Kinetic-based Multiphase Flow Simulation

Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu

Abstract—Multiphase flows exhibit a large realm of complex behaviors such as bubbling, glugging, wetting, and splashing which emerge from air-water and water-solid interactions. Current fluid solvers in graphics have demonstrated remarkable success in reproducing each of these visual effects, but none have offered a model general enough to capture all of them concurrently. In contrast, computational fluid dynamics have developed very general approaches to multiphase flows, typically based on kinetic models. Yet, in both communities, there is dearth of methods that can simulate density ratios and Reynolds numbers required for the type of challenging real-life simulations that movie productions strive to digitally create, such as air-water flows. In this paper, we propose a kinetic model of the coupling of the Navier-Stokes equations with a conservative phase-field equation, and provide a series of numerical improvements over existing kinetic-based approaches to offer a general multiphase flow solver. The resulting algorithm is embarrassingly parallel, conservative, far more stable than current solvers even for real-life conditions, and general enough to capture the typical multiphase flow behaviors. Various simulation results are presented, including comparisons to both previous work and real footage, to highlight the advantages of our new method.

Index Terms—multiphase flow, kinetic theory, phase-field lattice Boltzmann model, interface phenomena

1 Introduction

Fluid simulation has been an important research topic in computer graphics (CG) for decades, and by now, realistic bodies of water or complex smoke plumes are routinely produced in movie productions. Until about a decade ago, the vast majority of fluid animation works focused on single-phase simulation [1]–[5] where the air surrounding the fluid is assumed to be exerting no pressure on the surface of the fluid, with the unfortunate consequence that any amount of air trapped under water collapses instantly instead of forming bubbles. More recently, the CG community turned its attention to multiphase flows, in which simultaneous evolutions of interacting fluids are computed in order to capture important visual phenomena emerging from the air-water and water-solid interplay such as splashing, sloshing, glugging, boiling, cavitation, or even the complex wetting patterns that a fluid creates over hydrophilic or hydrophobic surfaces.

Among the rich gamut of behaviors that immiscible multiphase flows exhibit, specific effects involving bubbles [6]–[8], surface tension flows [9], [10] or even wetting [11], [12] have been successfully addressed in our field. However, these approaches typically rely on simplified models to achieve each characteristic behavior efficiently, but none can simulate real-life examples of multiphase flows exhibiting all these behaviors at once. Moreover, most CG techniques are unable to numerically capture the type of multiphase flows they strive to emulate, i.e., those with density ratios (800 for air vs. water for instance, since $\rho_{\text{air}} = 1.2\,kg/m^3$ and $\rho_{\text{water}} = 1000\,kg/m^3$) and Reynolds numbers ($\text{Re} > 4,000$ for turbulent water flows) that are seen in natural phenomena. This limitation, while rarely mentioned, both restricts the range of simulations that can be handled through digital animation and negatively impacts visual realism.

Multiphase flows have attracted even greater interest in the computational fluid dynamics (CFD) community, see e.g. [13]–[16]. In contrast to CG, there has been a slew of efforts to develop unified numerical methods capable of capturing *all* of the interesting behaviors at once. A large number of works have also been dedicated to handle high density ratios, or high Reynolds numbers, far beyond what CG fluid solvers can handle — although none can boast of offering a stable treatment of multiphase flows with *both* large density ratio and high Reynolds number, surprisingly.

Currently, the most general and efficient solvers that can reproduce all the typical characteristics of multiphase flows are based on the lattice Boltzmann method (LBM), combined with a conservative phase-field (PF) model to handle interfacial computations [17]. This is a significant departure from the most common approaches used in graphics: these solvers rely on a kinetic formulation of the flow derived from statistical mechanics using the Boltzmann (transport) equation of a probability distribution for the position and velocity of fluid particles. While the original formulation of LBM using explicit time stepping with local spatial interactions (thus, devoid of global solves) had been recognized in graphics as highly parallelizable [18]–[21], it quickly fell into disuse due to its substandard visual results and its limited stability and/or accuracy with respect to density ratios and Reynolds numbers. Yet, LBM has experienced a series of developments in recent years [22]–[24], especially with the development of central-moment relaxation models and coupling with Shan-Chen, free energy, or phase-field models [25]. As a result, modern

- *Wei Li, Daoming Liu and Xiaopei Liu are all with the School of Information Science and Technology, ShanghaiTech University, China. E-mails: {liwei, liudm, liuxp}@shanghaitech.edu.cn.*
- *Jin Huang is with the State Key Lab of CAD & CG, College of Computer Science, Zhejiang University, China. Email: hj@cad.zju.edu.cn.*
- *Mathieu Desbrun is with the School of Information Science and Technology, ShanghaiTech University, China, during his sabbatical, and with the Computing + Mathematical Sciences department, Caltech, USA. Email: mathieu@cms.caltech.edu.*

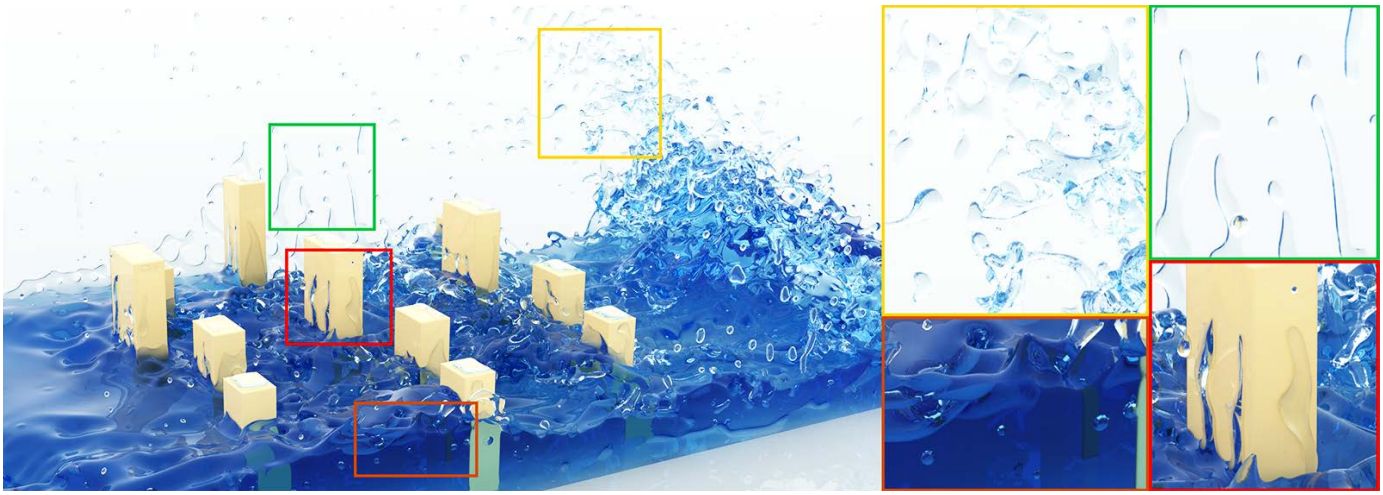

Fig. 1: **Efficient simulation of multiphase flows.** Snapshot of a simulation of water dam breaking through solid obstacles, where the materials of obstacles and domain boundaries are set to be hydrophilic and hydrophobic, respectively. The water flows from left to right, passing through a set of solid pillars with different sizes and heights. During this process, several typical phenomena associated with water happen at different stages of the simulation, such as water splashing due to interface splitting (see the yellow box), wetting on solid surfaces (see the green and red boxes for hydrophilic wetting on obstacles and hydrophobic wetting on domain boundaries), as well as bubbling (see the orange box), requiring a unified model and solver for the simulation to capture all of these multi-phase phenomena simultaneously. Our kinetic method faithfully achieves this goal, which substantially increases the visual realism for complex liquid simulations.

LBM approaches have made great strides in accuracy and stability without sacrificing its efficiency [26], to the point that they have been recently adopted in the automotive and aerospace industries for CFD-aided design [27].

Overview. In this paper, we revisit the kinetic approach to fluid simulation to provide an efficient, yet unified fluid solver able to capture all relevant visual behaviors of multiphase flows. We demonstrate that the main weakness of the current state-of-the-art lattice Boltzmann techniques coupled with a phase-field model resides in two crucial deficiencies in the numerical treatment of the phase field: a spurious rotational dependence of its discrete gradient evaluation, and the use of a single-relaxation time, both conspiring to create artificially large velocities near interfaces and thus ruining both accuracy and stability. We introduce a fix for each issue, and show that the resulting LBM-PF numerical scheme offers a unified framework to simulate most multiphase flow visual effects (Fig. 1) that is not only as *efficient* as many of the existing CG simulation techniques, but also vastly *more stable and accurate* than existing CFD approaches to multiphase flows (Fig. 7).

2 Background and Motivation

In order to motivate our approach, we first mention related works in graphics (focusing on liquids for conciseness), before reviewing LBM-based kinetic approaches for handling multiphase flow phenomena in CFD.

2.1 Single-phase flows

Early CG techniques relied on free-surface flow simulation ignoring air pressure on interfaces.

Interface tracking methods. One of the most commonly used methods to track a fluid interface is the level-set method [28], [1]. Its inherent volume loss led Enright et al. [29], [30] to propose a particle level-set method which significantly reduced the issue, while Kim et al. [31] designed a controller-based method. Increased sampling near the interface to improve visual details was offered by Losasso et al. [32] through an octree structure, while Heo and Ko [33] used sub-grid quadrature points; up-resing and closest point turbulence as post-processing steps were also proposed by Kim et al. [34] to further improve surface details. An alternative tracking approach from CFD, the volume-of-fluid (VOF) approach was also proposed to precisely preserve fluid volume [2], [35], but reconstructing a visually appealing interface from grid volume fractions is notoriously difficult. In order to avoid the artifacts that the aforementioned implicit representations may generate, explicit representations and tracking of interfaces have been proposed in recent years [5], [36]–[38], and a few examples of surface-only simulation of fluids solely using an explicit surface representation have even been demonstrated [10].

Lagragian particle methods. In contrast to the Eulerian methods described above, a number of approaches implement a Lagrangian description of a fluid through a discretization of its volume via "particles". A first approach, based on smoothed particle hydrodynamics (SPH) has been widely used for liquid simulations [39] (and even for ferrofluids recently [40]), but it requires proper interface reconstruction from unstructured boundary particles. Müller et al. [41] proposed a simple smooth kernel to reconstruct the liquid surfaces, which often looks blobby. This issue was mostly addressed in Zhu and Bridson [3] through normalized weight, and further improved by Adams et al. [42] and Yu and Turk [43]. Solenthaler and Pajarola [4] introduced an incompressible SPH method, while Schechter and Bridson [44] improved the numerical treatment of free surface and solid boundaries through a ghost fluid SPH approach, followed by He et al. [45] who

offered an improved handling of sparsely-sampled and thin features. Macklin and Müller [46] proposed a position-based alternative to SPH allowing large time steps suitable for real-time applications, while other authors focused on improving incompressibility [47] and boundary treatment around solids [48], [49]. Another common approach to fluid simulation for graphics is via the Particle-In-Cell (PIC) method, which combines grid and particles for fast pressure solves [50]. While its original formulation exhibits strong numerical diffusion, the Fluid-Implicit-Particle (FLIP) method was later used to mitigate this issue [3], [51]. Ando et al. [52] further combined FLIP with adaptive particle sampling, while Cornels et al. [53] combined FLIP with SPH; for flows with thin obstacles and narrow gaps, Azevedo et al [54] proposed a modified FLIP method. Additionally, Fu et al. [55] addressed the issue of momentum conservation through a polynomial version PIC, based on an earlier affine version [56]. Finally, up-resing the particles coupled with a surface-only wave simulation was proposed by Mercier et al. [57] to increase the visual complexity of particle-based liquid simulations.

2.2 Multiphase flow methods

Free-surface methods can produce splashes with surface tension, but cannot produce bubbles reliably (despite efforts in this direction, see [8]) or exhibit other multiphase phenomena such as wetting. While ad-hoc methods were introduced to efficiently model clusters of bubbles [**?**], [58], two-phase fluid solvers are needed to obtain a good range of convincing multiphase flow effects. Consequently, interface tracking methods were adapted to handle multiphase flows, either using regional level-sets [59] or VOF methods [60], [61] to properly track different fluid components. Similarly, SPH methods were adapted to handle multi-fluid simulation [62]–[64], as well as FLIP techniques [7], and even hybrid methods such as the Material-Point Method (MPM [65]–[67]) were introduced. A recent approach, called power particles, leveraged power diagrams to offer a geometric alternative to multiphase SPH and FLIP with localized control of volume and better local momentum preservation [68], [69]. By incorporating specific surface tension or adhesion force models for instance, each of these multiphase flow techniques was shown to visually capture some of the intricate phenomena of multiphase flows such as wetting or glugging. However, rare are the methods that can capture the whole gamut of multiphase flows behaviors. For instance, methods appropriate to simulate bubbling and glugging cannot handle wetting, and vice-versa. Moreover, most of these methods become unstable for turbulent flows or for density ratios over 10 (note that [70] supports density ratios up to 100, but cannot handle turbulent multiphase flows), rendering them unable to simulate the typical water-air interactions that movie productions typically wish to produce digitally.

2.3 Kinetic methods

Initially proposed by the CFD community, kinetic models take a very different approach to fluid simulation. Instead of directly discretizing Navier-Stokes equations, they rely on statistical mechanics instead, focusing on the evolution in time of the probability density encoding the presence of fluid volumes at a given position with a given velocity. The lattice Boltzmann method (LBM) simulates Boltzmann (transport) equation of this probability distribution, which amounts to move fictitious particles through consecutive propagation (or transport, also called streaming) and collision (or relaxation) processes over a discrete lattice mesh. The simplicity of the resulting algorithm, tantamount to an automata, makes it highly parallelizable, and its kinetic nature allows the incorporation of microscopic interactions to offer more complex flow behaviors while remaining conservative by construction. Defining a proper relaxation to approximate the fluid equation well has been the most difficult aspect of LBM: the Bhatnagar-Gross-Krook (BGK) [13] and multiple-relaxation-time (MRT) [71] models used in early LBM methods had relatively low approximation order and were not very stable. Recently, so-called "central-moment" relaxation models [72] have been proposed to guarantee higher accuracy and stability, but at the cost of higher computational times. In the last decade or so, kinetic methods have been used in CG to simulate smoke or free-surface liquid flows [73]–[79], and recently, Guo et al. [80] proposed a two-phase kinetic method for liquids — but here again, demonstrated density ratios were below 20, and wetting was not considered.

Kinetic models have been extended to multiphase flow simulation in the CFD community, with variants ranging from color-gradient models [81]–[83], [24], pseudo-potential models [84]–[86], free-energy models [87], [88], and phase-field models [89], [17]. This last approach, to which we will refer as LBM-PF since it combines phase field for the encoding of the interface with an LBM simulation of the fluids, has been widely accepted as the current standard approach for multiphase flows, e.g., see [25]. Yet, while some of these LBM-PF methods demonstrated improved handling of large density ratios *or* improved stability for high Reynolds numbers, none can handle both of these required characteristics concurrently.

2.4 Contributions

In this paper, we propose a novel multiphase flow solver. We adopt the kinetic-based LBM-PF formalism, designed to run efficiently on massively parallel architectures, to contribute a unified approach for simulating real-world multiphase flows both efficiently and accurately. Our solver include several contributions addressing key issues of the current LBM-PF algorithm:

- *Rotationally-symmetric stencils*: We noticed that the accuracy of the typical approximation of the phase-field gradient near an interface is particularly sensitive to the normal direction and curvature of the interface, at times causing large numerical artifacts that affect the fine dynamics coupling the two phases. Thus, we propose a new rotationally-symmetric discretization, allowing for both low- or high-order accuracy, to reduce significantly this error and improve stability.
- *Variational weighted scheme*: Through the analysis of a phase-field encoding a spherical interface, we introduce a variational approach to derive an optimal weighted sum of low- and high-order rotationally-symmetric ap-

proximations of the gradient in order to remove most oscillations around sharp interfaces. Akin to the Weighted Essentially Non-Oscillatory (WENO) schemes used in level-set methods, our blending of multi-order operators further reduces the errors in gradient evaluation, even close to sharp transitions of the phase field.

- *Variational weighted upwinding*: We also introduce an upwinding treatment of hyperbolic terms present in the collision model, to replace the standard practice of using the macroscopic (average) velocity and a centered approximation of the phase-field gradient which is known to generate spurious oscillations near interfaces. This novel evaluation significantly enhances stability for large Reynolds numbers and large density ratios.
- *Relaxation for phase field*: We identify the use of a single-relaxation-time collision model for the phase-field equation as being responsible for large numerical errors at high Reynolds numbers. Switching to a weighted multiple-relaxation-time (WMRT) model leads to a significant reduction of numerical artifacts, and increases stability when strong turbulence is present.
- *Velocity-limiting adaptive viscosity*: While the aforementioned numerical improvements greatly improve the stability of our LBM-PF approach, the case of a flow with high density ratio *and* a high Reynolds number can still lead to spurious oscillations near interfaces. We thus add a final filtering of the velocity field which selectively cuts off very large velocities through localized viscosity addition.

These numerical contributions, along with secondary improvements such as the use of a subgrid model based on large-eddy turbulence and a soft-start initialization to prevent early transient compression waves, allow us to simulate multiphase flows exhibiting a wide range of typical complex behaviors: Fig. 1 shows an example where splashing, surface waves, bubbling, and wetting can all be captured with our solver. A variety of simulation results are demonstrated, and comparisons to both real experiments and existing solvers are provided to highlight the advantages of our new method compared to both CG and CFD solvers in terms of both efficiency and accuracy.

3 Multiphase Fluid Model

In this section, we briefly remind the reader of the traditional macroscopic model for multiphase fluid simulation, before reviewing its mesoscopic version based on the Boltzmann transport equation that the LBM-PF solve numerically and efficiently on a discrete grid. Our contributions will build on top of these basic foundations.

3.1 Macroscopic model

In order to simplify our explanations, let us consider the case of a two-phase flow first — we will discuss wetting against solid objects later to extend our approach to multiphase flows (see Eq. 33). Immiscible, incompressible and isothermal two-phase flows are well modeled by the Navier-Stokes equations coupled with the conservative phase-field equation [17]; they are macroscopically described through the following set of equations:

$$\begin{cases} \nabla\cdot\mathbf{u} = 0, & \text{(1a)}\\ \dfrac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0, & \text{(1b)}\\ \dfrac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) = -\nabla p + \rho\nu\nabla^2\mathbf{u} + \mathbf{F}, & \text{(1c)}\\ \dfrac{\partial\phi}{\partial t} + \nabla\cdot(\phi\mathbf{u}) = \nabla\cdot\left[M\left(\nabla\phi - \dfrac{4}{\xi}\phi(1-\phi)\mathbf{n}\right)\right], & \text{(1d)} \end{cases}$$

$$\text{with}\quad \mathbf{F} = \rho\mathbf{g} + \left[4\eta\phi(\phi-1)\left(\phi-\frac{1}{2}\right) - \kappa\nabla^2\phi\right]\nabla\phi\,, \quad \text{(2)}$$

$$\text{and}\quad \rho(\phi) = (1-\phi)\rho_L + \phi\rho_H. \quad \text{(3)}$$

Eqs. (1a–c) are the typical equations governing the motion of an incompressible fluid (encoded by its velocity $\mathbf{u}$) in its two possible phases (encoded by the density field ρ), while Eq. (1d) is the conservative phase-field (PF) equation that describe the (diffuse) interface motion between the two fluids. The phase field ϕ is a spatially- and temporally-varying scalar field with values in $[0,1]$ indicating the percentage of a particular phase at a given location and time, with the convention that $\phi=1$ for the high-density (ρ_H) phase (i.e., liquid) and $\phi=0$ for the low-density (ρ_L) phase (i.e., air). Note that the PF equation (1d) simply indicates that the phase field is advected in the velocity field with an additional conservative term (in divergence form) to diffuse the field based on the mobility M (controlling the degree of interface splitting; smaller values of M imply stronger splitting as less diffusion is introduced), and to steer the field towards a constant profile of width ξ of the form:

$$\phi(\mathbf{x}) = \frac{1}{2}\left[1 - \tanh\left(\frac{2d(\mathbf{x})}{\xi}\right)\right]\,, \quad (4)$$

where $d(\mathbf{x}) = \pm|\mathbf{x}-\mathbf{x}_0|$ is the signed distance of any point $\mathbf{x}$ to its *nearest* interface point $\mathbf{x}_0$ defined as $\phi(\mathbf{x}_0) = 1/2$. (Our formulation uses a signed distance function instead of the unsigned one in [17], as it will make our formulation simpler later on.) From this phase field, the exact geometry of the interface for rendering purposes can be extracted as an isosurface of ϕ, with an isovalue of (or around) 0.5, with its normal being $\mathbf{n} = \nabla\phi/\|\nabla\phi\|$ (pointing towards the high density phase given our convention for ϕ). The local fluid density is then derived from the phase field through Eq. (3). Finally, the force term $\mathbf{F}$ includes both body (gravity $\mathbf{g}$) and interface forces, where ν is the kinematic viscosity, while η and κ are determined by the surface tension σ and the interface width ξ as: $\eta = 12\sigma/\xi$ and $\kappa = 3\sigma\xi/2$. Because these six equations are highly coupled together, designing a numerical solver to simulate this macroscopic model in a stable and accurate manner is particularly challenging, even more so when efficiency is paramount.

3.2 Kinetic model

A way to bypass some of the difficulties in solving the macroscopic model described above is to adopt a kinetic model instead: using statistical mechanics, the macroscopic fluid equations can be rewritten as the evolution in time of the probability density encoding the presence of fluid volumes at a given position with a given velocity,

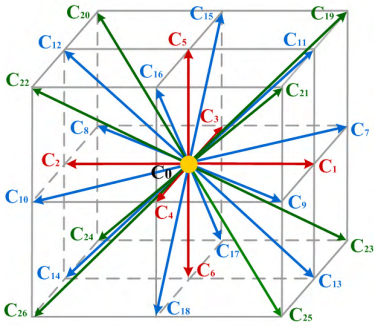

Fig. 2: **Velocity discretization.** The symmetric lattice structure D3Q27 used in 3D, where the 27 vectors $\mathbf{c}_i$ are used to discretize the local velocity. Note that every grid node stores a probability distribution g_i for each corresponding $\mathbf{c}_i$.

and incompressibility is approximated via a weakly-compressible flow. The resulting Boltzmann (transport) equation of this probability distribution is then simulated very efficiently using the lattice Boltzmann method (LBM), which amounts to move fictitious microscopic particles through consecutive steps of streaming and collision over a discrete lattice. The simplicity of the resulting localized update rules, tantamount to an automata, makes LBM highly parallelizable. The reader is invited to consult one of the many reviews of this approach such as, e.g., [90] for an introduction to kinetic models and their discretization, which have become quite standard in the computational physics and CG literature. We discuss very briefly the resulting equations and update rules in this section to define relevant notations and to prepare the grounds for the exposition of our contributions later on.

Solving fluid flow equations. After Eqs. (1a–c) are turned into their Boltzmann transport counterparts, a discretization of time (with time steps normalized to $\delta t = 1$) and of space and velocity directions (typically, using the nodes of a regular grid for positions and the symmetric D3Q27 structure in 3D for the velocity at each node as shown in Fig. 2) leads to equations of the form, for $i = 0 \dots 26$:

$$g_i(\mathbf{x} + \mathbf{c}_i, t+1) - g_i(\mathbf{x}, t) = \Omega_i^g(\rho, \mathbf{u}) + G_i(\rho, \mathbf{u}, \phi) \ , \quad (5)$$

where $\mathbf{x}$ is a given grid node, g_i is the probability distribution of particles moving with discrete velocity $\mathbf{c}_i$ at position $\mathbf{x}$ and time t (stored as a scalar at $\mathbf{x}$), and G_i is a force term:

$$G_i = (\mathbf{c}_i - \mathbf{u}) \cdot \left[\frac{\rho}{3}(\Gamma_i - \frac{\bar{w}_i}{3})\nabla\phi + \Gamma_i \mathbf{F} \right] , \quad (6)$$

with $\mathbf{F}$ defined in Eq. (2), $\delta\rho \equiv \rho_H - \rho_L$ represents the phase density difference, while $\bar{w}_i$ are fixed lattice (quadrature) weights and Γ_i is constructed from the BGK collision model as both described in [17]. Note that converting this statistical formulation back to the components of the macroscopic formulation is also feasible: to obtain the macroscopic momentum and pressure fields for instance, one can evaluate

$$\rho\mathbf{u} \ \equiv \ 3\sum_i \mathbf{c}_i g_i + \frac{\mathbf{F}}{2} \quad \text{and} \quad (7)$$

$$p \ \equiv \ \sum_i g_i + \frac{1}{6}\delta\rho \ \mathbf{u} \cdot \nabla\phi \ . \quad (8)$$

Due to the simple form of Eqs. (5), they can be handled particularly efficiently through operator splitting, i.e., they are rewritten as two consecutive steps (called streaming and collision, respectively) as:

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

where the components of the distribution g are first updated to handle streaming (advection), from which collision and forcing terms are added to get updated values at the next time step. Note that the actual expression of the collision terms Ω_i^g is key to guaranteeing an accurate correspondence with Eqs. (1a–c), so many different formulations have been offered over the years in the LBM literature. In this paper, we employ the recent weighted multiple-relaxation time (WMRT) model proposed in [17] for Ω_i^g, as it has been proven to offer a good compromise between accuracy and efficiency. Its expression is:

$$\mathbf{\Omega}^g = -\mathbf{M}^{-1}\mathbf{S}_g\mathbf{M}(\mathbf{g} - \mathbf{g}^{\text{eq}}) \ , \quad (9)$$

where $\mathbf{\Omega}^g = [\Omega_0^g, ..., \Omega_{26}^g]^T$ is the vector containing all collision operators at each node, $\mathbf{M}$ is a constant (pre-computed) "moment projection" matrix, and $\mathbf{S}_g = [s_0^g, ..., s_{26}^g]^T$ is a constant and diagonal relaxation matrix containing predefined relaxation rates [17]. The equilibrium distribution $\mathbf{g}^{\text{eq}}$ is modeled as:

$$\mathbf{g}^{\text{eq}} = \tfrac{\rho}{3}\mathbf{\Gamma} + (p - \tfrac{\rho}{3})\bar{\mathbf{w}} - \tfrac{1}{2}\mathbf{G} \ , \quad (10)$$

with $\mathbf{\Gamma}$, $\bar{\mathbf{w}}$ and $\mathbf{G}$ being the vectors containing all Γ_i, $\bar{w}_i$ and G_i for each node. Note that some of the relaxation rates $\{s_i^g\}_i$ are related to the kinematic viscosity ν as:

$$s_i^g = \left(3\nu + \tfrac{1}{2}\right)^{-1} , \quad i \in \{4, 5, 6, 7, 8\} \ . \quad (11)$$

The other relaxation rates for $i > 8$ are usually fixed to 1, and we adopt the same practice. The specific form of $\mathbf{M}$ and more derivation details can be found in [17]. Note that the kinematic viscosity ν is used in Ω_i^g, which can easily be interpolated at every node based on ϕ as: $\nu = [(1-\phi)\nu_L^{-1} + \phi\nu_H^{-1}]^{-1}$ [91], where ν_L and ν_H are the kinematic viscosity coefficients for the two fluid phases.

Solving conservative phase-field equation. In recent LBM-PF approaches, Eq. (1d) is also converted into a similar form:

$$h_i(\mathbf{x} + \mathbf{c}_i, t+1) - h_i(\mathbf{x}, t) = \Omega_i^h(\phi) \ , \quad (12)$$

where h_i is the phase-field probability distribution; the collision operator Ω_i^h is derived in order to faithfully simulate Eq. (1d), which is modeled in [17] as a single-time-relaxation process, leading to the expression:

$$\Omega_i^h = -(h_i - h_i^{\text{eq}})/\tau_\phi \ , \quad (13)$$

where $\tau_\phi = 3M + 1/2$ is the phase-field relaxation rate related to the mobility parameter M, and h_i^{eq} is the i-th phase-field equilibrium distribution modeled as:

$$h_i^{\text{eq}} = \phi\Gamma_i + M\bar{w}_i\left[\frac{4}{\xi}\phi(\phi - 1)\right](\mathbf{c}_i \cdot \mathbf{n}) \ . \quad (14)$$

Eq. (12) can also be solved through operator splitting as a streaming step followed by a collision step through:

$$h_i^*(\mathbf{x}, t) = h_i(\mathbf{x} - \mathbf{c}_i, t),$$
$$h_i(\mathbf{x}, t+1) = h_i^*(\mathbf{x}, t) + \Omega_i^h.$$

The phase-field function ϕ is then deduced directly (e.g., for rendering purposes) at any time t for any node $\mathbf{x}$ by taking the zeroth moment of h_i at $\mathbf{x}$, i.e.,

$$\phi(\mathbf{x}, t) = \sum_i h_i(\mathbf{x}, t) \ . \quad (15)$$

While the resulting LBM-PF scheme is arguably the most popular numerical approaches for a kinetic treatment of multiphase flows, its conservative phase-field kinetic model is only stable for either a large density ratio at low Reynolds number, or a small density ratio at high Reynolds number, see Fig. 7. It remains very difficult to have stable and accurate simulations for flows with both large density ratios and high Reynolds numbers, which are particularly desirable in practice to produce realistic results, e.g., in multiphase flow simulations between water and air.

4 Our Improved LBM-PF Solver

As pointed earlier, the LBM-PF technique still suffers from important limitations in terms of stability and accuracy, despite being one of the most popular and fastest multiphase flow solvers. In this section, we introduce a few key changes to the overall solver to improve both its accuracy and stability for both large density ratios and high Reynolds numbers, without noticeably affecting its computational efficiency.

4.1 Importance of phase-field gradient and advection

One of the first issues that we found about the LBM-PF approach is its discretization of the phase gradient ($\nabla\phi$ in the kinetic model presented in the previous section), which too often results in a dramatic loss of stability or accuracy near the interface: using the traditional central difference scheme for approximating the gradient results in approximation errors heavily dependent on the actual direction of the gradient (with respect to the grid-aligned directions of its operator stencil) and of the local curvature of the interface. This lack of rotational-invariance introduces spurious oscillations near the interface and often generates, in turn, high velocities leading to blow-ups or visual artifacts. While level-set based numerical methods use least-oscillatory finite difference schemes ((W)ENO schemes [92], [93]) that have been designed to remove these numerical issues, such schemes cannot be used in LBM formulations for a number of reasons. First, the D3Q27 discretization (Fig. 2) or any of its variants clashes with the fundamentally coordinate-per-coordinate treatment of (W)ENO schemes. Second, rotational symmetry of the stencil used to approximate gradients has been found important to the overall stability and accuracy of the discrete phase field treatment as it handles local directions of the interface normal much more uniformly [94] — and again, the very nature of (W)ENO schemes clearly fail to be isotropic. Yet, designing LBM-specific discretization schemes for the gradient operator to improve accuracy and stability remains largely unattended.

A second related issue that our tests have identified as a significant obstacle to stability is the numerical issues that the phase field creates for flows with large density ratios or high Reynolds numbers. Not only the advection of the phase field in strong velocity fields is often introducing artifacts as discussed above, but the truncation errors in the evaluation of the equilibrium distribution h_i^{eq} to approximate Eq. (1d) and the use of a single relaxation rate also lead to spurious oscillations near interfaces. These spurious oscillations lead to even stronger variations in the velocity, which end up creating a positive feedback loop that often leads to blowups.

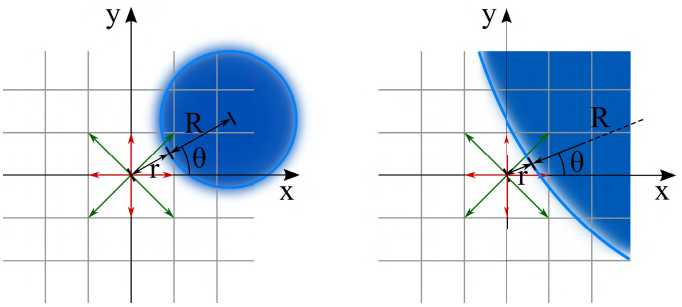

Fig. 3: **Local sphere interface model.** Our analysis of the discretization error uses a family of phase fields (shown in 2D here) representing spherical interfaces, parameterized by the radius R and the center of the sphere it encodes. By considering all possible configurations (from small droplets (left) to smooth surfaces (right)), an overall accuracy E is deduced via Eq. (19) for a given operator discretization.

In this section, we describe a variational, geometric approach to deriving gradient approximations and computing phase-field advection to drastically reduce these numerical shortcomings, and present further numerical changes to the collision models to render the LBM-PF technique much more robust.

4.2 Improving gradient estimates

In order to design symmetric stencils for which discretization errors of the gradient operator are small and as invariant as possible of the interface position, orientation and curvature, we follow a variational approach which optimizes stencil weights such that the gradient is well approximated for any local spherical approximation of the interface.

4.2.1 *Local sphere interface model*

Deriving error-minimizing discretizations requires defining first an error functional measuring the discretization error compared to an analytical ground-truth. Because we want the discretization error of our gradient operator to be as invariant as possible to the proximity, orientation, and curvature of the interface, we restrict our analysis to spherical approximations of the interface shape. We thus consider only phase fields whose ½-isosurface encodes a sphere of arbitrary center $\mathbf{x}_0$ and arbitrary radius R. Since time integration of the phase field is designed to maintain its profile to be of the form of Eq. (4), we can parameterize all local spherical phase fields as:

$$\phi(\mathbf{x};\mathbf{x}_0,R)=\frac{1}{2}\left[1-\tanh\left(\frac{\|\mathbf{x}-\mathbf{x}_0\|-R}{\xi/2}\right)\right], \qquad (16)$$

as it covers all positions and orientations of a spherical interface seen from $\mathbf{x}=\mathbf{0}$. The gradient of such a template phase field at $\mathbf{x}=\mathbf{0}$ is analytically expressed as:

$$\nabla\phi(\mathbf{0};\mathbf{x}_0,R)=\frac{\mathbf{x}_0}{\xi\|\mathbf{x}_0\|}\operatorname{sech}^2\left(\frac{\|\mathbf{x}_0\|-R}{\xi/2}\right). \qquad (17)$$

See Fig. 3 for a 2D illustration of the resulting diffuse interface, where the high-density side of the interface is colored in blue. If we denote by $\nabla^d\phi$ the discretization of

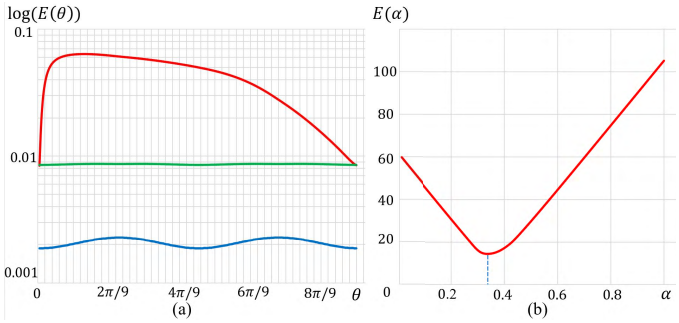

Fig. 4: **Variational symmetric discretization.** (a) Polar angle discretization error E_θ of the gradient operator for a centered difference evaluation (red), vs. the low-order symmetric variant [94] of Eq. (20) (green) and our optimal $\nabla^d_{1/3}$ from Eq. (22) (blue); (b) Error distribution $E(\alpha)$ when varying the weight α, showing a clear global minimum.

$\nabla\phi$ by a discrete stencil on the phase field grid values, the ℓ^2 approximation error can now be defined as:

$$e(\mathbf{x}_0, R) = \left\| \nabla^d\phi(\mathbf{0}; \mathbf{x}_0, R) - \nabla\phi(\mathbf{0}; \mathbf{x}_0, R) \right\|_2^2 . \tag{18}$$

To account for all local spherical approximations, we further take the integral of this error for a whole range of $\mathbf{x}_0$ and R, and define the total error functional E as:

$$E = \iiiint_{(\mathbf{x}_0,R)\in\mathcal{D}} e(\mathbf{x}_0, R) \frac{\exp(-\frac{1}{2}(\frac{R}{\xi})^2)}{\sqrt{2\pi}\xi} \mathrm{d}\mathbf{x}_0 \mathrm{d}R \, , \tag{19}$$

where we weighted the error with a centered Gaussian function of standard deviation ξ to emphasize errors for small droplets (small radius R) since they are more likely to create large errors and induce instability in our computations. We limit the integration to a domain $\mathcal{D}$ by using only radii from 0 to $R_{\max} \equiv 20$ and for $\mathbf{x}_0$ within a ball of radius R; the resulting error is then easy to evaluate through Gauss quadrature for any given choice of discrete operator ∇^d in order to provide a quantitative estimate of its accuracy within the family of phase fields we consider.

4.2.2 Variational symmetric discretizations

Recall that we seek a discrete gradient operator whose accuracy depends minimally on the position, orientation and curvature of the interface. A particular form of the discretization suggested in [94] and [17] is to evaluate the gradient by taking the zeroth moment of the directional derivatives along the 27 directions (for D3Q27) of the velocity to properly leverage all the directions instead of sticking to the usual centered difference in each coordinate. With this recommended approach, the discrete gradient is thus of the following second-order accurate form:

$$\nabla^d\phi = \sum_i \bar{w}_i \mathbf{c}_i \phi(\mathbf{x} + \mathbf{c}_i), \tag{20}$$

where $\bar{w}_i$ are the quadrature weights already used in Eqs. 6 and 14, and due to symmetry of the directions, the terms in $\phi(\mathbf{x})$ of the directional derivatives all cancel out. Since we have defined a parameterized family of phase fields above, we can compare how much better this form behaves compared to a regular centered difference typically used in finite-difference methods. Fig. 4 shows a comparison of the two discretizations, where we show the numerical error E_θ as a function of the spherical coordinate θ of the center $\mathbf{x}_0$; i.e., we compute the integral mentioned in Eq. (19) over a restricted domain $\mathcal{D}_\theta$ corresponding to the intersection of $\mathcal{D}$ with the set of all centers $\mathbf{x}_0$ of spherical coordinate θ, and $E = \int_{2\pi} E_\theta d\theta$. Using the rotationally-symmetric discretization of Eq. (20) (green curve), the error variation is already significantly reduced compared to that from the second order centered difference (red curve) — and the total error E is thus over an order of magnitude improved.

High-order approximations. However, the error magnitude is still of the same (second) order, which is not accurate enough to capture subtle interfacial effects in a stable manner: we need *higher order* symmetric discretizations. Fortunately, we can easily increase the order of the gradient approximation while *keeping the same 27 directions* by simply using values of the phase field further away in these directions; that is, we simply use a a second-order approximation of the directional derivative along each $\mathbf{c}_i$. Due to the cancellation of both the second-order terms and of $\phi(\mathbf{x})$ along symmetric directions, we obtain a third-order gradient approximation, denoted $\nabla^{d,3}$, of the form:

$$\nabla^{d,3}\phi = \sum_i \bar{w}_i \mathbf{c}_i \frac{4\phi(\mathbf{x} + \mathbf{c}_i) - \phi(\mathbf{x} + 2\mathbf{c}_i)}{2}. \tag{21}$$

Weighted approximations. Note that higher-order approximations are, however, not desirable everywhere: near rapid changes of ϕ (i.e., close to the interface), higher-order stencils can in fact hurt accuracy as they generate spurious oscillations (Gibbs phenomenon). Instead, one would ideally want an accurate operator in smooth regions of ϕ, while switching to a low-order stencil near fast value changes. Following the idea behind the weighted variant of ENO [93], we achieve this goal by linearly combining Eqs. (20-21):

$$\nabla^d_\alpha\phi = (1-\alpha)\nabla^d\phi + \alpha\nabla^{d,3}\phi, \tag{22}$$

where the parameter α is in $[0, 1]$. In order to obtain an optimal value for α without having to estimate the local behavior of ϕ, we assume it constant and employ the previously developed variational approach by replacing $\nabla^d\phi$ in Eq. (19) with $\nabla^d_\alpha\phi$ from Eq. (22), and minimize the total error from Eq. (19) w.r.t. α. Fortunately, the graph of the error as a function of α is remarkably simple and convex, with a clear minimum for $\alpha \approx 1/3$ as shown in Fig. 4(b). We thus use $\nabla^d_{1/3}$ to safely and accurately estimate the gradient of the phase field anywhere in the domain. Fig. 4(a) demonstrates that our new weighted discretization (blue curve) leads to significantly higher accuracy (an order of magnitude improvement in this case), while maintaining the orientation bias to a minimum as evidenced by the small oscillation amplitude of the error curve. Note finally that higher-order weighted gradient operators could be similarly derived, but we found this second- to third-order accurate version to be sufficient to vastly improve stability and accuracy without adding undue computational complexity as we will demonstrate in Section 6.

4.2.3 Hyperbolic weighted discretizations

While Eq. (22) can be applied anywhere a phase-field gradient estimate is needed, one term requires special attention: the advection of the phase field, which is a typical numerical issue in hyperbolic problems and requires its own hybrid

expression for enhanced accuracy and stability. While the phase field advection term is often performed using the macroscopic velocity $\mathbf{u}$ through $\mathbf{u}\cdot\nabla\phi$ in the pressure evaluation (8), using the first moment of g_i at a node in combination with a centered stencil for the gradient of ϕ is known to lead to the traditional Gibbs phenomenon near the interface compared to flux-based upwind approximations, in particular when the density ratio is large. Therefore, we avoid the macroscopic velocity altogether and propose evaluating the hyperbolic part of our collision model directly at the level of the probability distribution (i.e., at what is sometimes called the mesoscopic scale) through upwinding to gain on both accuracy and stability.

Mesoscopic pressure terms. If we substitute the definition of the macroscopic velocity Eq. (7) into the pressure equation (8), we obtain

$$p = \underbrace{\sum_i g_i}_{p^{(1)}} + \underbrace{\frac{\delta\rho}{2\rho}\sum_i \mathbf{c}_i\cdot\nabla\phi\, g_i}_{p^{(2)}} + \underbrace{\frac{\delta\rho}{12\rho}\mathbf{F}_{sb}\cdot\nabla\phi}_{p^{(3)}}, \tag{23}$$

where we separated the components of the pressure to discuss their treatment individually. The first term involves no differential and can be evaluated straightforwardly. Similarly, the $p^{(3)}$ component involving the forcing term $\mathbf{F}$ can be evaluated using the gradient estimate from Eq. (22). It is the second component, involving both the probability distribution g_i and the velocity direction $\mathbf{c}_i$, which requires a finer treatment: indeed, the term $g_i\,\mathbf{c}_i\cdot\nabla\phi$ can be far from symmetric depending on the values g_i (whose first moment encodes the phase field ϕ). So we must now include terms that used to cancel out in the previous gradient discretization, and we should also use upwinding to provide a better numerical treatment of this advective term.

Upwinding. Using the same derivation as before with now restoring the term in $\phi(\mathbf{x})$ as it may no longer cancel out (depending on the values of g_i) , we can get a low-order *upwind* expression for $p^{(2)}$ (that we call $p^{(2),a}$) of the form:

$$p^{(2),a} = \sum_i g_i\,\big(\phi(\mathbf{x}) - \phi(\mathbf{x}-\mathbf{c}_i)\big), \tag{24}$$

where the use of $\phi(\mathbf{x}-\mathbf{c}_i)$ instead of $\phi(\mathbf{x}+\mathbf{c}_i)$ reflects the typical approach to upwinding. As in the previous case, we can also derive a higher-order approximation by simply using a larger stencil of values of ϕ while still keeping the same 27 directions only, leading to a second approximation we denote $p^{(2),b}$ written as:

$$p^{(2),b} = \sum_i g_i\,\big(3\phi(\mathbf{x}) - 4\phi(\mathbf{x}-\mathbf{c}_i) + \phi(\mathbf{x}-2\mathbf{c}_i)\big). \tag{25}$$

As we have discussed in Section 4.2.2, higher order schemes tend to be oscillatory around the interface, creating instability at large density ratio and/or with high Reynolds number. We thus combine low-order and high-order templates again, similar to Eq. (22), with a constant parameter β, which gives us the following new form for $p^{(2)}$ as:

$$p^{(2)}_\beta = (1-\beta)\,p^{(2),a} + \beta\,p^{(2),b}\ . \tag{26}$$

To determine β, we employ the same numerical procedure described in Section 4.2.2 to minimize a similar error functional based on our local sphere interface model, integrated over different interface distributions; the only difference is that we now sum all the errors of $\mathbf{c}_i\cdot\nabla\phi$ vs. the closed-form ground truth, instead of just $\nabla\phi$. We found this time that the optimal parameter value is $\beta\approx 2/3$, with an error curve very similar to the one depicted in Fig. 4(b). One could use even higher order stencils to further improve accuracy, but this discretization offers the best balance between locality and accuracy. The final numerical "upwind" approximation p_u of the pressure p is then assembled as:

$$p_u = p^{(1)} + p^{(2)}_{2/3} + p^{(3)}, \tag{27}$$

using $p^{(2)}_{2/3}$ defined via the expression of $p^{(2)}_\beta$ for $\beta = 2/3$.

4.2.4 Adaptive estimation of pressure

While the mesoscopic approach to the hyperbolic part in the pressure evaluation provides much improved accuracy and stability on or close to the interface, it is not quite appropriate when dealing with boundary conditions that are typically enforced macroscopically: we cannot really take into account constraints on the macroscopic velocity $\mathbf{u}$ as we directly manipulate the probability distributions. It is better, instead, to rely on a simpler approximation p_s where the pressure is directly evaluated using a symmetric approximation of the gradient of ϕ as:

$$p_s = \sum_i g_i + \frac{\delta\rho}{6}\mathbf{u}\cdot\nabla^d_{1/3}\phi. \tag{28}$$

As a simple toggle between the approximants provided in Eq. (28) near boundaries and in Eq. (26) near interfaces can create adverse effects, we prefer to blend these two numerical schemes based on the magnitude of $\nabla^d_{1/3}\phi$, since a large magnitude of the phase field indicates proximity of an interface. Our final expression for the pressure is thus:

$$p = \max\!\left(1 - \frac{\|\nabla^d_{1/3}\phi\|}{\sqrt{3}/5}, 0\right) p_s + \min\!\left(\frac{\|\nabla^d_{1/3}\phi\|}{\sqrt{3}/5}, 1\right) p_u\ , \tag{29}$$

where the normalization term $\sqrt{3}/5$ is the maximum gradient norm of $\|\nabla^d\phi\|$ from the analytical profile of ϕ in Eq. (4), and we threshold any exceeding value due to numerical inaccuracy. This simple blend now offers a stable and accurate approach for both smooth regions where macroscopic boundary conditions may be imposed on and near interfaces.

4.3 Improving collision models

A large amount of works in lattice Boltzmann methods have been dedicated to developing good collision models. While initial methods use the lattice Bhatnagar–Gross–Krook model (a single-relaxation-time collision model), recent developments have popularized the use of multiple-relaxation-time (MRT) collision models as they offer finer control over the probability distributions. The use of a mesoscopic treatment for the phase-field equation (Eq. 1d) is relatively recent, and the current state-of-the-art LBM-PF approach [17] advocates a single relaxation time for the phase field as reviewed in Eq. (13) due to its simplicity. Surprisingly, our tests showed that keeping such a simplistic model for the collision term is actually harmful numerically *even when our improved gradient estimates are used*, as it contributes to the current stability restrictions of LBM-PF. Additionally, very turbulent flows are not handled

well for a given grid size as gradient estimates may become inaccurate due to a poorly-resolved vector field, so the use of an artificial viscosity in the relaxation process to account for subgrid flow contributions and to limit velocity magnitudes can dramatically help increase stability while capturing even finer turbulence details. We detail our changes to the collision models of both the mesoscopic simulation of the velocity field (through g_i) and of the phase field (through h_i) in this section.

4.3.1 WMRT model for phase-field collision

One of the main reasons for the instability of the single-relaxation-time model of Ω_i^h is that it lumps all the relaxation modes together, instead of having control over which modes to target. As motivated by the weighted multiple-relaxation-time (WMRT) model which is only applied to Ω_i^g in [17], we adopt a WMRT treatment for the phase field as well: we also project h_i and h_i^{eq} onto orthogonal moment spaces by the same precomputed matrix $\mathbf{M}$ used in Eq. (9), before scaling these moments through a constant, diagonal matrix $\mathbf{S}_h$, and reconverting the result to the collision vector $\mathbf{\Omega}^h$ through:

$$\mathbf{\Omega}^h = -\mathbf{M}^{-1}\mathbf{S}_h\mathbf{M}(\mathbf{h}-\mathbf{h}^{\text{eq}}) \;, \tag{30}$$

where $\mathbf{\Omega}^h$, $\mathbf{h}$ and $\mathbf{h}^{\text{eq}}$ are vectors containing Ω_i, h_i and h_i^{eq} at each grid node for $i = 0..26$. Like in $\mathbf{S}_g$, the first relaxation rate s_0^h of $\mathbf{S}_h$ is set to zero since we do not want to affect the first moment of h_i (which is the phase field, see Eq. (15)); the next three relaxation rates in $\mathbf{S}_h$ are related to the mobility M used in Eq. (13) through:

$$s_i^h = \left(3M+\frac{1}{2}\right)^{-1} \quad \forall i \in \{1,2,3\}. \tag{31}$$

Other higher-order relaxation rates for $i>3$ are all set to 1.5 to exert a minute amount of damping on these high-order modes in order to reduce the high-frequency oscillations due to dispersion error. Using this simple WMRT model for collision (which, given our choice of relaxation rates, amounts to a two-relaxation-time model) removes most of the issues described earlier; most notably, the truncation errors in the evaluation of the equilibrium distribution are no longer creating visible artifacts.

4.3.2 Resolving sub-grid-scale flow details

As mentioned before, turbulent flow simulations may suffer from undersampling of the velocity field in a fixed grid resolution. Various subgrid models have been proposed to palliate numerical errors over coarse grids. We propose to employ the wall-adapted large-eddy (WALE) turbulence model [95] to predict an eddy viscosity ν' at each grid node, which is then added to the global kinematic viscosity ν to form the total spatial-varying viscosity $\nu+\nu'$. This total viscosity is then used in Eq. (31) in lieu of ν, thus modifying the collision term for g_i. This simple change, already proposed in single-phase simulations but never in multiphase simulations to our knowledge, adds relatively large eddy viscosity in highly turbulent regions to dampen the typical velocity dispersion errors witnessed in these under-resolved regions, while capturing slightly improved details in less turbulent regions. Note also that this change on the handling of the probability distribution g_i does impact the accuracy and stability of our phase field treatment, since phase-field advection becomes less prone to numerical errors as well.

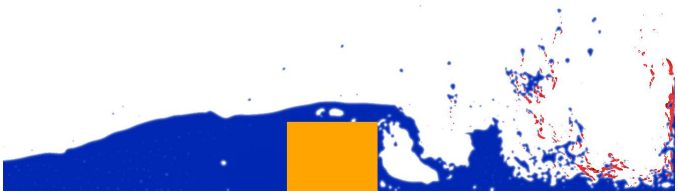

Fig. 5: **Velocity-limiting adaptive viscosity I.** In this turbulent two-phase example (liquid in blue), regions where our velocity-limiting adaptive viscosity is turned on to prevent potential blowups are highlighted in red, demonstrating its spatial sparsity and proving that its impact on the overall flow structure is typically negligible.

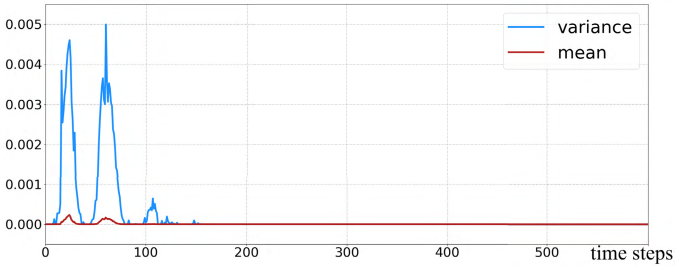

Fig. 6: **Velocity-limiting adaptive viscosity II.** Variations in time of the spatial average value our velocity-limiting adaptive viscosity (red) and of its variance (blue) for the 2D simulation in Fig. 5, confirming its temporal sparsity.

4.3.3 Velocity-limiting adaptive viscosity

Even though the above fixes can further reduce simulation error effectively and enhance stability, if the density ratio is large enough for high Reynolds number (for example, a density ratio of 500 with a Reynolds number around 10^4), local error of $\nabla\phi$ near interfaces can quickly accelerate the local velocity to the point of exceeding the stability range of the LBM time updates. To maintain stability, we further propose to incorporate another dissipation acting, this time, as the equivalent of a slope limiter in typical high-resolution schemes to cutoff spurious high velocities near interfaces without affecting the overall flow. Such a filtering of the velocity is achieved by further adding an artificial viscosity ν'' to ν in Eq. 11, with this limiting viscosity defined as:

$$\nu'' = \omega\frac{|\mathbf{S}|(\max\{0,|\mathbf{u}|-\lambda\})^2}{d+\epsilon} \;, \tag{32}$$

where λ is a threshold in the range $[0.1, 0.15]$ defining what is considered as high velocity to make sure only very high velocity regions are dampened; ω controls the amount of artificial viscosity, which we fixed to 1000; $|\mathbf{S}|$ is the norm of the strain rate computed as $\mathbf{S} = (\nabla\mathbf{u}+\nabla\mathbf{u}^T)/2$; d is the distance of a node to its nearest interface, which we deduce from the phase-field profile given in Eq. (4); and ϵ is set to 10^{-12} to avoid divisions by zero. The physical meaning behind such a limiting process is thus simple: high velocities with large strain rates near interfaces are quickly suppressed to effectively stabilize the simulation. Fig. 5 illustrates on a simple 2D dam break example the regions where such a filtering is applied, while Fig. 6 plots

Algorithm 1 Pseudo-code of our kinetic solver.

1: Initialize density ρ and velocity $\mathbf{u}$ and ϕ ;
2: Initialize g_i and h_i with their equilibrium states through Eq. (10) and Eq. (14), respectively;
3: **while** iteration $\leq$ max iteration **do**
4: Perform streaming steps for g_i and h_i via Eq. (5) and Eq. (12), respectively;
5: Apply non-slip boundary conditions for g_i and h_i using standard bounce-back rule;
6: Compute ϕ using Eq. (15) and obtain ρ by Eq. (3), where solid wetting boundary condition Eq. (33) is applied for contact angles based on Eq. (34);
7: Compute $\mathbf{F}$ via Eq. (2) w/ $\nabla\phi$ evaluated thru Eq. (22);
8: Compute $\mathbf{u}$ using Eq. (7);
9: Compute pressure p through Eq. (29) where p_s and p_u are calculated by Eq. (28) and Eq. (27), respectively;
10: Perform colliding steps of fluid flow and phase-field equations via Eq. (9) and Eq. (30), respectively.
11: **end while**

the variations in time of the mean artificial viscosity ν'' and its variance, reinforcing that the impact on the overall flow is very limited, and accuracy can be preserved while making the system stable by acting just when and where it is needed to prevent blowups.

4.4 Discussion

In this section, we proposed several modifications of the basic computational framework of [17] to improve both stability and accuracy of the LBM-PF technique. As a summary, we give pseudocode in Alg. 1 of our new kinetic multiphase flow simulation. In particular, we substantially improved the way $\nabla\phi$ is approximated (for both force evaluation and advection), and enhanced the two collision processes. While the former brings significant accuracy improvements, both of these contributions partake in guaranteeing much more stable simulations for turbulent flows with large density ratios: while we postpone describing our results until Section 6, Fig. 7 (plotting the density ratios and Rayleigh numbers for which the 2D simulation shown in Fig. 5 does not blow up) clearly shows that our solver significantly outperforms [17] for multiphase flows in terms of stability.

5 Implementation Details

We now discuss specific aspects of our implementation, both to ease reproducibility and comment on a few choices we made in the process of implementing our approach.

Setup. Aside from the differences we described in Section 4, our approach uses the typical LBM-PF setup of [17]. In particular, our solver employs *normalized units*, where the grid spacing δx and the time step size δt are both assumed to be unit; as a consequence, the linear advection in the Boltzmann equation (with constant lattice velocities $\mathbf{c}_i$) corresponds to a unit CFL number as well, independent of the Reynolds number. Similarly, the highest phase density is always set to $\rho_H = 1$, so we set $\rho_L = \rho_H/\gamma$, where γ represents the density ratio of the multiphase flow we wish

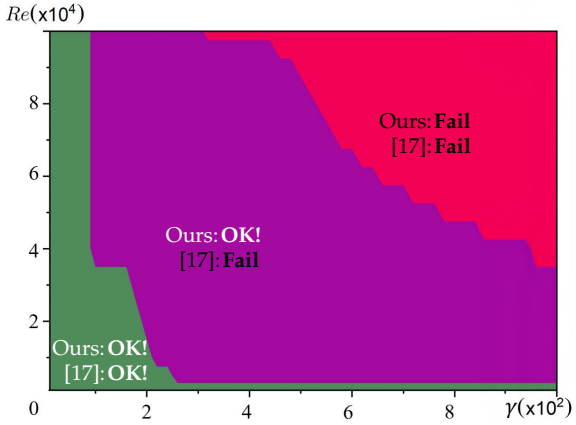

Fig. 7: **Stability range.** For the 2D animation of Fig. 5, we tried both our solver and the original one from [17] for a variety of density ratios $\gamma = \rho_H/\rho_L$ and Reynolds numbers Re. The green region shows the stability range for their solver (i.e., simulations that did not end up blowing up), while the purple region is for our solver. The red region represents values that systematically result in blowups for both methods. Our solver has a stability range over 5 times larger than the original LBM-PF it started from, with essentially the same computational complexity.

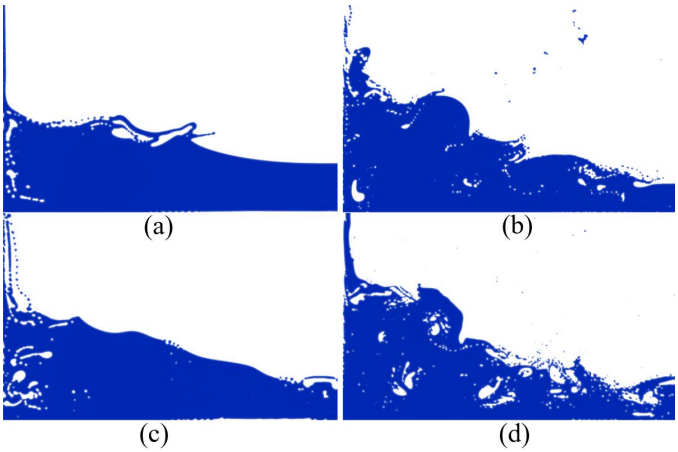

Fig. 8: **Dam-break improvements.** We show the effect of the various contributions to [17] that we propose in this paper on a 2D dam break example for a density ratio of 800: (a) result from [17]; (b) result when proper gradient discretizations are used; (c) result when new relaxation is used. (d) result when combining both improvements. The same frame of the simulation is shown in each case.

to simulate. Our implementation always uses an interface thickness ξ of 5, but this value can be adapted to the actual grid resolution used for simulation. All other parameters are properly scaled accordingly to ensure that the LBM-PF results correspond to the correct physical parameters. For instance, gravity is typically set to $g = 10^{-5}$, corresponding to a normal gravity on earth (but other values can of course be used to simulate weaker or stronger gravity), while surface tension for water corresponds to $\sigma = 10^{-6}$, but again, other values can be used to simulate different Weber numbers. We also used a mobility M set to 0.2 in all examples, but a mobility (propensity to splash) in a range

Figures	Resolution	Time/Frame	ν_H	ν_L	δt ($\times 10^{-5}$s)
Fig. 1	800×400×400	3.5 min.	0.002	0.02	12.5
Fig. 15 (a)	640×360×640	2.7 min.	0.04	0.8	8.3
Fig. 15 (b)	640×360×640	2.7 min.	0.01	0.8	8.3
Fig. 15 (c)	640×360×640	3.0 min.	0.01	0.8	8.3
Fig. 15 (d)	640×360×640	3.0 min.	0.002	0.8	8.3
Fig. 15 (e)	640×360×640	3.0 min.	0.0006	0.8	8.3
Fig. 9 (a,b)	800×400×400	3.5 min.	0.001	0.1	12.5
Fig. 9 (c,d)	640×360×640	3.0 min.	0.003	0.5	12.5
Fig. 10 (a,b)	700×140×700	3.0 min.	0.002	0.08	12.5
Fig. 10 (c,d)	800×400×400	3.5 min.	0.005	0.5	12.5
Fig. 11 (a,b)	300×300×300	50 sec.	0.01	0.1	10.3
Fig. 11 (c)	300×300×300	50 sec.	0.01	0.1	10.3
Fig. 12 (a,b)	400×600×400	3.25 min.	0.001	0.01	3.3
Fig. 13 (a,b,c)	400×600×400	3.25 min.	0.002	0.05	2.2
Fig. 14 (a,b)	860×200×860	5.5 min.	0.005	0.0005	12.5
Fig. 16 (c)	300×600×300	2.5 min.	0.002	0.02	10.3

TABLE 1: **Statistics.** We provide parameters and timings of our results; ν_H and ν_L are liquid and air viscosities, which depend on the Reynolds number of their associated simulation; δt indicates the time step size in physical unit (second), and δx (grid size) in physical unit is 0.01 meters; the surface tension σ is always set to 10^{-6} except for Fig. 11(a,b) (where σ= 0.03) & Fig. 11(c) (where σ= 0.03/0.0005) for illustration purposes. We used 80 frames per second for all animations sequences, except for the glugging example (Fig. 13, 200fps for better visualization) and the surface tension examples (Fig. 11, 30fps due to motion simplicity).

of $[0.1, 0.2]$ also results in stable simulations. Viscosities for different phases (ν_L and ν_H) are fixed depending on the Reynolds number, characteristic velocity and domain size. Table 1 shows the parameters (and resolutions) used for all the simulations shown in this paper.

Soft-start initialization. Before starting a simulation, we need to initialize the distribution functions g_i and h_i. We use $g_i = g_i^{\text{eq}}$ (Eq. 10) with an initial density field ρ and velocity field $\mathbf{u}$, and $h_i = h_i^{\text{eq}}$ (Eq. 14) with an initial placement of the interface ϕ with the ideal profile. Since g_i^{eq} requires a usually unknown pressure p, it is often initialized as $p = 3\rho$ (based on an ideal gas equation of state), and as time proceeds, p will quickly converge to the correct field within a few time steps. However, this approach generates an artificial transient compression wave, similar to what is seen in many compressible SPH methods. Such a "hard-start" is a critical issue when large forces (e.g., large surface tension or body forces) are used, as it can quickly generate very large velocities and numerical instabilities. Therefore, we prefer a "soft-start" by simply turning off all forces and making all boundary conditions open for the computational domain: compression waves will then be quickly dampened out, and we re-activate all forces and boundary conditions after just a dozen of time steps. This simple procedure has been verified by our experiments to be a very effective initialization to support stable simulations.

Boundary treatment. It is also important that different boundary conditions are properly set for different simulation scenarios. For domain boundaries and for inlets and outlets, we apply the standard treatment of [96]. For object boundaries, the traditional bounce-back treatment of [97] is applied to both h_i and g_i to ensure no-slip condition. Note that the evaluation of $\nabla\phi$ should also be properly treated at the solid boundary, especially to capture the wetting phenomena where different contact angles of liquids may form at equilibrium on solid surfaces. Following [98], the wetting condition is modeled using:

$$\mathbf{n}_w \cdot \nabla\phi(\mathbf{x}_w) = \Theta\, \phi_w\, (1 - \phi_w)\ , \tag{33}$$

where $\mathbf{n}_w$ is the surface normal at a point $\mathbf{x}_w$ of the solid boundary, which may not be located at the fluid grid; ϕ_w is the phase-field value at the solid surface; and Θ is related to the equilibrium contact angle θ through:

$$\Theta = -\sqrt{2\eta/\kappa}\cos\theta. \tag{34}$$

This wetting boundary condition is then enforced using the discretization method proposed in [17].

Viscosity selection for different phases. In all our examples, we always set ν_H and ν_L to be the viscosities corresponding to water and air. In the general case, viscosities for different phases (ν_H and ν_L) should be determined based on the desired Reynolds number. However, in practice, the grid resolution may not be sufficient to resolve the thin boundary layer that typically forms between two phases. This may create visual artifacts; for instance, small droplets may be more strongly influenced by the surrounding turbulent air flow than they should. In order to overcome such a problem, it is common practice to increase the viscosity ratio between different phases.

Hardware setup and timings. Since our multiphase fluid solver is embarrassingly parallel, we implemented it with CUDA on a multi-GPU system where four NVIDIA P40 GPUs are installed, each with 22 GB of memory. Capturing the fine details of multiphase flows typically requires regular grids of resolution above $400 \times 400 \times 400$, thus requiring over 32GB of GPU memory — note that lower resolution can be used if necessary, but the treatment of the coupling between the two phases through the interface will only be predictive for high-enough resolutions. Depending on the Reynolds number, one (visual) frame of animation is generated after 50 to 200 time steps, which correspond to around 1.8 mins of computations for 100 times steps (or 3.5 mins at $800 \times 400 \times 400$). Note that the 2D simulations we show in this paper were performed using the D2Q9 lattice structure, and typically require around 6 seconds per frame of animation on the same hardware. Compared to existing CFD tools for multiphase flows like Gerris [99], our 3D results are significantly faster; for instance, our glugging example is achieved almost 10,000 times faster when Gerris is run on a 20-core CPU. Compared to other CG approaches (which can only reproduce specific multiphase flow behaviors), we are also very competitive: based on the timings given in [7], our glugging example is around 3 orders of magnitude faster per frame —but this acceleration is likely to be less large due to hardware improvements over the last six years; note, however, that their use of a (global) Poisson solver will not make their approach scale as well as ours for larger grids. Even compared to adaptive methods that use particles only near interfaces to speed up computations such as in [8], we are only 30% slower, see Section 6.2 for a finer analysis of this case.

Rendering. Finally, our results were rendered with Mitsuba [100], taking around 20 minutes per frame on a Linux server with a 2.6GHz 28-core CPU.

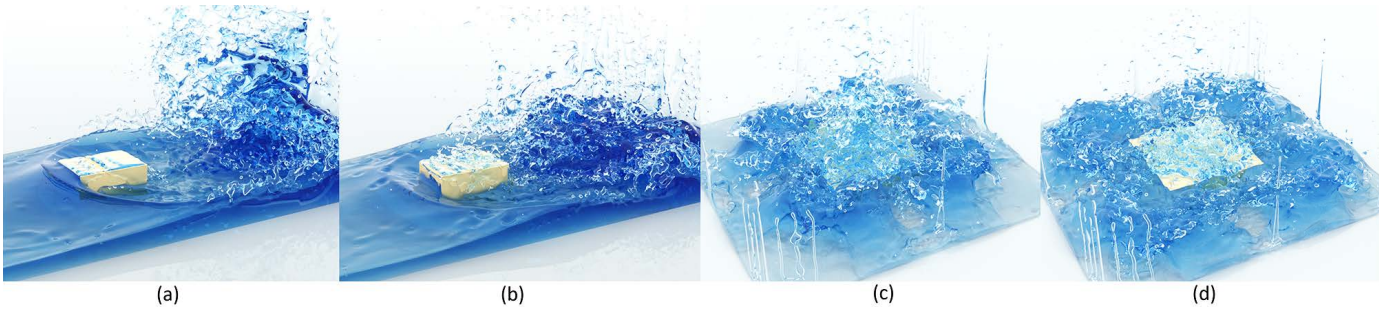

Fig. 9: **Splashing.** Simulations of water splashing in different scenarios; note that splashing not only depends on parameter M, but also on the degree of turbulence around the interface. Wetting on obstacle and walls also appears in both cases.

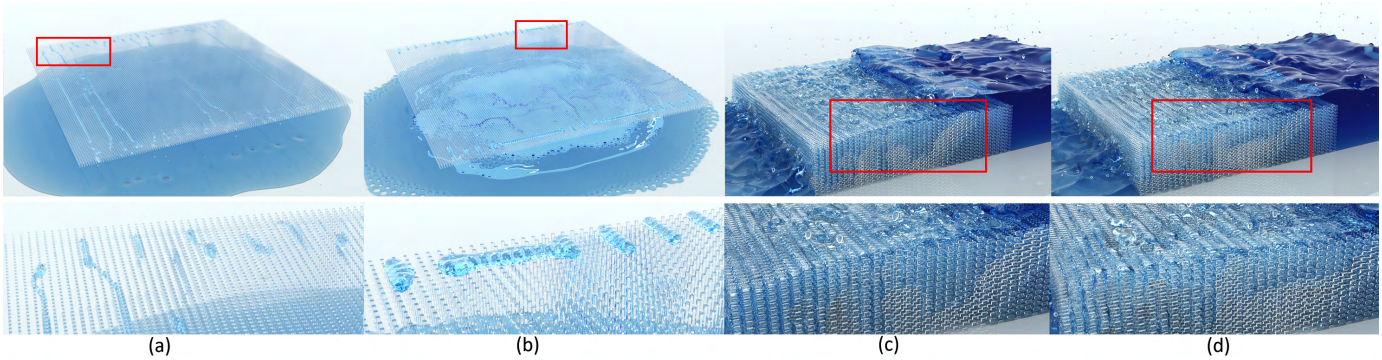

Fig. 10: **Wetting.** Simulations exhibiting strong wetting phenomena. (a) & (b): liquid flows over a thin-plate porous solid with hydrophilic ($\theta = 40$) and hydrophobic ($\theta = 160$) wetting, respectively; (c) & (d): wetting on volumetric porous solid with the same hydrophilic and hydrophobic wetting properties as in (a) & (b).

6 Results and Discussions

In this section, we describe some of our results in order to demonstrate the variety of visual effects that our unified solver can generate, and also provide comparisons to existing multiphase flow solvers.

6.1 Multiphase flow behaviors

In order to demonstrate the range of complex behaviors that our multiphase solver can tackle, our results cover a variety of different scenarios. Since many multiphase flow phenomena can be captured simultaneously by our unified solver, one particular example may exhibit more than just the targeted visual behavior.

Splashing. An important (and very recognizable) phenomenon in liquid flows is splashing. Fig. 9 shows two examples produced by our solver. Different liquids may have different splashing behavior, which is easily controlled by the parameter M in our solver. (All our results were generated with $M = 0.2$.) Note that splashing also occurs due to the existence of vortices near interfaces, creating an imbalanced pressure on the interface and forming splashes of irregular shapes.

Wetting. When liquids hit solid surfaces, they naturally wet them, which can form very complex patterns. The degrees of wetting, as specified by the contact angle, is seldom considered by most liquid simulators in graphics; yet it is omnipresent in examples as mundane as pouring water out of a pitcher. Our solver can specify a variety of wetting conditions by setting different contact angles.

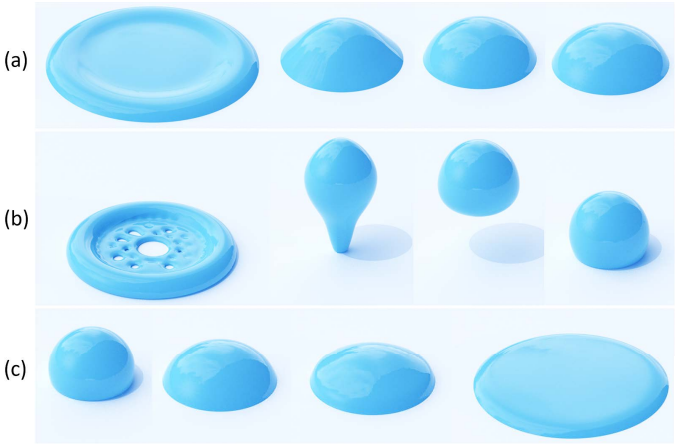

Fig. 11: **Surface tension.** Examples with different parameters for surface tension and wetting. Top: large surface tension ($\sigma = 0.03$) with hydrophilic wetting ($\theta = 40$); middle: same large surface tension with now hydrophobic wetting ($\theta = 160$); bottom: transition from the same large surface tension with hydrophobic wetting ($\theta = 40$) to a small surface tension ($\sigma = 0.0005$) with hydrophilic wetting ($\theta = 160$).

Moreover, we tried a scenario where wetting dominates the behavior of the liquid flow near solid surfaces (like in thin fabric or on solid porous materials) in Fig. 10(a–d), and our solver faithfully captured such a case. Note also that since liquids may often get in contact with domain boundaries, wetting is observable in many of our examples on domain walls — but it can easily be removed if desired by changing the boundary condition for $\nabla\phi$ such that discrete samples

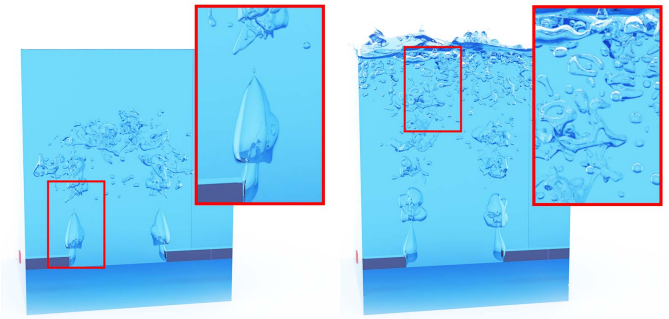

Fig. 12: **Bubbling.** Bubble generation, where two bottom air inlets are given sufficient pressure to push air into the liquid to form complex bubbles. Due to air pressure, the bubbles initially form mushroom-like shapes, see (a). Then, they gradually become unstable and start splitting, see (b).

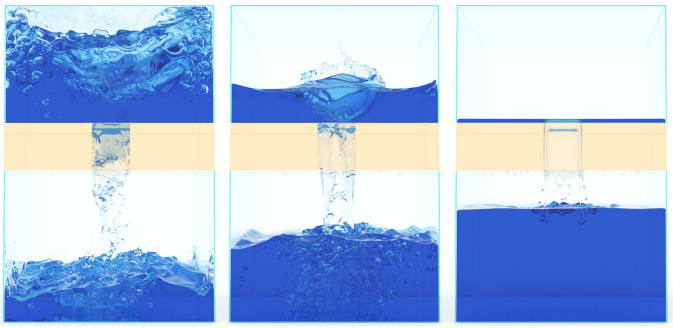

Fig. 13: **Glugging.** As water is flowing down from an upper container, (a) large bubbles first appear inside the top container, along with several small bubbles; (b) as the liquid accumulates below, many small bubbles are observed; (c) near the end of the process, wetting is obvious along the container walls.

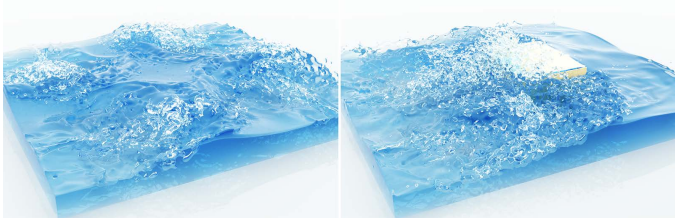

Fig. 14: **Air-driven flows.** Our approach can also simulate the motion of water purely driven by high air speed. Strong splashing and surface waves are generated by a high-speed flow of air over the liquid, along with bubbles and wetting.

of ϕ outside the domain are all zero.

Surface tension. In addition to splashing and wetting, liquids may also exhibit a diversity of surface tension induced phenomena. Liquids with large surface tension (thus creating forces that dominates the flow behavior) have already been demonstrated by many previous works, see for instance [101], [10]; we can reproduce similar cases in our framework, but we differ from these previous works in that different wetting conditions can *also* be set in these flows when they come in contact with solids, see Fig. 11. Note that this example shows that, as expected, large surface tension with hydrophobic wetting can make the liquid bounce up; and we can even change the surface tension and wetting conditions dynamically during the simulation (e.g., to simulate the addition of a detergent).

Bubbling. In many multi-phase flows, the generation of bubbles inside turbulent liquids is an important visual phenomena. Our solver can easily capture complex bubbles that are automatically triggered by the motion of the liquid. As a special example, we inject air from two inlets at the bottom of a water container, see Fig. 12. The air pressure will first form large bubbles, which will then rise and split into smaller ones. (This is different from typical bubble-rising simulations where bubbles are manually put in front of the inlets.) Due to air pressure, the initial bubbles at the inlets can form mushroom-like shapes that are further stretched in time, see Fig. 12(a). Then, they gradually become unstable and start automatically splitting, see Fig. 12(b). Again, all the behaviors witnessed in this example emerge from the simulation, and are not hand-directed in any way.

Glugging effect. A common multi-phase phenomenon where bubbles are automatically generated is glugging (also called gurgling), which has also been demonstrated on its own in previous works [8], [7]. Our solver can efficiently produce such a phenomenon, see Fig. 13 for instance. However, due to the unified nature of our solver targeting a whole spectrum of multiphase phenomena, our simulated glugging can exhibit a wide range of bubble sizes deep inside the liquid (see Fig. 13(a–b)), as well as wetting near solid boundaries (see Fig. 13(c)), which becomes very obvious near the end of the animation, but is completely ignored in existing simulators in graphics. We will demonstrate how our results are close to real experiments in the next section, which is a unique advantage of our method over existing solvers.

Air-driven liquid flow. An even more powerful aspect of our solver is its ability to simulate liquid flows which are purely driven by fast blowing air, see examples in Fig. 14. Air-driven liquid flow is another typical example of multiphase flows where air and liquid tightly interact with each other, and possibly in a very turbulent manner. This definitely requires high stability of the solver to support strong turbulence for both air and liquid with a high density ratio. To the best of our knowledge, this work is the first to be able to simulate air-driven water flows.

6.2 Comparisons

With CFD method. As we argued earlier, Fakhari et al. [17] is currently the state-of-the-art approach within existing LBM-PF techniques, and our contributions are enhancing, at times drastically, their results. Fig. 15 compares the simulation results of [17] and ours, without and with artificial velocity-limiting diffusion. For fair comparison, we set the density ratio to 500 as done in [17], and pick the largest Reynolds number that their method can handle. For this choice of parameters, both results are visually similar as the same crown of droplets appear, see Figs. 15(c) & (e). However, our approach can handle much higher Reynolds numbers due to the stability that our changes have brought, see Figs. 15(b) & (d). In fact, to visualize how large a region of stability we can achieve, we took the animation example of Fig. 5 and tried it

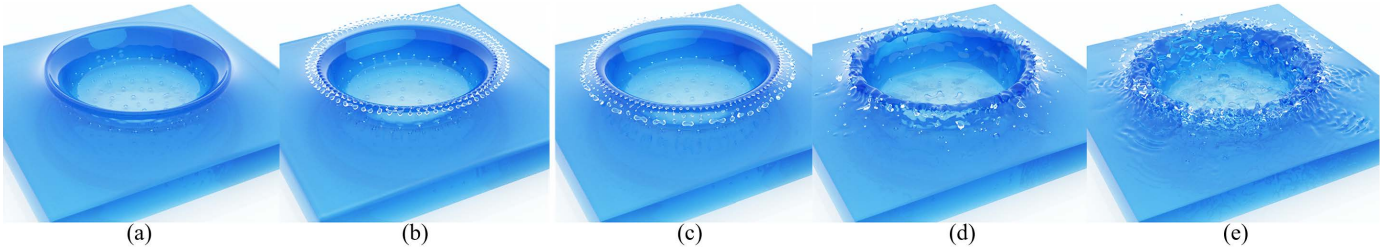

Fig. 15: **Stability for density ratio of 500.** For an example involving a ball of liquid falling into a pool of liquid (for an air-liquid density ratio of 500 as used in [17]), we show (a) the simulation obtained with their original method for the *largest possible* Reynolds number, here $Re_{\max} = 400$, before their solver blows up; if we add just our velocity-limiting adaptive viscosity to their approach, their modified solver can now handle (b) $Re_{\max} = 1000$. In contrast, our solver *without using* any velocity-limiting adaptive viscosity (c) can *already* match the results (b) of [17] for $Re = 1000$, and can go all the way to $Re_{\max} = 5000$ (d); *with* the adaptive viscosity added, we remain stable until $Re_{\max} = 16000$ (e). As a rough estimate based on the actual values of Re, this means that [17] could only simulate a small drop of water in a glass, while we can simulate all the way to having a ton of water being dropped in a pond (hence the surface waves) without having numerical blowups.

with both our solver and the one from [17] on *a whole range* of Reynolds numbers (up to 10^5) and density ratios (up to 1000). Fig. 7 indicates the regions where each solver were able to run the animation without blowups. As is evident on the final plot, we cover a range area over 5 times larger than the solver we started from. Fig. 8 also offers insight on how the various improvements we formulated over [17] impact the result for a simple dam break in 2D.

With CG and real-life flows. To further validate our solver ability to produce realistic liquid flows, we compare in Fig. 16 our simulation result of the glugging effect with the stream function solver of [8], which can also automatically capture bubbles in liquid flows, and to a video-captured real experiment similar to our simulation setup. Note that it is difficult to match the real experiment exactly in simulation, in particular, because of the inexact shape of the opening; moreover, the approach in [8] does not offer any control over the density ratio, but claims that it is relevant for water-air, so we use their results as is; consequently, we only compare the most critical macroscopic phenomena (presence of bubbles, their sizes and typical motions, etc). Fig. 16(a) is the one of the snapshots captured from the real-life experiment, while Fig. 16(b) is the corresponding result from the stream function solver with a background grid of size $128 \times 256 \times 128$, while around liquid-air interfaces, each grid cell contains 12 to 14 particles. Since particles are involved in their computations, it creates a higher resolution near interfaces. Thus, in order to match the number of degrees of freedom used to discretize the flow and its interface, our simulation in Fig. 16 (c) uses a resolution of $300 \times 600 \times 300$, since we do not currently support adaptive resolution. It is clear from the figure, and even more obvious from the video sequences, that the stream function solver misses bubbles inside the liquid near the bottom container, whereas our solver can faithfully retain these deep bubbles. The bubble shapes formed during the sequence are also more realistic than with the stream function solver as compared to the real experiment. In addition, wetting can be observed especially at the end of the simulation, just like in the real experiment, whereas such phenomena cannot be reproduced in the stream function solver (or in many other existing multiphase flow simulators in graphics). Finally, we note that the

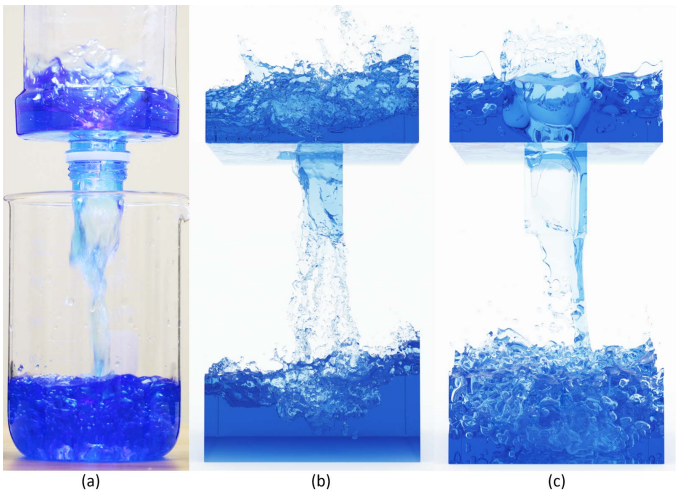

Fig. 16: **Real vs. simulated glugging.** (a) a real experiment exhibiting glugging is qualitatively reproduced with (b) the stream function solver [8] and (c) our solver. While the stream function solver do not match the right density of bubbles in the bottom container, our method captures the glugging phenomenon (wetting included) quite closely.

computational time required by their method (with their code run on our 28-core machine) is nearly 30% faster than our simulation times (once normalized by the number of degrees of freedom used in the simulation), proving that the parallel nature of our LBM-PF makes it competitive in terms of overall efficiency.

7 Conclusion

In this paper, we argued that with proper numerical treatments of gradient estimates and improved collision terms, a kinetic solver can offer an effective and unified computational framework for complex liquid simulations with which many multiphase flow phenomena can be concurrently captured. The resulting solver is as efficient as most ad-hoc CG fluid solvers designed to exhibit only a subset of typical multiphase behavior—but much more general and flexible. Perhaps more strikingly, our solver is also significantly more stable and sometimes also more accurate than existing CFD simulators, for which kinetic

approaches are the current industry standard in multiphase flows. We show through various flow simulations that typical multiphase behaviors such as glugging and wetting can be simulated faithfully and efficiently, even for real-life density ratios (800 for air-water) and Reynolds numbers (over 400 for interesting turbulent flows). While macroscopic solvers have dominated computer animation over the past two decades, they have generated exquisite simulations of often very viscous fluids (from dough, to honey, to slightly syrupy fluids), our paper challenges our community by proposing a computational alternative that is in nearly all aspects superior to current methods when it comes to barely-viscous real-life liquids.

Limitations. Our method still suffers from a number of limitations which will require future work. Although many liquid phenomena can be captured with our method more faithfully than existing solvers, its memory usage is up to two to three times larger than competing solvers; this limitation mostly stems from our absence of adaptive sampling, forcing the use of fine grids. Adaptive grids, able to capture finer sampling of the interface geometry more economically, are likely to address this issue, but a proper treatment of the mesoscopic simulations over these new data structures needs to be devised. We also did not address the case of coupling with deformable solids, which would bring a whole slew of interesting applications such as blood flow simulation. More in-depth investigation on how to efficiently and accurately simulate the Boltzmann transport equation may also uncovered a series of further improvements, possibly with unexpected applicability beyond the multiphase flow realm.

Acknowledgment

The authors would like to thank Dr. Ryoichi Ando from National Institute of Informatics, Tokyo, Japan, for sharing his fluid simulation codes for comparison and Jinglei Yang from the University of California, Santa Barbara, for her early help on rendering. This work is supported by the Young Scientists Fund of the National Natural Science Foundation of China (Grant No. 61502305), as well as startup funds from ShanghaiTech University. We also thank the reviewers for their comments.

References

[1] N. Foster and R. Fedkiw, "Practical animation of liquids," in *Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, 2001, pp. 23–30.

[2] V. Mihalef, D. Metaxas, and M. Sussman, "Animation and control of breaking waves," in *Symposium on Computer Animation*, 2004, pp. 315–324.

[3] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 965–972, 2005.

[4] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 40:1–6, 2009.

[5] C. Wojtan, M. Müller-Fischer, and T. Brochu, "Liquid simulation with mesh-based surface tracking," *ACM SIGGRAPH Course Notes*, 2011.

[6] D. Kim, O.-Y. Song, and H.-S. Ko, "A practical simulation of dispersed bubble flow," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 70:1–5, 2010.

[7] L. Boyd and R. Bridson, "MultiFLIP for energetic two-phase fluid simulation," *ACM Transactions on Graphics*, vol. 31, no. 2, pp. 16:1–12, 2012.

[8] R. Ando, N. Thuerey, and C. Wojtan, "A stream function solver for liquid simulations," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 53:1–9, 2015.

[9] S. Patkar, M. Aanjaneya, D. Karpman, and R. Fedkiw, "A hybrid Lagrangian-Eulerian formulation for bubble generation and dynamics," in *Symp. Comp. Anim.*, 2013, pp. 105–114.

[10] F. Da, D. Hahn, C. Batty, C. Wojtan, and E. Grinspun, "Surface-only liquids," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 78:1–12, 2016.

[11] Y. Zhang, H. Wang, S. Wang, Y. Tong, and K. Zhou, "A deformable surface model for real-time water drop animation," *IEEE Trans. Vis. Comp. Graph.*, vol. 18, no. 8, pp. 1281–1289, 2012.

[12] S. Patkar and P. Chaudhuri, "Wetting of porous solids," *IEEE Trans. Vis. Comp. Graph.*, vol. 19, no. 9, pp. 1592–1604, 2013.

[13] S. Chen and G. D. Doolen, "Lattice Boltzmann method for fluid flows," *Annual Review of Fluid Mechanics*, vol. 30, no. 1, pp. 329–364, 1998.

[14] H. Liu, A. J. Valocchi, and Q. Kang, "Three-dimensional lattice Boltzmann model for immiscible two-phase flow simulations," *Physical Review E*, vol. 85, no. 4, pp. 046 309:1–14, 2012.

[15] D. Lycett-Brown, K. H. Luo, R. Liu, and P. Lv, "Binary droplet collision simulations by a multiphase cascaded lattice Boltzmann method," *Physics of Fluids*, vol. 26, pp. 023 303:1–26, 2014.

[16] M. Geier, A. Fakhari, and T. Lee, "Conservative phase-field lattice Boltzmann model for interface tracking equation," *Physical Review E*, vol. 91, no. 6, pp. 063 309:1–11, 2015.

[17] A. Fakhari, D. Bolster, and L.-S. Luo, "A weighted multiple-relaxation-time lattice Boltzmann method for multiphase flows and its application to partial coalescence cascades," *Journal of Computational Physics*, vol. 341, pp. 22–43, 2017.

[18] W. Li, X. Wei, and A. Kaufman, "Implementing lattice Boltzmann computation on graphics hardware," *Visual Computer*, vol. 19, no. 7, pp. 444–456, 2003.

[19] J. Tölke, "Implementation of a lattice Boltzmann kernel using the Compute Unified Device Architecture developed by NVidia," *Computing and Visualization in Science*, vol. 13, no. 1, pp. 29–39, 2008.

[20] N. Thürey, T. Pohl, and U. Rüde, "Hybrid parallelization techniques for lattice Boltzmann free surface flows," in *Parallel Computational Fluid Dynamics 2007*. Springer, 2009, pp. 179–186.

[21] C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux, "A new approach to the lattice Boltzmann method for Graphics Processing Units," *Computers & Mathematics with Applications*, vol. 61, no. 12, pp. 3628–3638, 2011.

[22] P. Lallemand and L.-S. Luo, "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability," *Physical Review E*, vol. 61, no. 6, pp. 6546–6562, 2000.

[23] R. Huang, H. Wu, and P. Cheng, "A new lattice Boltzmann model for solid–liquid phase change," *International Journal of Heat and Mass Transfer*, vol. 59, pp. 295–301, 2013.

[24] S. Saito, A. De Rosis, A. Festuccia, A. Kaneko, Y. Abe, and K. Koyama, "Color-gradient lattice Boltzmann model with nonorthogonal central moments: Hydrodynamic melt-jet breakup simulations," *arXiv preprint arXiv:1804.08923*, 2018.

[25] S. Nabavizadeh, M. Eshraghi, and S. Felicelli, "A comparative study of multiphase lattice Boltzmann methods for bubble-dendrite interaction during solidification of alloys," *Appl. Sci.*, vol. 9, no. 1, pp. 57:1–24, 2018.

[26] Q. Li, K. H. Luo, Q. Kang, Y. He, Q. Chen, and Q. Liu, "Lattice Boltzmann methods for multiphase flow and phase-change heat transfer," *Progress in Energy and Combustion Science*, vol. 52, pp. 62–105, 2016.

[27] Dassault System, "Xflow," software package. [Online]. Available: https://www.simuleon.com/simulia-xflow-cfd/

[28] M. Desbrun and M.-P. Cani-Gascuel, "Active implicit surface for animation," in *Graphics Interface*, 1998, pp. 143–150.

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

[30] D. Enright, F. Losasso, and R. Fedkiw, "A fast and accurate semi-lagrangian particle level set method," *Computers and Structures*, vol. 83, no. 6-7, pp. 479–490, 2005.

[31] B. Kim, Y. Liu, I. Llamas, X. Jiao, and J. Rossignac, "Simulation of bubbles in foam with the volume control method," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 98:1–10, 2007.

[32] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," in *ACM Transactions on Graphics*, 2004, pp. 457–462.

[33] N. Heo and H.-S. Ko, "Detail-preserving fully-eulerian interface tracking framework," *ACM Transactions on Graphics*, vol. 29, no. 6, pp. 176:1–8, 2010.

[34] T. Kim, J. Tessendorf, and J. Thurey, "Closest point turbulence for liquid surfaces," *ACM Transactions on Graphics*, vol. 32, pp. 15:1–13, 2013.

[35] V. Mihalef, B. Unlusu, D. Metaxas, M. Sussman, and M. Y. Hussaini, "Physics based boiling simulation," in *Symposium on Computer Animation*, 2006, pp. 317–324.

[36] M. Bojsen-Hansen and C. Wojtan, "Liquid surface tracking with error compensation," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 79:1–10, 2013.

[37] ——, "Generalized non-reflecting boundaries for fluid re-simulation," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 96:1–7, Jul. 2016.

[38] N. Chentanez, M. Müller, M. Macklin, and T.-Y. Kim, "Fast grid-free surface tracking," *ACM Transactions on Graphics*, vol. 34, pp. 48:1–11, 2015.

[39] M. Desbrun and M.-P. Gascuel, "Smoothed particles: A new paradigm for animating highly deformable bodies," in *Computer Animation and Simulation*, 1996, pp. 61–76.

[40] L. Huang, T. Hädrich, and D. L. Michels, "On the accurate large-scale simulation of ferrofluids," *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 93:1–15, 2019.

[41] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Symposium on Computer Animation*, 2003, pp. 154–159.

[42] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, "Adaptively sampled particle fluids," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 48:1–7, 2007.

[43] J. Yu and G. Turk, "Reconstructing surfaces of particle-based fluids using anisotropic kernels," in *Symposium on Computer Animation*, 2010, pp. 217–225.

[44] H. Schechter and R. Bridson, "Ghost SPH for animating water," *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.

[45] X. He, H. Wang, F. Zhang, H. Wang, G. Wang, and K. Zhou, "Robust simulation of sparsely sampled thin features in SPH-based free surface flows," *ACM Transactions on Graphics*, vol. 34, no. 1, 2014.

[46] M. Macklin and M. Müller, "Position based fluid," *ACM Transactions on Graphics*, vol. 32, pp. 104:1–5, 2013.

[47] J. Bender and D. Koschier, "Divergence-free SPH for incompressible and viscous fluids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1193–1206, 2016.

[48] D. Koschier and J. Bender, "Density maps for improved SPH boundary handling," in *Symposium on Computer Animation*, 2017, pp. 1–10.

[49] S. Band, C. Gissler, M. Ihmsen, J. Cornelis, A. Peer, and M. Teschner, "Pressure boundaries for implicit incompressible SPH," *ACM Transactions on Graphics*, vol. 37, no. 2, pp. 14:1–11, 2018.

[50] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graphical Models and Image Processing*, vol. 58, pp. 471–493, 1996.

[51] C. Batty and R. Bridson, "Accurate viscous free surfaces for buckling, coiling, and rotating liquids," in *Symposium on Computer Animation*, 2008, pp. 219–228.

[52] R. Ando, N. Thurey, and C. Wojtan, "Highly adaptive liquid simulations on tetrahedral meshes," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 103:1–10, 2013.

[53] J. Cornels, M. Ihmsen, A. Peer, and M. Teschner, "IISPH-FLIP for incompressible fluids," *Computer Graphics Forum*, vol. 33, no. 2, pp. 255–262, 2014.

[54] V. C. Azevedo, C. Batty, and M. M. Oliveira, "Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps," *ACM Transactions on Graphics*, vol. 35, pp. 97:1–12, 2016.

[55] C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran, "A polynomial particle-in-cell method," *ACM Transactions on Graphics*, vol. 36, no. 6, p. 222, 2017.

[56] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, "The affine particle-in-cell method," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 51:1–10, 2015.

[57] O. Mercier, C. Beauchemin, N. Thuerey, T. Kim, and D. Nowrouzezahrai, "Surface turbulence for particle-based liquid simulations," *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 102:1–10, 2015.

[58] N. Thuerey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Gross, "Real-time simulations of bubbles and foam within a shallow water framework," in *Symposium on Computer Animation*, 2007, pp. 191–198.

[59] B. Kim, "Multi-phase fluid simulations using regional level sets," *ACM Transactions on Graphics*, vol. 29, no. 6, pp. 175:1–8, 2010.

[60] J. Cho and H.-S. Ko, "Geometry-aware volume-of-fluid method," *Computer Graphics Forum*, vol. 32, no. 2, pp. 379–388, 2013.

[61] T. R. Langlois, C. Zheng, and D. L. James, "Toward animating water with complex acoustic bubbles," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 95:1–13, 2016.

[62] B. Ren, C. Li, X. Yan, M. C. Lin, J. Bonet, and S.-M. Hu, "Multiple-fluid SPH simulation using a mixture model," *ACM Trans. Graph.*, vol. 33, no. 5, pp. 171:1–11, 2014.

[63] X. Yan, Y.-T. Jiang, C. Li, R. R. Martin, and S.-M. Hu, "Multiphase SPH simulation for interactive fluids and solids," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 79:1–11, 2016.

[64] T. Yang, J. Chang, M. C. Lin, R. R. Martin, J. J. Zhang, and S. min Hu, "A unified particle system framework for multi-phase, multi-material visual simulations," *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 224:1–13, 2017.

[65] F. Zhang, X. Zhang, K. Y. Sze, Y. Lian, and Y. Liu, "Incompressible material point method for free surface flow," *Journal of Computational Physics*, vol. 330, pp. 92–110, 2017.

[66] Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun, "Continuum foam: A material point method for shear-dependent flows," *ACM Transactions on Graphics*, vol. 34, no. 5, pp. 160:1–20, 2015.

[67] M. Gao, A. Pradhana, X. Han, Q. Guo, G. Kot, E. Sifakis, and C. Jiang, "Animating fluid sediment mixture in particle-laden flows," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 149:1–11, 2018.

[68] F. de Goes, C. Wallez, J. Huang, D. Pavlov, and M. Desbrun, "Power particles: an incompressible fluid solver based on power diagrams," *ACM Transactions on Graphics*, vol. 34, pp. 50:1–11, 2015.

[69] M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis, "Power diagrams and sparse paged grids for high resolution adaptive liquids," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 140:1–12, 2017.

[70] B. Solenthaler and R. Pajarola, "Density contrast SPH interfaces," in *Symposium on Computer Animation*, 2008, pp. 211–218.

[71] D. d'Humières, "Multiple–relaxation–time lattice Boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002.

[72] A. De Rosis, "Nonorthogonal central-moments-based lattice Boltzmann scheme in three dimensions," *Physical Review E*, vol. 95, no. 1, p. 013310, 2017.

[73] N. Thürey and U. Rüde, "Free surface lattice-Boltzmann fluid simulations with and without level sets," in *Vision, Modeling and Visualization*, 2004, pp. 199–207.

[74] X. Wei, W. Li, K. Mueller, and A. E. Kaufman, "The lattice-Boltzmann method for simulating gaseous phenomena," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 2, pp. 164–176, 2004.

[75] Y. Zhao, F. Qiu, Z. Fan, and A. Kaufman, "Flow simulation with locally-refined LBM," in *Symposium on Interactive 3D Graphics and Games*, 2007, pp. 181–188.

[76] U. R. Alim, A. Entezari, and T. Moller, "The lattice-Boltzmann method on optimal sampling lattices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 4, pp. 630–641, 2009.

[77] N. Akinci, G. Akinci, and M. Teschner, "Versatile surface tension and adhesion for SPH fluids," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 965–972, 2013.

[78] X. Liu, W.-M. Pang, J. Qin, and C.-W. Fu, "Turbulence simulation by adaptive multi-relaxation lattice Boltzmann modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 2, pp. 289–302, 2014.

[79] W. Li, K. Bai, and X. Liu, "Continuous-scale kinetic fluid simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 9, pp. 2694–2709, Sep. 2019.

[80] Y. Guo, X. Liu, and X. Xu, "A unified detail-preserving liquid simulation by two-phase lattice Boltzmann modeling," *IEEE Transaction on Visualization and Computer Graphics*, vol. 23, no. 5, pp. 1479–1491, 2017.

[81] A. K. Gunstensen, D. H. Rothman, S. Zaleski, and G. Zanetti, "Lattice Boltzmann model of immiscible fluids," *Physical Review A*, vol. 43, no. 8, pp. 4320–4327, 1991.

[82] S. Leclaire, M. Reggio, and J.-Y. Trépanier, "Isotropic color gradient for simulating very high-density ratios with a two-phase flow lattice Boltzmann model," *Computers & Fluids*, vol. 48, no. 1, pp. 98–112, 2011.

[83] Y. Ba, H. Liu, Q. Li, Q. Kang, and J. Sun, "Multiple-relaxation-time color-gradient lattice Boltzmann model for simulating two-phase flows with high density ratio," *Physical Review E*, vol. 94, no. 2, pp. 023 310:1–15, 2016.

[84] M. R. Swift, W. Osborn, and J. Yeomans, "Lattice Boltzmann simulation of nonideal fluids," *Physical Review Letters*, vol. 75, no. 5, pp. 830–833, 1995.

[85] J. Shao, C. Shu, H. Huang, and Y. Chew, "Free-energy-based lattice Boltzmann model for the simulation of multiphase flows with density contrast," *Physical Review E*, vol. 89, no. 3, pp. 033 309:1–14, 2014.

[86] X.-D. Niu, Y. Li, Y.-R. Ma, M.-F. Chen, X. Li, and Q.-Z. Li, "A mass-conserving multiphase lattice Boltzmann model for simulation of multiphase flows," *Physics of Fluids*, vol. 30, no. 1, pp. 013 302:1–13, 2018.

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

[88] R. Huang and H. Wu, "Third-order analysis of pseudopotential lattice Boltzmann model for multiphase flow," *Journal of Computational Physics*, vol. 327, pp. 121–139, 2016.

[89] G. Falcucci, S. Ubertini, C. Biscarini, S. Di Francesco, D. Chiappini, S. Palpacelli, A. De Maio, and S. Succi, "Lattice Boltzmann methods for multiphase flow simulations across scales," *Communications in Computational Physics*, vol. 9, no. 2, pp. 269–296, 2011.

[90] C. Coreixas, B. Chopard, and J. Latt, "Comprehensive comparison of collision models in the lattice Boltzmann framework: Theoretical investigations," *Phys. Rev. E*, vol. 100, pp. 033 305:1–46, 2019.

[91] T. Lee and L. Liu, "Lattice Boltzmann simulations of micron-scale drop impact on dry surfaces," *Journal of Computational Physics*, vol. 229, no. 20, pp. 8045–8063, 2010.

[92] C.-W. Shu, "Numerical experiments on the accuracy of eno and modified eno schemes," *Journal of Scientific Computing*, vol. 5, pp. 127–149, 1990.

[93] X.-D. Liu, S. Osher, and T. Chan, "Weighted essentially non-oscillatory schemes," *Journal of Computational Physics*, vol. 115, no. 1, pp. 200–212, 1994.

[94] T. Lee and L. Liu, "Lattice Boltzmann simulations of micron-scale drop impact on dry surfaces," *Journal of Computational Physics*, vol. 229, no. 20, pp. 8045–8063, 2010.

[95] F. Menter and Y. Egorov, "The scale-adaptive simulation method for unsteady turbulent flow predictions. part 1: theory and model description," *Flow, Turbulence and Combustion*, vol. 85, no. 1, pp. 113–138, 2010.

[96] L. Li, R. Mei, and J. F. Klausner, "Boundary conditions for thermal lattice Boltzmann equation method," *Journal of Computational Physics*, vol. 237, pp. 366–395, 2013.

[97] T. Zhang, B. Shi, Z. Guo, Z. Chai, and J. Lu, "General bounce-back scheme for concentration boundary condition in the lattice-Boltzmann method," *Physical Review E*, vol. 85, no. 1, pp. 016 701:1–14, 2012.

[98] D. Jacqmin, "Contact-line dynamics of a diffuse fluid interface," *Journal of Fluid Mechanics*, vol. 402, pp. 57–88, 2000.

[99] S. Popinet, "Gerris flow solver," 2013. [Online]. Available: http://gfs.sourceforge.net/wiki/

[100] W. Jakob, "Mitsuba renderer," 2010. [Online]. Available: http://www.mitsuba-renderer.org/

[101] N. Akinci, G. Akinci, and M. Teschner, "Versatile surface tension and adhesion for SPH fluids," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 182:1–8, 2013.

Wei Li received his B.S. degree in computer science from Northwestern Polytechnical University, Shaanxi, China in 2015. He is now a Ph.D. student at School of Information Science and Technology (SIST) of ShanghaiTech University. He is interested in making virtual worlds with computer, and more specifically in the area of computer graphics, physically-based animation, computational physics, scientific computing, rendering, and robotic learning.

Daoming Liu is currently doing postdoctoral research in School of Information Science and Technology, ShanghaiTech University. He obtained his Ph.D. degree in Communication and Information System from the Shanghai Advanced Research Institute, Chinese Academy of Sciences in 2017. During his Ph.D., he studied fluid mechanics in Delft University of Technology. His research interests are mainly on the physical modeling, numerical simulation, and scientific visualization of fluid dynamics and scalar transport phenomena. He is also interested in high performance computing, machine learning, and wireless sensor networks.

Mathieu Desbrun is currently the Carl Braun Professor at Caltech, and a Visiting Professor in the School of Information Science and Technology, ShanghaiTech University. His research interests include geometry processing, geometric mechanics and simulation. He has served as technical paper chair or program committee member for most of the graphics conferences, including ACM SIGGRAPH, the Symposium on Computer Animation and the Symposium on Geometry Processing. He hates writing his own bio.

Jin Huang received the Ph.D. degree from Computer Science Department, Zhejiang University, Hangzhou, China, in 2007, with Excellent Doctoral Dissertation Award of China Computer Federation. He is a full professor in the State Key Lab of CAD & CG of Zhejiang University. His research interests include geometry processing and physically-based simulation. He is an associate editor of Computer Aided Geometric Design, and has served as PC for ACM SIGGRAPH, ACM SIGGRAPH Asia, and chair of Symposium on Computer Animation, Geometric Modeling and Processing etc.

Xiaopei Liu is currently an assistant professor at School of Information Science and Technology, ShanghaiTech University. He received his Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong in 2010, and later worked as a Research Fellow at Nanyang Technological University. His current research interests include computer graphics and physically-based simulation, computational physics, scientific visualization, machine learning and robotics, as well as parallel computing techniques, with their applications in different domains.