partial^3\omega(\boldsymbol{c}_i)}{\partial x\partial x\partial y}$ according to Eq. (16). 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

$$
\begin{aligned}
\bar{f}_i \approx & w_i\rho\left[1 + \frac{\boldsymbol{c}_i\cdot\boldsymbol{u}}{c_s^2} + \frac{1}{2c_s^4}\mathbf{H}^{(2)}(\boldsymbol{c}_i) : \boldsymbol{u}\otimes\boldsymbol{u}\right.\\
&+ \frac{1}{2c_s^6}\left(\mathbf{H}^{(3)}_{ixxy}\boldsymbol{u}_x^2\boldsymbol{u}_y + \mathbf{H}^{(3)}_{ixxz}\boldsymbol{u}_x^2\boldsymbol{u}_z + \mathbf{H}^{(3)}_{ixyy}\boldsymbol{u}_x\boldsymbol{u}_y^2\right.\\
&\left.+\ \mathbf{H}^{(3)}_{ixzz}\boldsymbol{u}_x\boldsymbol{u}_z^2 + \mathbf{H}^{(3)}_{iyzz}\boldsymbol{u}_y\boldsymbol{u}_z^2 + \mathbf{H}^{(3)}_{iyzz}\boldsymbol{u}_y^2\boldsymbol{u}_z + \mathbf{H}^{(3)}_{ixyz}\boldsymbol{u}_x\boldsymbol{u}_y\boldsymbol{u}_z\right)\\
&+ \frac{1}{4c_s^8}\left[\mathbf{H}^{(4)}_{ixxyy}\boldsymbol{u}_x^2\boldsymbol{u}_y^2 + \mathbf{H}^{(4)}_{ixxzz}\boldsymbol{u}_x^2\boldsymbol{u}_z^2 + \mathbf{H}^{(4)}_{iyyzz}\boldsymbol{u}_y^2\boldsymbol{u}_z^2\right.\\
&\left.+\ 2\left(\mathbf{H}^{(4)}_{ixyzz}\boldsymbol{u}_x\boldsymbol{u}_y\boldsymbol{u}_z^2 + \mathbf{H}^{(4)}_{ixyyz}\boldsymbol{u}_x\boldsymbol{u}_y^2\boldsymbol{u}_z + \mathbf{H}^{(4)}_{ixxyz}\boldsymbol{u}_x^2\boldsymbol{u}_y\boldsymbol{u}_z\right)\right]\\
&+ \frac{1}{4c_s^{10}}\left(\mathbf{H}^{(5)}_{ixxyzz}\boldsymbol{u}_x^2\boldsymbol{u}_y\boldsymbol{u}_z^2 + \mathbf{H}^{(5)}_{ixxyyz}\boldsymbol{u}_x^2\boldsymbol{u}_y^2\boldsymbol{u}_z\right.\\
&\left.+\ \mathbf{H}^{(5)}_{ixyyzz}\boldsymbol{u}_x\boldsymbol{u}_y^2\boldsymbol{u}_z^2\right) + \frac{1}{8c_s^{12}}\mathbf{H}^{(6)}_{ixxyyzz}\boldsymbol{u}_x^2\boldsymbol{u}_y^2\boldsymbol{u}_z^2\Bigg].
\end{aligned}
$$

Interestingly, while the full expansions of $\bar{f}_i$ look quite complex, their projections in central-moment space have simple forms: most of the components vanish, regardless of the macroscopic velocity $\boldsymbol{u}$, making the CM-MRT relaxation model Galilean invariant. If we denote $\boldsymbol{q} \coloneqq \mathbf{M}\bar{\boldsymbol{f}}$, then in 2D, the non-zero elements are

$$
\boldsymbol{q}_0 = \rho, \quad \boldsymbol{q}_3 = 2\rho c_s^2, \ \text{ and } \boldsymbol{q}_8 = \rho c_s^4. \tag{37}
$$

and in 3D, they are

$$
\boldsymbol{q}_0 = \boldsymbol{q}_9 = \rho, \ \ \boldsymbol{q}_{17} = \rho c_s^2, \quad \boldsymbol{q}_{18} = \rho c_s^4, \ \text{ and } \boldsymbol{q}_{26} = \rho c_s^6. \tag{38}
$$

4.3 Determining Relaxation Rates

With the projection matrix $\mathbf{M}$ constructed, we now determine the diagonal matrix $\mathbf{R}$ to compute the relaxation matrix $\mathbf{C} = -\mathbf{M}^{-1}\mathbf{R}\mathbf{M}$.

Each diagonal element r_i of $\mathbf{R}$ determines how quickly the specific moment of f relaxes toward its equilibrium state.

Low-order relaxation rates. The low-order moments, as discussed above, have physical interpretations. We can therefore determine their relaxation rates from a physical perspective. The zero-th order moment corresponds to fluid density (recall Eq. (9)), which should always be conserved. Therefore, $r_0 = 0$. For $i = 1, 2, ..., D$, r_i are used to relax first-order moments which correspond to fluid momentum which can be affected by external forces. Therefore, $r_i = 2$ is commonly used. For $i = D+1, ..., i^*$ (where $i^* = 5$ in 2D and $i^* = 9$ in 3D), r_i represent the relaxation rates of second-order moments which are related to the momentum flux. Therefore, they are determined by the kinematic viscosity ν:

$$r_i = (3\nu + 1/2)^{-1}, \quad \text{for } i = D+1, \ldots, i^*. \tag{39}$$

High-order relaxation rates. Without physical justification, it is unclear what the relaxation rates for high-order moments should be; and in fact, most previous methods simply assume them to be 1... However, in turbulent flows (for example, those involving shear instability), it turns out that this choice of value introduces such a strong filtering that many important turbulent details get smeared out. Recently, Li et al. [2018] proposed to use the relation rates of Eq. (39) to determine high-order relaxation rates as well, but with a different, artificially defined "viscosity" value ν_i for each high-order r_i. For example, in 3D, they proposed to use

$$\begin{aligned} \nu_i &= 0.005, \quad \text{for } i = 9, 10, ..., 16, \\ \nu_i &= 0.007, \quad \text{for } i = 17, 18, ..., 22, \\ \nu_i &= 0.009, \quad \text{for } i = 23, 24, ..., 25, \\ \nu_i &= 0.01, \quad \text{for } i = 26. \end{aligned} \tag{40}$$

The choice of these values was empirically found. In stark contrast to this empirical approach, we propose in our main paper to optimize the high-order relaxation rates adaptively, aiming to minimize both dissipation and dispersion errors. It is this new formulation for how to choose r_i that constitutes one of our main technical contributions.

5 EXTERNAL FORCE APPROXIMATION

We now focus on the computation of F_i in Eq. (28) to incorporate external forces. The key challenge here is to compute $\nabla_{\boldsymbol{c}_i} f_i$ appearing in the last term of Eq. (26). Many prior works have investigated the external force model in lattice Boltzmann methods [Fei and Luo 2017; Guo et al. 2002; De Rosis 2017a; Premnath and Banerjee 2011]. A popular model, introduced by Guo et al. [2002], approximates $\nabla_{\boldsymbol{c}_i} f_i$ using $\nabla_{\boldsymbol{c}_i} \bar{f}_i$, which involves only the equilibrium distribution function. In particular, it computes F_i through

$$F_i = \boldsymbol{F} \cdot \nabla_{\boldsymbol{c}_i} f_i \approx \boldsymbol{F} \cdot \nabla_{\boldsymbol{c}_i} \bar{f}_i = (1 - \frac{1}{2\tau}) w_i \left(\frac{\boldsymbol{c}_i - \boldsymbol{u}}{c_s^2} + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^4} \boldsymbol{c}_i \right) \cdot \boldsymbol{F} \ . \tag{41}$$

This approximation can be projected onto the (central) moment space, similarly to what we present in §4 (see derivation in [De Rosis et al. 2019]):

$$\mathbb{F}^* = \mathbf{M}^{-1} \left(\mathbf{I} - \frac{\mathbf{R}}{2} \right) \mathbf{M} \mathbb{F} = \mathbf{M}^{-1} \left(\mathbf{I} - \frac{\mathbf{R}}{2} \right) \tilde{\mathbb{F}}, \tag{42}$$

where the vector $\mathbb{F}$ stacks all F_i at a lattice node (computed using Eq. (41)), $\tilde{\mathbb{F}}$ is the resulting force vector in moment space, and $\mathbb{F}^*$ consists of force terms after moment-space relaxation, which are in turn used for the force term in Eq. (28).

The approximation of the force term in Eq. (41) uses the Hermite expansion only up to the second order (recall Eq. (27)). For high Reynolds number turbulent flows, this is not sufficient and causes unstable artifacts. To improve the approximation accuracy, we expand $\bar{f}_i$ to its highest possible order, similar to what we described in §4.2. In 2D, this gives a fourth-order expansion of F_i [Huang et al. 2018] of the form:

$$\begin{aligned} F_i = w_i \rho \Bigg[& \frac{\boldsymbol{g} \cdot \boldsymbol{c}_i}{c_s} \mathbf{H}_i^{(1)} + \frac{\boldsymbol{g} \otimes \boldsymbol{u}}{2c_s^2} \mathbf{H}_i^{(2)} + \frac{[\boldsymbol{g} \otimes \boldsymbol{u} \otimes \boldsymbol{u}]}{6c_s^3} \Big(\mathbf{H}_{ixyy}^{(3)} \\ & + \mathbf{H}_{ixxy}^{(3)} \Big) + \frac{[\boldsymbol{g} \otimes \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u}]}{24c_s^4} \mathbf{H}_{ixxyy}^{(4)} \Bigg] , \end{aligned} \tag{43}$$

where $[\boldsymbol{F} \otimes \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u}]$ is a shorthand notation for

$$[\boldsymbol{F} \otimes \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u}] = \boldsymbol{F} \otimes \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \boldsymbol{F} \otimes \boldsymbol{u} \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{F} \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{F},$$

and $[\boldsymbol{F} \otimes \boldsymbol{u} \otimes \boldsymbol{u}] = \boldsymbol{F} \otimes \boldsymbol{u} \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \boldsymbol{F} \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{F}$.

In 3D, the Hermite series of F_i expand up to the sixth order [De Rosis et al. 2019], yielding

$$\begin{aligned} F_i = w_i \rho (& \frac{\boldsymbol{g} \cdot \boldsymbol{c}_i}{c_s^2} + \frac{1}{2c_s^4} [\mathbf{H}_{ixx}^{(2)} (2u_x g_x) + \mathbf{H}_{iyy}^{(2)} (2u_y g_y) + \mathbf{H}_{izz}^{(2)} (2u_z g_z) \\ & + 2\mathbf{H}_{ixy}^{(2)} (u_x g_y + u_y g_x) + 2\mathbf{H}_{ixz}^{(2)} (u_x g_z + u_z g_x) \\ & + 2\mathbf{H}_{iyz}^{(2)} (u_y g_z + u_z g_y)] + \frac{1}{2c_s^6} [\mathbf{H}_{ixxy}^{(3)} (2u_x u_y g_x + u_x^2 g_y) \\ & + \mathbf{H}_{ixxz}^{(3)} (2u_x u_z g_x + u_x^2 g_z) + \mathbf{H}_{ixyy}^{(3)} (2u_x u_y g_y + u_y^2 g_x) \\ & + \mathbf{H}_{ixzz}^{(3)} (2u_x u_z g_z + u_z^2 g_x) + \mathbf{H}_{iyzz}^{(3)} (2u_y u_z g_z + u_z^2 g_y) \\ & + \mathbf{H}_{iyyz}^{(3)} (2u_y u_z g_y + u_y^2 g_z) + 2\mathbf{H}_{ixyz}^{(3)} (u_x u_y g_z + u_x g_y u_z + g_x u_y u_z)] \\ & + \frac{1}{4c_s^8} [\mathbf{H}_{ixxyy}^{(4)} (2u_x^2 u_y g_y + 2u_x u_y^2 g_x) + \mathbf{H}_{ixxzz}^{(4)} (2u_x^2 u_z g_z + \\ & 2u_x u_z^2 g_x) + \mathbf{H}_{iyyzz}^{(4)} (2u_y^2 u_z g_z + 2u_y u_z^2 g_y) + 2\mathbf{H}_{ixyzz}^{(4)} (2u_x u_y u_z g_z \\ & + u_x g_y u_z^2 + g_x u_y u_z^2) + 2\mathbf{H}_{ixyyz}^{(4)} (2u_x u_y u_z g_y + u_x u_y^2 g_z + g_x u_y^2 u_z) \\ & + 2\mathbf{H}_{ixxyz}^{(4)} (2u_x u_y u_z g_x + u_y u_x^2 g_z + u_x^2 u_z g_y)] \\ & + \frac{1}{4c_s^{10}} [\mathbf{H}_{ixxyyz}^{(5)} (2u_x u_y^2 u_z g_x + 2u_x^2 u_y u_z g_y + u_x^2 u_y^2 g_z) \\ & + \mathbf{H}_{ixxyzz}^{(5)} (2u_x u_y u_z^2 g_x + u_x^2 u_z^2 g_y + 2u_x^2 u_y u_z g_z) \\ & + \mathbf{H}_{ixyyzz}^{(5)} (u_y^2 u_z^2 g_x + 2u_x u_y u_z^2 g_y + 2u_x u_y^2 u_z g_z)] \\ & + \frac{1}{8c_s^{12}} \mathbf{H}_{ixxyyzz}^{(6)} (2u_x u_y^2 u_z^2 g_x + 2u_x^2 u_y u_z^2 g_y + 2u_x^2 u_y^2 u_z F_z)). \end{aligned} \tag{44}$$

Similar to the projection results in §4.2, the projection of F_i onto the central moment space greatly simplifies their forms — many components vanish, and only a few remain nonzero. In 2D, the nonzero components are

$$\tilde{\mathbb{F}}_1 = F_x, \quad \tilde{\mathbb{F}}_2 = F_y, \quad \tilde{\mathbb{F}}_6 = c_s^2 F_y, \text{ and } \tilde{\mathbb{F}}_7 = c_s^2 F_x \ . \tag{45}$$

In 3D, the nonzero components include

$$\begin{aligned} &\tilde{\mathbb{F}}_1 = F_x, && \tilde{\mathbb{F}}_2 = F_y, && \tilde{\mathbb{F}}_3 = F_z, \\ &\tilde{\mathbb{F}}_{10} = 2c_s^2 F_x, && \tilde{\mathbb{F}}_{11} = 2c_s^2 F_y, && \tilde{\mathbb{F}}_{12} = 2c_s^2 F_z, \\ &\tilde{\mathbb{F}}_{23} = c_s^4 F_x, && \tilde{\mathbb{F}}_{24} = c_s^4 F_y, && \tilde{\mathbb{F}}_{25} = c_s^4 F_z\ . \end{aligned} \tag{46}$$

6 IMMERSED BOUNDARY KINETIC METHOD

We couple fluid dynamics with solid (or deformable) object motion using the classical immersed boundary (IB) method. This method was introduced by Peskin [1973; 1977; 2002] for simulating heart vessel blood flows coupled with deformable heart valves in Navier-Stokes simulation. Feng and Michaelides [2004] were among the first to use the IB method in an LBM framework, and their goal was to simulate rigid disks flowing in a 2D fluid. We refer the reader to a recent paper [Li et al. 2016] for more details on using IB in an LBM framework.

An IB-based kinetic method for two-way coupling can be viewed as a prediction-correction scheme. We first advance the simulation state of the fluid region, ignoring the solid object inside. At the end of this step, the fluid may violate the boundary condition of the solid object. Then, in a correction step, penalty forces (impulses) are computed and locally applied to both the solid object and the fluid to enforce the boundary conditions. More precisely, the IB-based kinetic method proceeds as follows.

- *Solid boundary samples.* We first uniformly sample points on the surface of the solid object (Poisson-disk surface sampling [Yuksel 2015] can be used to generate these point samples). The purpose of sampling surface points is for computing a penalty force at each surface sample in a later step.

- *Momentum interpolation.* We then interpolate the fluid momentum $\boldsymbol{m}_f$ from the lattice grid nodes to the sampled surface points. This is achieved through kernel-based interpolation:

$$\begin{aligned} \boldsymbol{m}_f(\boldsymbol{x}_s) &= \int \boldsymbol{m}_f(\boldsymbol{x})\delta(\boldsymbol{x}-\boldsymbol{x}_s)d\boldsymbol{x} \\ &\approx \sum_{\boldsymbol{x}_f \in \mathcal{D}_s} \boldsymbol{m}_f(\boldsymbol{x}_f)\bar{\delta}(\boldsymbol{x}_f-\boldsymbol{x}_s)\Delta v, \end{aligned} \tag{47}$$

  where $\mathcal{D}_s$ denotes the set of lattice nodes around the solid surface sample at $\boldsymbol{x}_s$; $\boldsymbol{m}_f(\boldsymbol{x}_f)$ is the fluid momentum at the lattice node $\boldsymbol{x}_f$ computed using Eq. (25); and $\Delta v = \Delta x\Delta y$ in 2D and $\Delta v = \Delta x\Delta y\Delta z$ in 3D. Here Δx, Δy and Δz are lattice grid spacings along x-, y- and z-dimensions. $\bar{\delta}(\boldsymbol{x}_f - \boldsymbol{x}_s)$ is a smoothed approximation to the Dirac delta function as the interpolation kernel, whose specific form will be given in Eqs. (50)-(53). The interpolated fluid velocity (i.e., $\boldsymbol{m}_f(\boldsymbol{x}_s)/\rho$) may disagree with the solid object surface velocity at $\boldsymbol{x}_s$. To correct this discrepancy, we compute a penalty force

$$\boldsymbol{F}_{f\to s}(\boldsymbol{x}_s) = \Big(\boldsymbol{m}_f(\boldsymbol{x}_s) - \rho \boldsymbol{v}_s(\boldsymbol{x}_s)\Big)/\Delta t, \tag{48}$$

  where $\boldsymbol{v}_s(\boldsymbol{x}_s)$ is the solid object surface velocity at a surface sample $\boldsymbol{x}_s$.

- *Applying penalty forces.* After computing the penalty forces at surface sample points, we apply them to the solid object. They are also transferred to lattice nodes through a kernel-based interpolation; namely, the force applied to a lattice node $\boldsymbol{x}_f$ is

$$\begin{aligned} \boldsymbol{F}_{s\to f}(\boldsymbol{x}_f) &= -\int \boldsymbol{F}_{f\to s}(\boldsymbol{x})\delta(\boldsymbol{x}-\boldsymbol{x}_f)d\boldsymbol{x} \\ &= -\sum_{\boldsymbol{x}_s \in \mathcal{D}_f} \boldsymbol{F}_{f\to s}(\boldsymbol{x}_s)\bar{\delta}(\boldsymbol{x}_s-\boldsymbol{x}_f)\Delta v, \end{aligned} \tag{49}$$

  where $\mathcal{D}_f$ denotes the set of surface samples around a lattice node at $\boldsymbol{x}_f$. At each timestep, these forces are added to the external force term to update F_i as described in §5.

Lastly, we present the specific forms of the kernel function $\bar{\delta}(\boldsymbol{x}_j - \boldsymbol{x}_i)$ for 2D and 3D cases. In 2D, it is defined as

$$\bar{\delta}(\boldsymbol{x}_j-\boldsymbol{x}_i) = \frac{1}{\Delta x}\hat{\delta}\left(\frac{|x_j-x_i|}{\Delta x}\right)\frac{1}{\Delta y}\hat{\delta}\left(\frac{|y_j-y_i|}{\Delta y}\right), \tag{50}$$

and in 3D, it has the following form:

$$\bar{\delta}(\boldsymbol{x}_j-\boldsymbol{x}_i) = \frac{1}{\Delta x}\hat{\delta}\left(\frac{|x_j-x_i|}{\Delta x}\right)\frac{1}{\Delta y}\hat{\delta}\left(\frac{|y_j-y_i|}{\Delta y}\right)\frac{1}{\Delta z}\hat{\delta}\left(\frac{|z_j-z_i|}{\Delta z}\right). \tag{51}$$

In these definitions, we use the smoothed Dirac $\hat{\delta}$ function from [Li et al. 2016], namely,

$$\hat{\delta}(r) = \begin{cases} \frac{1}{3}(1+\sqrt{-3r^2+1}), & 0 \le r \le 0.5 \\ \frac{1}{6}(5-3r-\sqrt{-3(1-r)^2+1}), & 0.5 \le r \le 1.5 \\ 0, & \text{otherwise.} \end{cases} \tag{52}$$

where the interpolation kernel size is 3. To preserve flow sharpness around solid boundaries, we can also adopt a kernel size of 2 [Griffith and Patankar 2020], which we use in our paper:

$$\hat{\delta}(r) = \begin{cases} 1-r, & 0 \le r \le 1 \\ 0, & \text{otherwise.} \end{cases} \tag{53}$$

7 DIMENSIONAL SCALING

Lattice Boltzmann simulations are performed in dimensionless normalized units. Indeed, we introduced our lattice Boltzmann equation (LBE) solver in Eq. (28) with $\Delta t = 1$ and $c = 1$. Formally, it takes two steps to convert dimensional physical scales to dimensionless lattice scales [Latt 2007]: 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, $\boldsymbol{x}$, ρ, $\boldsymbol{u}$, $\boldsymbol{a}$ 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:

$$\begin{aligned} \hat{t} = \frac{t}{t_{\text{ref}}},\ \ \hat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{l_{\text{ref}}},\ \ \hat{\rho} = \frac{\rho}{\rho_{\text{ref}}},\ \ \hat{\boldsymbol{u}} = \frac{\boldsymbol{u}}{u_{\text{ref}}} = \boldsymbol{u}\frac{t_{\text{ref}}}{l_{\text{ref}}}, \\ \hat{\boldsymbol{a}} = \frac{\boldsymbol{a}}{a_{\text{ref}}} = \boldsymbol{a}\frac{t_{\text{ref}}^2}{l_{\text{ref}}},\ \text{and}\ \hat{p} = \frac{p}{\rho_{\text{ref}}}\frac{t_{\text{ref}}^2}{l_{\text{ref}}^2}, \end{aligned} \tag{54}$$

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

$$Re = \frac{u_{\text{ref}}\, l_{\text{ref}}}{\nu} = \frac{l_{\text{ref}}^2}{t_{\text{ref}}\, \nu}, \tag{55}$$

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. The Reynolds number also relates to the *law of similarity* in fluid dynamics, which states that two incompressible flow systems are dynamically similar if they have the same Reynolds number and geometry.

Next, we convert the variables in normalized lattice units. Since the characteristic length l_{ref} and time scale t_{ref} are both unit when expressed in dimensionless units, we consider the finest spatial and time lengths that an LBE solver can resolve, and denote those lengths in dimensionless units as

$$\Delta\hat{x} = \frac{1}{N}, \quad \text{and} \quad \Delta\hat{t} = \frac{1}{T}, \tag{56}$$

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:

$$\begin{aligned}
\mathbb{t} &= \frac{\hat{t}}{\Delta\hat{t}} = \frac{t}{t_{\text{ref}}\Delta\hat{t}} = \frac{Tt}{t_{\text{ref}}}, \ \mathbb{x} = \frac{\hat{\boldsymbol{x}}}{\Delta\hat{x}} = \frac{\boldsymbol{x}}{l_{\text{ref}}\Delta\hat{x}} = \frac{N\boldsymbol{x}}{l_{\text{ref}}}, \\
\mathbb{p} &= \hat{\rho} = \frac{\rho}{\rho_{\text{ref}}}, \ \mathbb{u} = \hat{\boldsymbol{u}}\frac{\Delta\hat{t}}{\Delta\hat{x}} = \boldsymbol{u}\frac{t_{\text{ref}}}{l_{\text{ref}}}\frac{\Delta\hat{t}}{\Delta\hat{x}} = \boldsymbol{u}\frac{t_{\text{ref}}}{l_{\text{ref}}}\frac{N}{T}, \\
\mathbb{a} &= \hat{\boldsymbol{a}}\frac{\Delta\hat{t}^2}{\Delta\hat{x}} = \boldsymbol{a}\frac{t_{\text{ref}}^2}{l_{\text{ref}}}\frac{\Delta\hat{t}^2}{\Delta\hat{x}} = \boldsymbol{a}\frac{t_{\text{ref}}^2}{l_{\text{ref}}}\frac{N}{T^2}, \\
\mathbb{p} &= \hat{p}\frac{\Delta\hat{t}^2}{\Delta\hat{x}^2} = \frac{p}{\rho_{\text{ref}}}\frac{t_{\text{ref}}^2}{l_{\text{ref}}^2}\frac{\Delta\hat{t}^2}{\Delta\hat{x}^2} = \frac{p}{\rho_{\text{ref}}}\frac{t_{\text{ref}}^2}{l_{\text{ref}}^2}\frac{N^2}{T^2}, \ \text{and} \\
\mathbb{v} &= \hat{\nu}\frac{\Delta\hat{t}}{\Delta\hat{x}^2} = \nu\frac{\Delta\hat{t}}{\Delta\hat{x}^2}\frac{t_{\text{ref}}}{l_{\text{ref}}^2} = \nu\frac{N^2}{T}\frac{t_{\text{ref}}}{l_{\text{ref}}^2}.
\end{aligned} \tag{57}$$

In lattice units, both the lattice grid size $\Delta\mathbb{x}$ and timestep $\Delta\mathbb{t}$ have unit lengths. In what follows, we rewrite the conversions listed in Eq. (57) 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 $\boldsymbol{u}_{\text{in}}$ as the reference velocity, i.e., $\boldsymbol{u}_{\text{in}} = u_{\text{ref}}\,\boldsymbol{r}$ where $\boldsymbol{r}$ 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. (57) and obtain

$$\begin{aligned}
&\mathbb{t} = \frac{Tt}{t_{\text{ref}}} = \frac{u_{\text{ref}}Tt}{l_{\text{ref}}}, \quad \mathbb{x} = \frac{N\boldsymbol{x}}{l_{\text{ref}}}, \quad \mathbb{p} = \frac{\rho}{\rho_{\text{ref}}}, \\
&\mathbb{u} = \boldsymbol{u}\frac{t_{\text{ref}}}{l_{\text{ref}}}\frac{N}{T} = \boldsymbol{u}\frac{N}{u_{\text{ref}}T}, \quad \mathbb{a} = \boldsymbol{a}\frac{t_{\text{ref}}^2}{l_{\text{ref}}}\frac{N}{T^2} = \boldsymbol{a}\frac{l_{\text{ref}}}{u_{\text{ref}}^2}\frac{N}{T^2}, \\
&\mathbb{p} = \frac{p}{\rho_{\text{ref}}}\frac{t_{\text{ref}}^2}{l_{\text{ref}}^2}\frac{N^2}{T^2} = \frac{p}{\rho_{\text{ref}}}\frac{N^2}{u_{\text{ref}}^2T^2}, \quad \mathbb{v} = \nu\frac{N^2}{T}\frac{t_{\text{ref}}}{l_{\text{ref}}^2} = \nu\frac{N^2}{u_{\text{ref}}l_{\text{ref}}T}.
\end{aligned} \tag{58}$$

To ensure stable simulation, the condition $\Delta\hat{t} = \lambda\Delta\hat{x}$ for a value λ satisfying $\lambda < c_s$ must hold. From (56), we have the relation $T = \lambda^{-1}N$, which we then use in (58) and obtain

$$\begin{aligned}
&\mathbb{t} = \frac{u_{\text{ref}}Nt}{\lambda l_{\text{ref}}}, \quad \mathbb{x} = \frac{N\boldsymbol{x}}{l_{\text{ref}}}, \quad \mathbb{p} = \frac{\rho}{\rho_{\text{ref}}}, \\
&\mathbb{u} = \lambda\frac{\boldsymbol{u}}{u_{\text{ref}}}, \quad \mathbb{a} = \lambda^2\boldsymbol{a}\frac{l_{\text{ref}}}{Nu_{\text{ref}}^2}, \\
&\mathbb{p} = \frac{p}{\rho_{\text{ref}}}\frac{\lambda^2}{u_{\text{ref}}^2}, \quad \mathbb{v} = \lambda\nu\frac{N}{u_{\text{ref}}l_{\text{ref}}}.
\end{aligned} \tag{59}$$

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 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 $\mathbb{u}_{\text{ref}}$. From the fourth relation in Eq. (59), we have $\lambda = \mathbb{u}_{\text{ref}}$, and using it to replace λ gives us the following relations:

$$\begin{aligned}
&\mathbb{t} = \frac{u_{\text{ref}}}{\mathbb{u}_{\text{ref}}\Delta x}t, \quad \mathbb{x} = \frac{\boldsymbol{x}}{\Delta x}, \quad \mathbb{p} = \frac{\rho}{\Delta x}, \\
&\mathbb{u} = \frac{\mathbb{u}_{\text{ref}}\boldsymbol{u}}{u_{\text{ref}}}, \quad \mathbb{a} = \frac{\mathbb{u}_{\text{ref}}^2\Delta x}{u_{\text{ref}}^2}\boldsymbol{a}, \\
&\mathbb{p} = \frac{p}{\rho_{\text{ref}}}\frac{\mathbb{u}_{\text{ref}}^2}{u_{\text{ref}}^2}, \quad \mathbb{v} = \frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}\Delta x}\nu.
\end{aligned} \tag{60}$$

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 (28), the LBE solver uses a unit timestep size $\Delta\mathbb{t} = 1$. Then, according to the first relation in Eq. (60), 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 $\boldsymbol{F} = \rho\boldsymbol{a}$ can also be converted in lattice units:

$$\mathbb{F} = \rho\mathbb{a} = \frac{\rho}{\rho_{\text{ref}}}\frac{\mathbb{u}_{\text{ref}}^2\Delta x}{u_{\text{ref}}^2}\boldsymbol{a} = \frac{\mathbb{u}_{\text{ref}}^2\Delta x}{\rho_{\text{ref}}u_{\text{ref}}^2}\boldsymbol{F}. \tag{61}$$

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

$$n_K = \frac{u_{\text{ref}}}{K\mathbb{u}_{\text{ref}}\Delta x}. \tag{62}$$

When the fluid is coupled with moving solids, the solid object simulator (such as the Bullet solver [Coumans and Bai 2019]) 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.

7.1 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 ($\boldsymbol{\nu}$), initial simulation values ($\boldsymbol{\rho}$, $\boldsymbol{u}$, and $\boldsymbol{g}$) 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 $\mathbb{u}_{\text{ref}}$, $\boldsymbol{\nu}$ and n_K) and the grid resolution N. We must then convert them to physical units through

$$u_{\text{ref}} = \mathbb{u}_{\text{ref}}\sqrt{\frac{Kn_K\nu}{\boldsymbol{\nu}}} \text{ and } \Delta x = \sqrt{\frac{\nu}{Kn_K\boldsymbol{\nu}}}, \tag{63}$$

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

Notice that the viscosity ν is a fixed material parameter in physical unit. Eq. (63) shows that when $\mathbb{u}_{\text{ref}}$ and the frame rate K are given, increasing the viscosity in lattice unit 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 increasing $\boldsymbol{\nu} \rightarrow \alpha\boldsymbol{\nu}$ 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 $\boldsymbol{\nu} \rightarrow \alpha\boldsymbol{\nu}$, we can select $n_K \rightarrow n_K/\alpha$, which in turn reduces $u_{\text{ref}} \rightarrow u_{\text{ref}}/\alpha$.

7.2 Allowable ranges of spatial and temporal spacings

Given a lattice size Δx (often determined by the scene to be simulated), we can define the time step in physical unit as:

$$\Delta t = \frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}}\Delta x, \tag{64}$$

where $\Delta\mathbb{t} = 1$ since we use normalized LBE unit. This indicates that if $\mathbb{u}_{\text{ref}}$ is fixed, then we can change u_{ref} in order to control the time step size in physical unit. However, u_{ref} cannot be freely tuned, but should be constrained to make LBE stable. This can be understood by first expressing:

$$\frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}} = \frac{\Delta t}{\Delta x}, \tag{65}$$

and then inserting into the dimensional scaling of velocity to have:

$$\|\mathbb{u}\|_2 = \frac{\Delta t}{\Delta x}\|\boldsymbol{u}\|_2 \leq \frac{\Delta t}{\Delta x}u_{\max} \leq \mathbb{u}_{\max}, \tag{66}$$

where $u_{\max}$ is the maximum velocity magnitude in physical unit, and $\mathbb{u}_{\max}$ is the maximum velocity magnitude allowable in normalized LBE scale, which is usually $\mathbb{u}_{\max}$=βc_s where $\beta \approx 0.5$ in our kinetic solver. This implies that

$$\Delta t = \frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}}\Delta x \leq \frac{\mathbb{u}_{\max}}{u_{\max}}\Delta x = \frac{\beta c_s}{u_{\max}}\Delta x. \tag{67}$$

Thus we have a constraint on selecting u_{ref}:

$$u_{\text{ref}} \geq \frac{u_{\max}\mathbb{u}_{\text{ref}}}{\beta c_s}. \tag{68}$$

However, u_{ref} should not be too large, which is limited by the floating-point precision, as can be seen from:

$$\epsilon \leq \|\mathbb{u}\|_2 = \frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}}\|\boldsymbol{u}\|_2 \leq \frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}}u_{\max}, \tag{69}$$

where ϵ is a small threshold. Since $\mathbb{u}$ will be involved in central-moment matrix computation, which may be powered by several orders, we select ϵ=10^{-2}. Thus, we have:

$$u_{\text{ref}} \leq \frac{u_{\max}\mathbb{u}_{\text{ref}}}{\epsilon}. \tag{70}$$

The range of u_{ref} is then:

$$\frac{u_{\max}\mathbb{u}_{\text{ref}}}{\beta c_s} \leq u_{\text{ref}} \leq \frac{u_{\max}\mathbb{u}_{\text{ref}}}{\epsilon}, \tag{71}$$

which leads to the range of tunable Δt as:

$$\frac{\epsilon}{u_{\max}}\Delta x \leq \Delta t \leq \frac{\beta c_s}{u_{\max}}\Delta x. \tag{72}$$

In case we constrain that u_{ref}=$u_{\max}$, this places a constraint on $\mathbb{u}_{\text{ref}}$ as:

$$\epsilon \leq \mathbb{u}_{\text{ref}} \leq \beta c_s, \tag{73}$$

leading to the same range of tunable Δt as in Eq. (72).

REFERENCES

Shiyi Chen and Gary D Doolen. 1998. Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics* 30, 1 (1998), 329–364.

Erwin Coumans and Yunfei Bai. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Alessandro De Rosis. 2017a. Alternative formulation to incorporate forcing terms in a lattice Boltzmann scheme with central moments. *Physical Review E* 95, 2 (2017), 023311.

Alessandro De Rosis. 2017b. Non-orthogonal central moments relaxing to a discrete equilibrium: A D2Q9 lattice Boltzmann model. *EPL (Europhysics Letters)* 116, 4 (2017), 44003.

Alessandro De Rosis. 2017c. Nonorthogonal central-moments-based lattice Boltzmann scheme in three dimensions. *Physical Review E* 95, 1 (2017), 013310.

Alessandro De Rosis, Rongzong Huang, and Christophe Coreixas. 2019. Universal formulation of central-moments-based lattice Boltzmann method with external forcing for the simulation of multiphysics phenomena. *Physics of Fluids* 31, 11 (2019), 117102.

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

Linlin Fei and Kai Hong Luo. 2017. Consistent forcing scheme in the cascaded lattice Boltzmann method. *Physical Review E* 96, 5 (2017), 053307.

Zhi-Gang Feng and Efstathios E Michaelides. 2004. The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems. *J. Comput. Phys.* 195, 2 (2004), 602–628.

M Geier, A Greiner, and JG Korvink. 2006. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Physical review. E, Statistical, nonlinear, and soft matter physics* 73, 6 Pt 2 (2006), 066705–066705.

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.

Boyce E Griffith and Neelesh A Patankar. 2020. Immersed Methods for Fluid–Structure Interaction. *Annual Review of Fluid Mechanics* 52 (2020).

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 (2002), 046308.

Rongzong Huang, Huiying Wu, and Nikolaus A Adams. 2018. Eliminating cubic terms in the pseudopotential lattice Boltzmann model for multiphase flow. *Physical Review E* 97, 5 (2018), 053308.

Jonas Latt. 2007. *Hydrodynamic limit of lattice Boltzmann equations.* Ph.D. Dissertation. University of Geneva.

Wei Li, Kai Bai, and Xiaopei Liu. 2018. Continuous-Scale Kinetic Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* (2018).

Zhe Li, Julien Favier, Umberto D'Ortona, and Sébastien Poncet. 2016. An immersed boundary-lattice Boltzmann method for single-and multi-component fluid flows. *J. Comput. Phys.* 304 (2016), 424–440.

Patrick Nathen, Daniel Gaudlitz, Mathias J Krause, and Nikolaus A Adams. 2017. On the stability and accuracy of the BGK, MRT and RLB Boltzmann schemes for the simulation of turbulent flows. *J. Commun. Comput. Phys* 23 (2017), 846–876.

Charles Peskin. 1973. Flow patterns around heart valves: a digital computer method for solving the equations of motion. *IEEE Transactions on Biomedical Engineering* 4 (1973), 316–317.

Charles S Peskin. 1977. Numerical analysis of blood flow in the heart. *Journal of computational physics* 25, 3 (1977), 220–252.

Charles S Peskin. 2002. The immersed boundary method. *Acta numerica* 11 (2002), 479–517.

Kannan N Premnath and Sanjoy Banerjee. 2009. Incorporating forcing terms in cascaded lattice Boltzmann approach by method of central moments. *Physical Review E* 80, 3 (2009), 036702.

Kannan N Premnath and Sanjoy Banerjee. 2011. On the three-dimensional central moment lattice Boltzmann method. *Journal of Statistical Physics* 143, 4 (2011), 747–794.

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.

Cem Yuksel. 2015. Sample elimination for generating Poisson disk sample sets. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 25–32.