Efficient Kinetic Simulation of Two-Phase Flows

WEI LI, Inria
YIHUI MA, ShanghaiTech University
XIAOPEI LIU, ShanghaiTech University
MATHIEU DESBRUN, Inria / Ecole Polytechnique

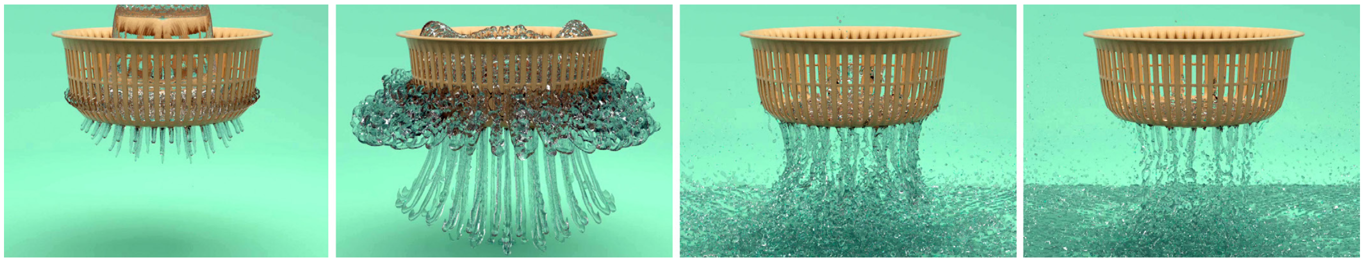

Fig. 1. **Multiphase flow simulation**. We contribute an efficient and robust approach to simulating two-phase flows of nearly inviscid & incompressible fluids such as water and air interacting with complex solids. Our kinetic solver couples the incompressible Navier-Stokes equations with a conservative phase-field equation to evolve the two fluids and their interface in time in a massively-parallel fashion. Key improvements in the lattice-Boltzmann collision operator and in boundary handling bring accuracy and robustness, allowing us to capture all salient multiphase flow behaviors (here, water is dropped in a colander, exhibiting complex patterns as it rushes through the perforations) while improving efficiency and reducing memory use compared to other kinetic solvers.

Real-life multiphase flows exhibit a number of complex and visually appealing behaviors, involving bubbling, wetting, splashing, and glugging. However, most state-of-the-art simulation techniques in graphics can only demonstrate a limited range of multiphase flow phenomena, due to their inability to handle the real water-air density ratio and to the large amount of numerical viscosity introduced in the flow simulation and its coupling with the interface. Recently, kinetic-based methods have achieved success in simulating large density ratios and high Reynolds numbers efficiently; but their memory overhead, limited stability, and numerically-intensive treatment of coupling with immersed solids remain enduring obstacles to their adoption in movie productions. In this paper, we propose a new kinetic solver to couple the incompressible Navier-Stokes equations with a conservative phase-field equation which remedies these major practical hurdles. The resulting two-phase immiscible fluid solver is shown to be efficient due to its massively-parallel nature and GPU implementation, as well as very versatile and reliable because of its enhanced stability to large density ratios, high Reynolds numbers, and complex solid boundaries. We highlight the advantages of our solver through various challenging simulation results that capture intricate and turbulent air-water interaction, including comparisons to previous work and real footage.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Multiphase flow, turbulent flow simulation, lattice Boltzmann method

ACM Reference Format:
Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient Kinetic Simulation of Two-Phase Flows. *ACM Trans. Graph.* 41, 4, Article 114 (July 2022), 17 pages. https://doi.org/10.1145/3528223.3530132

Authors' addresses: W. Li, M. Desbrun, Inria Saclay, Institut Polytechnique de Paris, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France; Y. Ma, X. Liu, Shanghaitech University, 393 Huaxia Middle Road, Shanghai 201210, China.

1 INTRODUCTION

With recent advances in fluid simulation, visually-realistic digital representations of liquid motion have become commonplace in movie productions. While the early use of single-phase simulation [Foster and Fedkiw 2001; Zhu and Bridson 2005; Solenthaler and Pajarola 2009; Wojtan et al. 2011] offered only limited realism since the air surrounding the fluid was assumed to exert no pressure, multiphase flow simulation can exhibit a far richer palette of behaviors: the air-water and water-solid interactions often generate bubbles, splashes, glugging, and even the intricate wetting patterns that a fluid forms over hydrophilic or hydrophobic surfaces.

Over the last decade, the animation of immiscible multiphase flows in graphics has successfully demonstrated complex behaviors of bubbles [Kim et al. 2010; Boyd and Bridson 2012; Ando et al. 2015], surface tension effects [Patkar et al. 2013; Da et al. 2016] and wetting [Zhang et al. 2012; Patkar and Chaudhuri 2013]. However, cinematic realism of complex water-air interactions calls for large density ratios (around 800 for the most common case of air vs. water) and high Reynolds numbers ($\mathrm{Re} > 4,000$ for turbulent flows) to display the type of visual complexity that natural multiphase flow phenomena have to offer. These numerical requirements have been out of reach for state-of-the-art CG solvers, and remain a challenge even in computational fluid dynamics (CFD) [Lycett-Brown et al. 2014; Geier et al. 2015]. Recently, Li et al. [2021] introduced a kinetic approach which, for the first time, can simulate most of the typical multiphase flow phenomena, and for a range of fluid viscosity and density ratio that solvers in graphics strive to simulate in order to visually rival reality. This new solver relies on the lattice Boltzmann method (LBM) and uses a phase-field (PF) model to handle interfacial computations. However, their low-order collision model and boundary handling via simple bounce-back severely lack stability

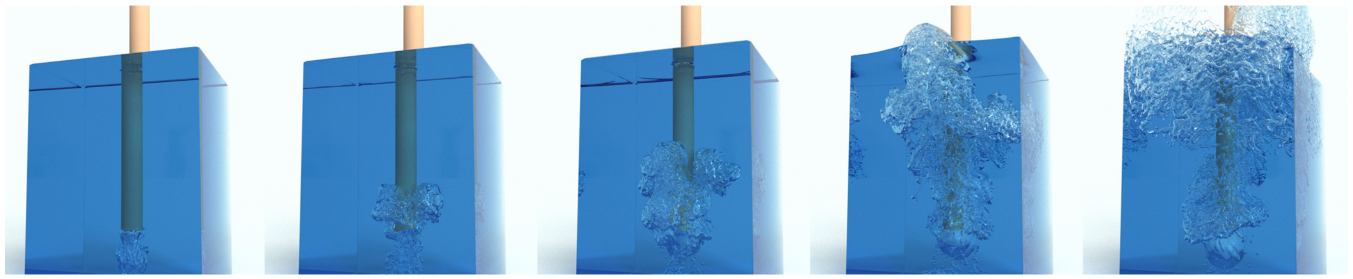

Fig. 2. **Blowing air into a glass of water.** In this simulation, air is being pushed down a straw immersed in a glass of water; a large cluster of bubbles forms at the bottom of the straw, before rising due to buoyancy, and eventually popping when it reaches the top.

when complex or fast air-fluid interactions occur, see Fig. 8. Moreover, its large memory footprint (in particular due to the encoding of the phase field via twenty-seven distribution functions) is a major impediment to its use in industrial applications.

Overview. In this paper, we introduce a kinetic approach to multiphase fluid simulation particularly well suited to capture efficiently and robustly the intricate and visually-appealing behaviors of multiphase flows. We show that our LBM-PF numerical scheme offers a unified framework to simulate the wide gamut of multiphase flow behaviors as demonstrated for instance in Fig. 1. It is not only much more *efficient* in simulation time than many existing CG simulation techniques, but as *accurate* as (and *more stable* than) existing CFD approaches to multiphase flows as well, through the use of an accurate collision model (based on a non-orthogonal central-moment formulation) and a robust discretization of the Neumann boundary conditions on solid obstacles. Finally, its reduced memory footprint compared to previous kinetic two-phase flow solvers renders it particularly practical for CG applications.

2 PREVIOUS WORK

Before introducing our solver, it is instructive to briefly review existing work in multiphase fluid simulation to note the strengths and weaknesses of previous methods.

2.1 Free surface fluids

Lagrangian discretizations of fluid based on SPH were initially favored for their efficiency [Desbrun and Gascuel 1996; Müller et al. 2003]. The resulting visual blobbiness for low to medium particle counts was a major drawback of this family of approaches, even if various palliative treatments were formulated [Desbrun and Cani-Gascuel 1998; Zhu and Bridson 2005]. Improvements in incompressibility [Solenthaler and Pajarola 2009; Bender and Koschier 2016] and in the numerical treatment of free surface and solid boundaries [Schechter and Bridson 2012; Koschier and Bender 2017; Band et al. 2018] quickly followed. For viscous fluids, Particle-In-Cell [Foster and Metaxas 1996] (PIC) and Fluid-Implicit-Particle [Zhu and Bridson 2005; Batty and Bridson 2008; Cornels et al. 2014; Azevedo et al. 2016; Fu et al. 2017] (FLIP) techniques (or hybrid methods, such as [Cornels et al. 2014] mixing FLIP and SPH) were also successfully used. Nevertheless, a representation of the interface and its motion based on the level set method proved to be an attractive alternative to particle methods [Foster and Fedkiw 2001]. An improved control over local volume change [Enright et al. 2002, 2005; Kim et al. 2007] and adaptive sampling [Losasso et al. 2004; Heo and Ko 2010] allowed for detailed and realistic simulation of fluid flows, potentially post-processed to further enhance the visual appearance of small wrinkles on the fluid surface [Kim et al. 2013]. Other approaches to interface tracking, based on volume-of-fluid [Mihalef et al. 2004, 2006] or explicit mesh representations [Wojtan et al. 2011; Bojsen-Hansen and Wojtan 2013; Chentanez et al. 2015; Bojsen-Hansen and Wojtan 2016; Da et al. 2016], have also been demonstrated; some hybrid methods can even capture sub-grid features such as thin liquid splashes, narrow air spaces, and droplets using cut-cells [Chen et al. 2020]. This whole family of free-surface methods is, however, limited by the fact that the air surrounding the fluid being simulated is always assumed to be exerting no pressure on the interface. Consequently, common fluid features such as bubbles cannot be directly simulated: a small amount of air trapped within the fluid will immediately collapse.

2.2 Navier-Stokes-based multiphase fluids

Ad-hoc inclusions of bubbles within a fluid can be performed efficiently [Kim et al. 2010; Thuerey et al. 2007; Goldade et al. 2020], but simulating the full spectrum of behaviors that multiphase flows exhibit has attracted increased attention over the past decade. A number of particle-based approaches were formulated: first, adapting SPH to handle multiple immiscible fluids of different densities using Navier-Stokes equations was introduced [Solenthaler and Pajarola 2008; Ren et al. 2014; Yan et al. 2016; Yang et al. 2017], followed by FLIP-based [Boyd and Bridson 2012] and MPM-based [Zhang et al. 2017; Yue et al. 2015; Gao et al. 2018] formulations. Exact volume preservation and improved treatment of surface tension and local momentum preservation near the interface were formulated using power particles [de Goes et al. 2015; Aanjaneya et al. 2017]. However, the blobbiness of these approaches limits their accuracy unless a huge number of particles is employed, which once again renders the use of interface tracking appealing. Both levelset (see, e.g., [Kim 2010] for the case of more than two immiscible fluids) and volume-of-fluid (VOF; see, e.g., [Cho and Ko 2013; Langlois et al. 2016]) discretizations have been used to encode the interface. Most related methods [Boyd and Bridson 2012; Kim et al. 2007; Losasso et al. 2006; Mihalef et al. 2006] rely on a variable density pressure projection [Kang et al. 2000] and the ghost fluid method [Hong and Kim 2005] to treat discontinuous jumps in fluid density and pressure at the interface, but a few authors proposed the use of a diffuse interface instead where the density is rapidly but smoothly changing [Song et al. 2005; Zheng et al. 2009]. While this interface-tracking

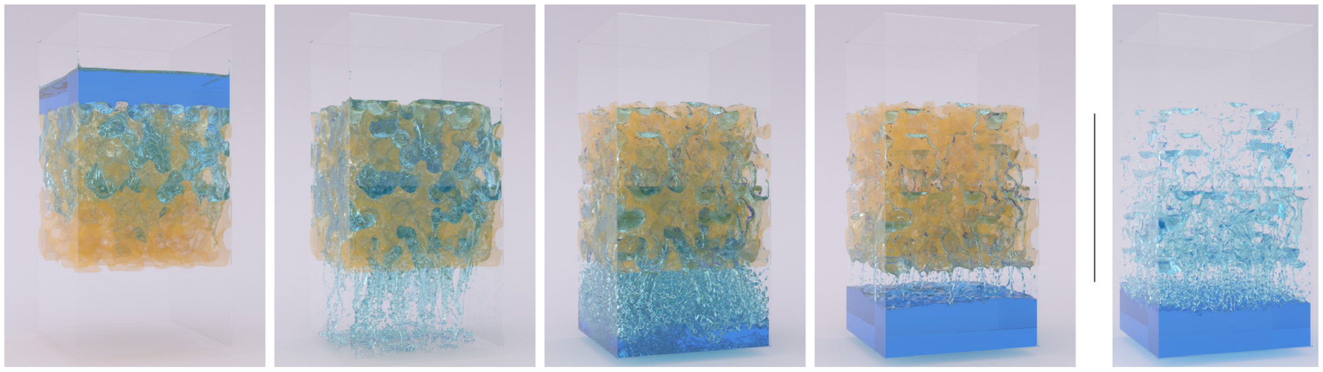
Fig. 3. **Flowing through pumice.** Water is poured on top of a porous rock containing a multitude of cavities and tunnels, making its way slowly through before falling into a container (left). Visualization of an intermediate frame *without* the pumice rock reveals the complexity of the flow (right).

family of approaches to multiphase flow simulation has been most popular, notable alternatives emerged too: an MPM formulation was introduced in [Su et al. 2021] to simulate viscoelastic liquids with phase change; a Lagrangian mesh-based approach to animate multiphase flow of immiscible fluid using unstructured moving meshes was also proposed for viscous fluids [Misztal et al. 2013]; and hybrid solvers using different numerical coupling between fluid velocity, pressure, and interface position were also explored [Saye 2016, 2017; Yang et al. 2021]. However, rare are the methods able to handle density ratios above 100, and none have demonstrated convincing results for turbulent flows. Additionally, many can only exhibit a subset of the typical multiphase flow phenomena such as bubbling, guggling, or wetting. Finally, just as argued in the single-phase case [Li et al. 2020], the efficiency of most of these incompressible Navier-Stokes solvers are hampered by their pressure projection step which does not parallelize well; moreover, capturing turbulent flows and proper interfacial forces both prevent the use of large time steps via stable semi-Lagrangian integration.

2.3 Kinetic multiphase fluids

While Navier-Stokes solvers are mainstream in graphics, the computational fluid dynamics (CFD) community has also explored so-called kinetic solvers for multiphase flow simulation, based on statistical physics where fluid dynamics is modeled through the evolution of a large number of *microscopic* fluid particles, moving and colliding: even at rest, fluid is not merely considered as a static continuum, but is teeming with molecules moving around. In the kinetic theory of statistical physics, these microscopic particles are efficiently encoded via a mesoscopic *probability distribution* of fluid particles. Formally, it employs a distribution function, $f(\boldsymbol{x}, \boldsymbol{v}, t)$, that indicates the *probability* for a microscopic particle to be at position $\boldsymbol{x}$ at time t and with velocity $\boldsymbol{v}$. Note that $\boldsymbol{v}$ denotes the possible velocity of a microscopic fluid particle, not the usual macroscopic fluid velocity $\mathbf{u}$; for example, even for a fluid at rest (i.e., $\boldsymbol{u} = 0$), the velocity $\boldsymbol{v}$ of a fluid particle may be arbitrarily away from zero: local fluid velocities just average to zero locally. The mesoscopic evolution of f is governed by the *continuous Boltzmann equation*,

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla f = \Omega(f - f^{\text{eq}}) + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f, \quad (1)$$

where $\boldsymbol{F}$ encapsulates all forcing terms (body and surface tension forces), while Ω is the so-called collision term modeling the change of distribution function $f(\boldsymbol{x}, \boldsymbol{v}, t)$ due to particle collisions. This operator effectively models the relaxation process of $f(\boldsymbol{x}, \boldsymbol{v}, t)$ towards its local equilibrium state f^{eq}, as a fluid over time tends to become distributed homogeneously if no external forces are applied. While this approach, called the lattice Boltzmann model (LBM), is used *as is* to simulate the motion of single-phase fluids, kinetic *multiphase flow* simulation requires introducing additional parameters to LBM that account for phase separation and interfacial tension. There exist six main families of approaches to that effect, derived from interparticle-potential models [Shan and Chen 1993, 1994; Chin 2002; Kang et al. 2004], multirange interaction models [Falcucci et al. 2007; Sbragaglia et al. 2007; Chibbaro et al. 2008], color-gradient models [Gunstensen et al. 1991; Grunau et al. 1993; Leclaire et al. 2011; Ba et al. 2016; Saito et al. 2018], pseudo-potential models [Swift et al. 1995; Shao et al. 2014; Niu et al. 2018], mean-field theory [He et al. 1998; Lee and Lin 2005; McCracken and Abraham 2005; Mukherjee and Abraham 2007a,b], or free-energy models [Orlandini et al. 1995; Swift et al. 1996; Holdych et al. 1998; Inamuro et al. 2000; Kalarakis et al. 2002; Zheng et al. 2006]. This last family is widely accepted as the current state-of-the-art approach for multiphase flows [Nabavizadeh et al. 2018]: it substitutes boundary conditions at the interface by a partial differential equation for the evolution of an auxiliary field called the phase field, based on a double-well potential describing the free energy density of the bulk of each phase. Yet, even if combining a phase field with an LBM simulation of the fluids can handle large density ratios *or* high Reynolds numbers, none of the existing CFD methods can handle these two required characteristics concurrently on a fixed grid resolution [Falcucci et al. 2011; Fakhari et al. 2017a]. Works in graphics usually have the same limitations (see, for instance, the two-phase kinetic method of [Guo et al. 2017] which only supports density ratios up to 20), with the notable exception of [Li et al. 2021] which has demonstrated stability over the widest range of density ratios and Reynold numbers thus far. However, this most recent work cannot handle thin obstacles and relies on a full D3Q27 lattice for the momentum and phase fields which imposes a much higher memory consumption (121 real numbers per grid node (Tab. 3) to store the current and previous time steps), hampering its wide adoption in practice since high-resolution simulations are usually sought after.

2.4 Discussion

There appears to be a dichotomy between efficiency and memory usage in the current state-of-the-art methods in multiphase flow simulation: while LBM-based approaches have clearly an edge on efficiency (owing to their massively-parallel implementation and absence of pressure projection), they require up to ten times more memory than the traditional macroscopic solvers focusing on the macroscopic equations of motion. However, kinetic approaches have proven to be very accurate; in particular, they are able to simulate a range of turbulence and density ratios that is out of reach for the solvers currently used in graphics. Yet, the robustness with which one can hope to simulate complex air-water interactions leaves a lot to be desired: classical boundary treatments in LBM, based on bounce-back and/or the immersed boundary method, often generate spurious pressure fluctuations or phase leakage near solid boundaries in most challenging examples. Therefore, developing an efficient kinetic solver for multiphase flows that can accurately capture intricate behaviors such as wetting and guggling with a reasonable memory footprint would offer a valuable simulation tool likely to be adopted by the graphics industry.

3 OUR MULTIPHASE FLUID SOLVER

This section presents the core of our contributions to offer an efficient and versatile incompressible multiphase flow simulator.

3.1 Overview

We begin with a brief overview of our approach, before delving into the details of each component in the remainder of this section.

Fluid equations. The momentum and continuity equations for incompressible multiphase flows are often described macroscopically as [Kendon et al. 2001; Badalassi et al. 2003; Li et al. 2012]:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, & \text{(2a)} \\ \dfrac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \nabla \cdot \Pi + \mathbf{F}_s + \mathbf{F}_b, & \text{(2b)} \\ \nabla \cdot \mathbf{u} = 0, & \text{(2c)} \end{cases}$$

where ρ and $\boldsymbol{u}$ are respectively the spatially-varying fluid density and velocity of the two fluids, p is the hydrodynamic pressure enforcing the incompressibility condition (2c), Π is the viscous stress tensor, while $\mathbf{F_b}$ and $\mathbf{F_s}$ are body and surface tension forces.

Our approach at a glance. Given the clear benefits of kinetic solvers discussed in Sec. 2, we adopt a lattice Boltzmann approach for our multiphase flow solver, with a *phase field* model of the interface location. The phase field $\phi(x, t)$ will be used to indicate density changes in space and time, with a steep change where the interface between the two fluids lies. Boundary conditions at the interface will be handled via this phase field, simplifying the simulation of flows with complex topology changes. However, we note that recent developments in computational multiphase flows have traded memory for stability: early works using a D3Q7 discretization for the phase field [Liang et al. 2018; Fakhari et al. 2019; Gruszczyński et al. 2020] were often unstable for typical values of mobility, which recently prompted Li et al. [2021] to go with a D3Q27 lattice to palliate these issues, at the cost of a nearly four-fold increase in memory. Instead, we propose to keep a D3Q27 lattice to capture the velocity field, but return to a D3Q7 lattice for the phase field, and increase the coupling accuracy — and thus stability — through a higher-order collision model and improved boundary treatments. Therefore, we perform an LBM-based integration of Eqs. (2) where the velocity and pressure fields are computed via a D3Q27-based distribution function g (Sec. 3.2), while the phase field encoding the density field (and thus, the interface between the two fluids) is handled via a D3Q7-based distribution function h (Sec. 3.3). The collision operators for both distribution functions are computed in central moment space to offer improved accuracy, while coupling between the two distributions is achieved via forces derived from their respective moments.

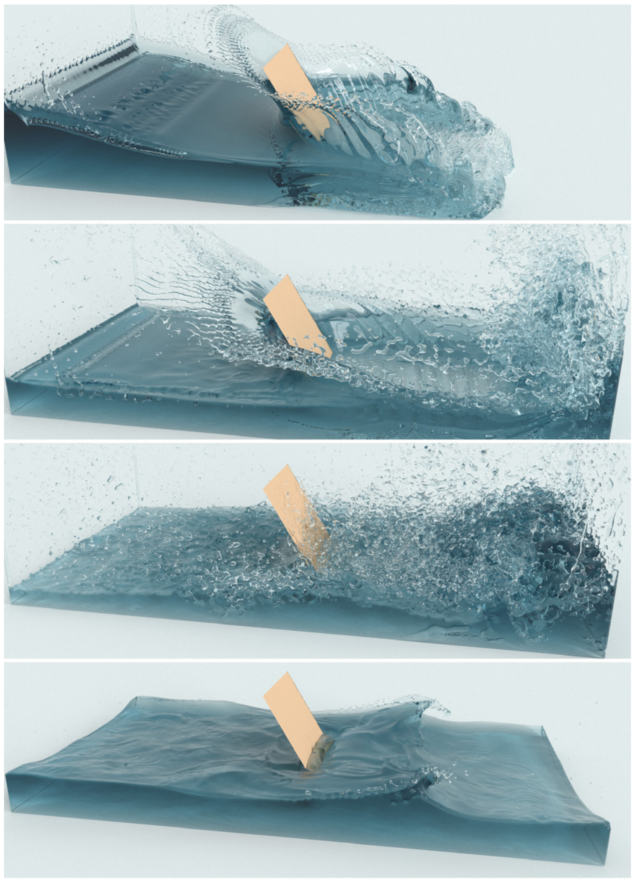
Fig. 4. **Dam break with thin obstacle.** A dam-break wave encounters a solid thin shell, forming an obvious crown splash and a wake zone, followed by a large amount of turbulence and bubbles, before subsiding.

3.2 Velocity-based fluid evolution

LBM solvers for single-phase fluids typically use a distribution function whose zeroth-order moments is the local fluid density. However, for multiphase flow simulations, a dedicated phase field is used to encode the spatially-varying density. Consequently, Fakhari et al. [2017b] proposed a change of variable (which we review in Sec. A of the supplemental material) such that the zeroth-order moment of the *modified distribution function* g is now proportional to the pressure, and for which the velocity can be deduced from the first-order moment and the forcing terms. We adopt this so-called "velocity-based" distribution function[1] g in our multiphase fluid simulation. After this change of variable, discretizing time (with LBM-normalized time steps $\delta t = 1$), space (using the nodes of a regular grid with LBM-normalized $\delta x = 1$) and velocity directions (using discrete velocities $\{\mathbf{c}_i\}_{0 \le i \le 26}$ at each grid node forming a D3Q27 lattice in 3D, see Fig. 6 (left)) turns Eq. (1) into the lattice-Boltzmann equations (LBE) of the form, for $i = 0 \ldots 26$:

$$g_i(\mathbf{x} + \mathbf{c}_i, t+1) - g_i(\mathbf{x}, t) = \Omega_i^g(\mathbf{x}, t) + G_i(\mathbf{x}, t), \tag{3}$$

where g_i represents the component of the distribution function in direction $\mathbf{c}_i$, while Ω_i^g and G_i account for the collision and forcing terms respectively in this direction. The equilibrium distribution g^{eq} for this velocity-based formulation is now expressed as:

$$g_i^{\text{eq}} = \Gamma_i + (p^* - 1) w_i^c - \frac{G_i}{2}, \tag{4}$$

where p^* is the *LBM-normalized pressure* (i.e., the dimensionless pressure p equals $\rho c^2 p^*$ with c being the D3Q27 speed of sound equal to $1/\sqrt{3}$ in 3D), w_i^c is the lattice weight for direction $\mathbf{c}_i$, and the distribution function Γ is expressed as:

$$\Gamma_i = w_i^c \left(1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c^2} + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{2c^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c^2} \right). \tag{5}$$

Finally, note that all the terms on the right-hand side of Eq. (2b) are counted as forcing terms in our multiphase context; that is, the components G_i are the projections of the total force $\mathbf{F}$ composed of

$$\mathbf{F} = \mathbf{F}_p + \mathbf{F}_\mu + \mathbf{F}_s + \mathbf{F}_b, \tag{6}$$

where $\mathbf{F}_p$ and $\mathbf{F}_\mu$ correspond, respectively, to the pressure gradient and viscosity forces. Aside from the last term (which embodies body forces in the fluid), these forces depend on the phase field ϕ so as to properly handle the interfacial coupling between the two fluids; we thus postpone discussing their expressions to Sec. 3.3.

Macroscopic variables. In order to solve Eqs. (3), we proceed through Strang splitting by applying the traditional two-step (collision-streaming) sequence, where the evaluation of the collision operator and forcing terms will be detailed next. The macroscopic hydrodynamic variables can be updated locally and independently through the moments of g and the forcing terms via:

$$p^*(\mathbf{x}, t) = \sum_i g_i(\mathbf{x}, t), \quad \mathbf{u}(\mathbf{x}, t) = \sum_i g_i(\mathbf{x}, t)\, \mathbf{c}_i + \frac{\mathbf{F}(\mathbf{x}, t)}{2}. \tag{7}$$

While our multiphase framework so far is a mix of existing formulations (each chosen for their stability and accuracy properties), we differ significantly in our estimation of collision and forcing terms, as we extend a high order approximation (based on a central-moment formulation) which has only been used for single-phase LBM [De Rosis et al. 2019; Li et al. 2020].

High-order central-moment collision model. Early LBM works used the Bhatnagar-Gross-Krook (BGK [Bhatnagar et al. 1954]) collision model, where the distribution function $\mathbf{g}$ is relaxed towards its equilibrium with a relaxation time τ directly proportional to the kinematic viscosity through $\tau = \nu / c^2$. In our two-phase flow context, each flow has its own potentially different viscosity — but we will see in Eq. (12) that the local viscosity can be directly inferred from the value of the phase field, making this relaxation time easily evaluated at each grid node. However, the poor numerical stability and results generated by this lattice BGK approximation due to truncation errors have led in the last decade to the proliferation of moment-based collision models: recent multiphase flow solvers have typically been using the multiple-relaxation-time (MRT) model [Fakhari et al. 2017a,b; Li et al. 2021], demonstrating far improved accuracy and stability over previous treatments. In our work, we adapt the high-order non-orthogonal central-moment multiple-relaxation-time (CM-MRT) model [Li et al. 2020] to our multiphase context instead to further enhance accuracy and stability. The idea is to project the $g_i - g_i^{\text{eq}}$ term first in a central moment space via a projection matrix, then relax each moment at a different time scale, then go back to the distribution space. We thus assemble the current vector $\mathbf{g} = (g_0, g_1, \ldots, g_{26})$ of all distribution components at a given node and the vector $\mathbf{g}^{\text{eq}}$ of all equilibrium distribution components for this same node; to determine the equilibrium distribution $\mathbf{g}^{\text{eq}}$, we take, instead of Eq. (5), the highest-order Hermite expansion of the distribution Γ (provided for the reader's convenience in Eq. (g) of the supplemental material). We then evaluate the vector $\boldsymbol{\Omega}^g = (\Omega_0^g, \Omega_1^g, \ldots, \Omega_{26}^g)$ containing all collision terms at the same node through:

$$\boldsymbol{\Omega}^g = -\mathbf{M}_c^{-1} \mathbf{S}_c \mathbf{M}_c (\mathbf{g} - \mathbf{g}^{\text{eq}}), \tag{8}$$

where $\mathbf{M}_c$ is the projection matrix to the central-moment space and the relaxation matrix $\mathbf{S}_c$ is a diagonal matrix for which some of the diagonal values are related to the local kinematic viscosity ν as detailed in [De Rosis et al. 2019]. We follow the approach of [Li et al. 2020] to compute the projection to and from the central moment space to improve computational efficiency. Note that this central moment based approach to collision is significantly more accurate than BGK: the dam break example from Fig. 5, for instance, cannot be completed with the BGK model unless the density ratio is lowered to 10. It is also better than the usual MRT treatment from [Suga et al. 2015; Fakhari et al. 2017b; Li et al. 2021] as Fig. 5 demonstrates: the spurious velocity artifacts present in the MRT approach are no longer visible when non-orthogonal central moment MRT is used.

Force terms in central moment space. The highest-order Hermite expansion (sixth order in 3D) of the forcing terms G_i yields a rather long expression [De Rosis et al. 2019]; see Eq. (h) of the supplemental material. Fortunately, one realizes that the projections $\tilde{G}_i$ of the G_i terms onto the non-orthogonal central moment space greatly simplifies as many components vanish, leaving only a few nonzero components:

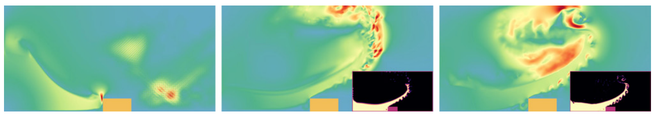
Fig. 5. **Comparing collision models.** Even for a simple 3D dam break with a grid-aligned obstacle, the MRT model [Suga et al. 2015] (left; 2D cut of the color-coded vector field magnitude shown as the water hits the obstacle) leads to pressure overshoots that crash the simulation early; the recent W-MRT collision model of [Li et al. 2021] leads to a stable simulation (center), but the velocity field is quite irregular, leading to a phase field (bottom inset) exhibiting misshapen Kelvin-Helmholtz instabilities; comparatively, our new central-moment MRT collision model demonstrates a smoother vector field and a well-captured Kelvin-Helmholtz instability for the phase field.

[1]This naming stems from the fact that the first-order moment (described later in Eqs. (7)) is related to the velocity, instead of the momentum in single-phase LBM works.

$$\begin{aligned} \tilde{G}_1 &= \frac{\boldsymbol{F}_x}{\rho}, & \tilde{G}_2 &= \frac{\boldsymbol{F}_y}{\rho}, & \tilde{G}_3 &= \frac{\boldsymbol{F}_z}{\rho}, \\ \tilde{G}_{10} &= 2c^2\frac{\boldsymbol{F}_x}{\rho}, & \tilde{G}_{11} &= 2c^2\frac{\boldsymbol{F}_y}{\rho}, & \tilde{G}_{12} &= 2c^2\frac{\boldsymbol{F}_z}{\rho}, \\ \tilde{G}_{23} &= c^4\frac{\boldsymbol{F}_x}{\rho}, & \tilde{G}_{24} &= c^4\frac{\boldsymbol{F}_y}{\rho}, & \tilde{G}_{25} &= c^4\frac{\boldsymbol{F}_z}{\rho}\,. \end{aligned} \tag{9}$$

Therefore, starting from the 3D force vector $\boldsymbol{F}$ including all four terms (see Eq. (6)), we compute the terms $\tilde{G}_i$ from Eq. (9), assemble them into a vector $\tilde{\mathbf{G}}$, then the final distribution components G_i are found as the 27 coordinates of the vector $\mathbf{G} = \mathbf{M}_c^{-1}\tilde{\mathbf{G}}$. Note that the original weighted MRT approach of Fakhari et al. [2017b] used an approximation to these forcing terms because they did not witness a noticeable difference with the proper MRT projection. This is *not* the case when central moments are used: the resulting increased accuracy is significant as our results will demonstrate.

3.3 Phase field evolution

In order to simulate a mixture of two immiscible, incompressible fluids with respective density ρ_H and ρ_L and respective viscosity ν_H and ν_L (where the subscripts H and L are used to differentiate the high-density fluid from the low-density fluid in our exposition), the diffuse phase field model [Kendon et al. 2001; Badalassi et al. 2003] proposes to introduce a time-varying phase function $\phi(x,t) \in [0,1]$ as an order parameter to identify the regions occupied by the two fluids through:

$$\phi(x,t) = \begin{cases} 1 & \text{inside high-density fluid,} \\ 0 & \text{inside low-density fluid,} \end{cases} \tag{10}$$

with a smooth, thin transition layer of thickness ξ between the two fluids. A sharp location of the interface may thus be defined as the surface where the phase field takes the value ½, but the purposely-continuous transition of the phase field allows to substitute boundary conditions at the interface by a partial differential equation for the evolution of the phase function to reproduce the correct interfacial dynamics. The density field ρ is then defined in space and time as:

$$\rho(\mathbf{x},t) = \rho_L + (\rho_H - \rho_L)\phi(\mathbf{x},t). \tag{11}$$

The viscosity ν is similarly expressed in space and time as:

$$\frac{1}{\nu(\mathbf{x})} = \frac{1}{\nu_L} + \phi(\mathbf{x})\left(\frac{1}{\nu_H} - \frac{1}{\nu_L}\right). \tag{12}$$

Equation of motion. A phase field model is usually constructed based on a free energy for the system, often involving a Ginzburg-Landau double-well potential describing the free energy density of the bulk of each phase and an interfacial Dirichlet energy. A *conservative* form of such an interface evolution was proposed in [Geier et al. 2015], resulting in a modified Allen-Cahn equation written as:

$$\frac{\partial \phi}{\partial t} + \nabla\cdot(\phi\mathbf{u}) = \nabla\cdot\left[M\left(\nabla\phi - \frac{4}{\xi}\phi(1-\phi)\mathbf{n}\right)\right], \tag{13}$$

where $\mathbf{n} = \nabla\phi/|\nabla\phi|$ represents the unit vector field aligned with the gradient of the phase function (thus orthogonal to the isovalue surfaces of the phase field), while the mobility parameter M controls the degree of interface splitting, i.e., small M implies greater ease for the interface to split due to a smaller influence of the interfacial Dirichlet energy. This equation is in divergence form, with the first divergence term making the phase field advected by the macroscopic fluid velocity $\mathbf{u}$, while the right-hand side term is the sum of a

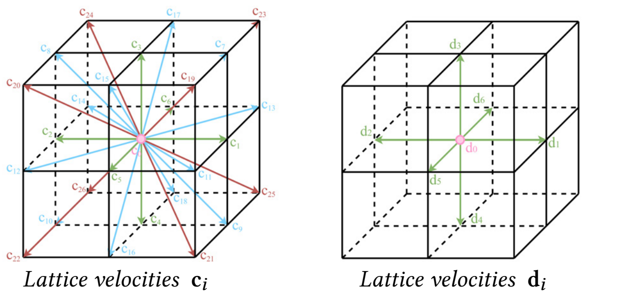

Fig. 6. **Lattice structures.** Velocity and pressure fields are computed via a D3Q27-based distribution function g (left, see Sec. 3.2), while the phase field indicating the density field (and thus, the interface between two fluids) is encoded via a D3Q7-based distribution function h (right, see Sec. 3.3)

diffusive flux density $M\nabla\phi$ and a separation flux density that favors a predefined interface profile in 3D [Geier et al. 2015] of the form:

$$\phi(\mathbf{x},t) = \frac{1}{2}\left[1 - \tanh\left(\frac{2d_\phi(\mathbf{x})}{\xi}\right)\right]. \tag{14}$$

where $d_\phi(\mathbf{x}) = \pm|\mathbf{x} - \bar{\mathbf{x}}|$ is the signed distance of any point $\mathbf{x}$ to its orthogonal projection $\bar{\mathbf{x}}$ on the interface with $\phi(\bar{\mathbf{x}}) = 1/2$, while ξ controls the interfacial thickness.

LBM discretization. In order to integrate Eq. (13) in time, we also base our approach on statistical physics to simplify the coupling between phase and fluid motion, but we use a D3Q7 lattice to dramatically reduce memory consumption compared to [Li et al. 2021]. Therefore, while we employ the same regular grid for the spatial discretization as for the fluid simulation, a set of only 7 lattice velocities in 3D ($\mathbf{d}_0, \ldots, \mathbf{d}_6$, see Fig. 6) are used, with associated quadrature weights $\{w_i^d\}_i$ and a speed of sound $d = 1/2$. A distribution function h is then used to find the phase field as its *zeroth-order moment*, i.e., we can evaluate the phase field at any node and any time through:

$$\phi(\mathbf{x},t) = \sum_{i=0}^{i=6} h_i(\mathbf{x},t). \tag{15}$$

Solving the phase-field equation. The phase field equation (13) can then be solved using the distribution function h whose components satisfy the lattice Boltzmann equations [Gruszczyński et al. 2020]:

$$h_i(\mathbf{x}+\mathbf{d}_i, t+1) - h_i(\mathbf{x},t) = \Omega_i^h(\mathbf{x},t) + H_i(\mathbf{x},t), \tag{16}$$

where the only force $\mathbf{H}$ acting on the phase field corresponds to the conservative term in the right-hand side of the phase equation [Wang et al. 2016], counteracting the diffusion of the phase-field and driving it to reach the predefined hyperbolic-tangent interface profile (Eq. (14)) in the equilibrium state. i.e.,:

$$\mathbf{H} = d^2\,\frac{4\phi(1-\phi)}{\xi}\,\frac{\nabla\phi}{|\nabla\phi|}. \tag{17}$$

The phase gradient $\nabla\phi$ is evaluated using the second-order weighted estimate from [Li et al. 2021], i.e., we use:

$$\nabla\phi = \frac{2}{3}\nabla^{[a]}\phi + \frac{1}{3}\nabla^{[b]}\phi, \tag{18}$$

using the two rotationally-symmetric gradient approximations:

$$\nabla^{[a]}\phi(\mathbf{x}) = \textstyle\sum_i 3w_i^c\mathbf{c}_i\phi(\mathbf{x}+\mathbf{c}_i),$$

$$\nabla^{[b]}\phi(\mathbf{x}) = \textstyle\sum_i 3w_i^c\mathbf{c}_i\frac{4\phi(\mathbf{x}+\mathbf{c}_i)-\phi(\mathbf{x}+2\mathbf{c}_i)}{2}.$$

This phase-field force is projected in distribution space via:

$$H_i(\mathbf{x},t) = w_i^d\,\boldsymbol{d}_i\cdot\mathbf{H}(\mathbf{x},t)/d^2. \tag{19}$$

Collision operator. While the forcing terms of Eq. (16) do not need any particular treatment due to their limited amplitude, a high accuracy for the collision term Ω_i^h is key to the stability of the phase field given the purposely-chosen low-order lattice. We thus employ a high-order (non-orthogonal) central-moment MRT model as we did for the fluid velocity. Concretely, the collision terms are evaluated as the component of a vector $\mathbf{\Omega}^h = (\Omega_0^h, \Omega_1^h, ..., \Omega_6^h)$ through

$$\mathbf{\Omega}^h = -\mathbf{M}_d^{-1}\mathbf{S}_d\mathbf{M}_d(h_i - h_i^{\text{eq}}), \tag{20}$$

where the projection matrix $\mathbf{M}_d$ is described in Sec. E of the supplemental material, the diagonal relaxation matrix $\mathbf{S}_d$ is set as $\text{diag}(1, 1/(4M+½), 1/(4M+½), 1/(4M+½), 3/2, 3/2, 3/2)$, while the phase-field equilibrium distribution functions are expressed as:

$$h_i^{\text{eq}} = \phi\,\Gamma_i^h - H_i/2, \tag{21}$$

based on trapezoidal integration as described in Sec. A of the supplemental material, where $\Gamma_i^h = w_i^d\left(1 + \frac{\boldsymbol{d}_i\cdot\boldsymbol{u}}{d^2}\right)$. The first four relaxation times in $\mathbf{S}_d$ are physically imposed by the interface mobility; the two highest-order relaxation rates were chosen experimentally to maximize accuracy and stability. Other choices, like a regression-based selection [Li et al. 2020], could be used as well.

Coupling forces. Finally, the fluid simulation from Sec. 3.2 requires an evaluation of forces allowing the proper coupling of the two phases, which we can now describe based on the phase field. For the surface tension force $\mathbf{F}_s$, we follow [Fakhari et al. 2017a] in using:

$$\mathbf{F}_s = \chi_\phi \nabla\phi, \tag{22}$$

where the gradient is computed as in Eq. 18, while we use the definition of the chemical potential χ_ϕ proposed in [Jacqmin 1999]:

$$\chi_\phi = 4\beta\phi(\phi-1)(\phi-\tfrac{1}{2}) - \kappa\nabla^2\phi, \tag{23}$$

where the second-order Laplacian of the phase is approximated as:

$$\nabla^2\phi(\mathbf{x}) = 6\sum_i w_i^c(\phi(\mathbf{x}+\mathbf{c}_i) - \phi(\mathbf{x})). \tag{24}$$

For the pressure force $\mathbf{F}_p$, we use the expression:

$$\mathbf{F}_p = -p^*\,c^2\,\nabla\rho, \tag{25}$$

while the viscosity force $\mathbf{F}_\mu$ is expressed as:

$$\mathbf{F}_\mu = \nu[\nabla\mathbf{u} + (\nabla\mathbf{u})^T]\nabla\rho, \tag{26}$$

where ν is the spatially-varying kinematic viscosity from Eq. (12) while the gradient of density is easily rewritten as a rescaled gradient of ϕ using Eq. (11). Note that these two last coupling forces, proposed by [Zu and He 2013], use terms like p^* and $\mathbf{u}$, thus depend on the zeroth and first moments of the other distribution functions g_i.

3.4 Stable boundary treatment

While the description given thus far simulates a multiphase fluid flow very reliably over a large range of Reynolds numbers and density ratios, the treatment of obstacles is notorious for being particularly difficult to handle robustly: a poor treatment of the interaction between obstacles, fluid, and phase field can ruin stability, preventing the simulation of anything but trivial examples with grid-aligned boundaries. We present a series of measures that we have adopted, both for numerical stability and for efficiency of implementation, to ensure fool-proof stability and parallel efficiency while maintaining accuracy at an acceptable level for visual realism, both for thick and thin obstacles.

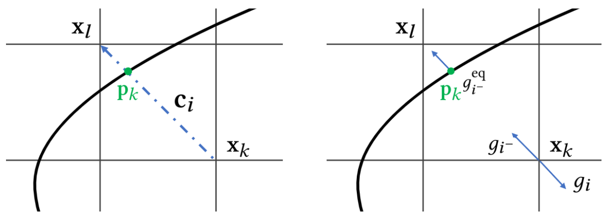

Fig. 7. **Boundary treatment.** A grid node $\mathbf{x}_k$ is a cut-cell node if at least one ray cast from it along a link intersects an obstacle boundary (left). The bounce-back from an obstacle used the equilibrium distribution function based on the fixed obstacle (right).

3.4.1 *Detecting and setting up cut-cells.*

Because we need to alter the updates of distribution functions near obstacles, we preprocess the simulation scene by first identifying cut cells, i.e., cells of the regular grid (used to discretize space) that straddle the boundary of at least one of the embedded obstacles. Contrary to [Lyu et al. 2021] that proceeds with samples spread over boundaries, we find the lattice links which cross a boundary directly through ray intersections with the triangle mesh(es) defining the obstacle boundaries in a preprocessing phase (see Fig. 7, left). Note that ray intersections allow us to deal with both thin shell obstacles and thick obstacles seamlessly. Any cell containing one of these straddling links is then tagged as a "cut cell". A dedicated data structure is then assembled for each cut-cell node (i.e., each node belonging to one or more cut cells), so that we can efficiently process all such nodes adjacent to boundaries to reduce warp divergence due to branching for a massively-parallel implementation on GPU. For each cut-cell node $\mathbf{x}_k$, we store the indices of its boundary-crossing links (among the 26 links, which also include the 6 links used for the phase field, see Fig. 6), as well as the closest projection $\mathbf{p}_k$ of this grid node onto the embedded boundary (computed by finding the closest triangle first through a GPU-based K-d tree, then finding the closest point of $\mathbf{x}_k$ in that triangle). Our boundary handling can then be efficiently applied for all cull-cell nodes based on these precomputed elements. Note that our treatment can handle even very fine geometric features (see Fig. 3), while sample-based approaches such as [Lyu et al. 2021] would require an inordinate sample density to avoid leakage issues.

3.4.2 *Hybrid bounce-equilibrium streaming near boundaries.*

Integrating in time Eqs. (3) and (16) for the distribution functions g and h involves a streaming stage, which needs to account for the "reflection" of the fluid or phase particles against boundaries. A prominent approach to handling the streaming of distribution functions near boundaries is to use the bounce-back method [Ladd 1994], which performs a direction reversal of streaming on crossing links: the distribution value g_i (resp. h_i) along the lattice direction $\mathbf{c}_i$ is transferred to the distribution value g_{i^-} (resp. h_{i^-}) for the direction $\mathbf{c}_{i^-} = -\mathbf{c}_i$ at the same node — with also a momentum transfer for moving boundary, which we do not consider here. This low-order approach is perfectly adequate for the phase field, as we only use the zeroth-order moment of h. However, for g, this bounce-back process is well-known to create spurious pressure fluctuations for large fluid velocity. These pressure artifacts trigger numerical instability, preventing simulations that involve complex boundaries or turbulent multiphase flows; see Fig. 8(a) for instance, where high-frequency artifacts in the velocity field of a dam-break simulation

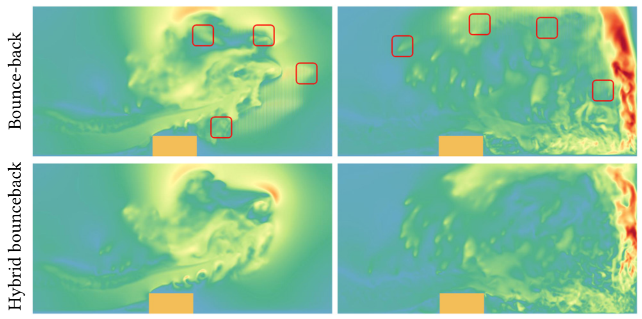

Fig. 8. **Hybrid bounce-back.** On the example from Fig. 5, handling obstacles via bounce-back generates oscillations (see red boxes) as the vector field magnitude demonstrates on these two frames (top); Using hybrid bounce-back in cut-cells removes this issue while preserving details of the flow.

can be seen extending up to the domain boundary. Higher-order accuracy has been proposed through *interpolated* bounce-backs [Kao and Yang 2008; Yin and Zhang 2012], but they involve a larger set of nodes near solid boundaries, which makes it more difficult to handle narrow boundary gaps. In our visual computing context, we stay with a first-order approximation so that we can keep our boundary handling treatment limited to cut cells for efficiency purposes, but we hybridize it with the use of the equilibrium distribution of the obstacle to significantly reduce pressure overshoots.

Our approach can be understood as a modified version of the boundary treatment proposed in the single-phase flow simulation method of [Filippova and Hänel 1998]. In this work, the authors noted that the incoming distribution function coming from the boundary towards a fluid node along a boundary-crossing link can be partially approximated by the equilibrium distribution function of a point at the intersection of the link and the boundary, see Fig. 7(right). This approximation, *also* first-order accurate, is far less subject to numerical fluctuations and thus to instability as they directly rely on the relaxed state for which truncation errors have very little impact. In our context of fluid simulation with static obstacles, we can use a *Neumann-pressure* and *zero-velocity* boundary condition imposed on the obstacle to evaluate local values of $\hat{\mathbf{f}}^{\mathrm{eq}}$. Because our tests show improved stability with this approach compared to the original bounce-back streaming when the pressure near a boundary is small, we blend both approximations based on the amount of local pressure to promote stability while maintaining accuracy: it amounts to adapting the relaxation rate to reliably prevent overshoots. In practice, this means that we use the following streaming step for all cut-cell nodes:

$$g_i(t+1,\mathbf{x}_k) = (1-\alpha_k)\, g_{i^-}(t,\mathbf{x}_k) + \alpha_k\, \hat{f}_{i^-}^{\mathrm{eq}}\big|_{\mathbf{u}=0,p^*=p^*(\mathbf{x}_k)}, \quad (27)$$

where the blending coefficient is set to $\alpha_k = 3\|p^*_{\mathbf{x}_k}\|$ to safely suppresses pressure fluctuations near obstacles, see Fig. 8. Note that one can also use an equilibrium distribution based on $p^*=0$ to further dampen any spurious pressure artifact due to violent splashes on walls or obstacles, but now at the cost of decreased accuracy.

3.4.3 *Phase gradient near boundary and wetting control.*

Besides its obvious relevance to the phase field motion as discussed in Sec. 3.3, approximating the gradient $\nabla\phi$ of the phase field is required in several force terms such as the surface, pressure, and viscosity forces,

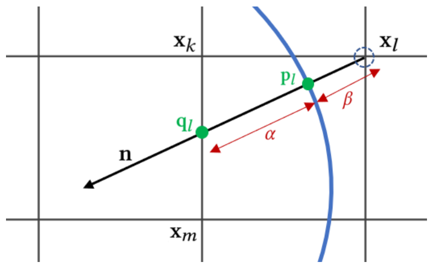

Fig. 9. **Wetting boundary condition for phase field.** For a cut-cell node $\mathbf{x}_k$, a "ghost" phase field value for the grid node $\mathbf{x}_l$ on the other side of the nearby obstacle boundary is found by enforcing the normal directional derivative imposed by the wetting boundary condition.

respectively, $\mathbf{F_s}$, $\mathbf{F_p}$ and $\mathbf{F}_\mu$. When evaluation is needed away from boundaries (and thus, not on cut-cell nodes), we use the weighted rotationally-symmetric evaluation proposed in [Li et al. 2021] as described in Eq. (18). However, the presence of obstacles nearby renders this approximation inappropriate, and thus, unstable in practice: it uses values of the phase field on the *other side* of the boundary, failing to account for the presence of the obstacle and for the implied Neumann condition. Using sided finite differences would not be able to impose the proper boundary conditions either. We thus reverse to the simple *centered finite difference* approximation of the gradient where *extrapolated (ghost) values* of the phase field ϕ are used for the nodes on the other side of a boundary. Fig. 9 shows a 2D example where to calculate $\nabla\phi$ on the fluid node $\mathbf{x}_k$, we need to deduce a ghost value of ϕ on the node $\mathbf{x}_l$ (which is on the other side of the boundary) so as to enforce boundary conditions. While one usually imposes zero Neumann boundary condition on the phase field of the form $\mathbf{n}\cdot\nabla\phi=0$ where $\mathbf{n}$ is the unit vector normal pointing outwards from the boundary, capturing the degree of wetting (the ability of a liquid to maintain contact with a solid surface, resulting from the balance between adhesive and cohesive intermolecular forces) requires to impose an arbitrary *contact angle.* A contact angle of less than 90° between a liquid and a boundary implies that the fluid will tend to spread over a large area (hydrophilic behavior), whereas a greater contact angle indicates that the fluid will try to minimize contact with the boundary, forming instead compact liquid droplets (hydrophobic behavior). To impose a specified contact angle θ at a solid boundary, the following wetting condition [Fakhari and Bolster 2017] has been formulated, stating that one should enforce at every point $\mathbf{p}$ on the boundary:

$$\mathbf{n}(\mathbf{p})\cdot\nabla\phi(\mathbf{p}) = \Theta\,\phi(\mathbf{p})\,(1-\phi(\mathbf{p})) \quad \text{for} \quad \Theta = -\sqrt{2\beta/\kappa}\cos\theta.$$

Given a desired contact angle θ, we can now find the "ghost" value $\tilde{\phi}(\mathbf{x}_l)$ that imposes this condition at point $\mathbf{p}_l$, see Fig. 9. Indeed, from the closest point $\mathbf{p}_l$ of $\mathbf{x}_l$ (precomputed as explained in Sec. 3.4.1), we first find the intersection point $\mathbf{q}_l$ on the line $\mathbf{x}_k\mathbf{x}_m$ in 2D, and we deduce its phase field from $\phi(\mathbf{x}_k)$ and $\phi(\mathbf{x}_m)$ through barycentric coordinates — the same numerical approach is used in 3D. We then use the normal derivative imposed by the wetting condition to directly extrapolate the ghost value we seek, leading to:

$$\tilde{\phi}(\mathbf{x}_l) = \frac{\alpha+\beta}{2\alpha^2\Theta}\left(1+\alpha\Theta-\sqrt{(1+\alpha\Theta)^2-4\alpha\Theta\phi(\mathbf{q}_l)}\right) - \frac{\beta}{\alpha}\phi(\mathbf{q}_l), \quad (28)$$

where α and β are shown in Fig. 9. When $\theta=90$ (neutral wetting, i.e., zero Neumann condition), one uses simply: $\tilde{\phi}(\mathbf{x}_l)=\phi(\mathbf{q}_l)$.

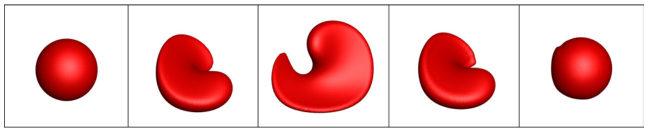

Fig. 10. **Spherical interface in shear flow.** We test our D3Q7 LBM-based integration of the phase field by passively advecting a spherical interface in a prescribed velocity field, see Eq. (29); snapshots obtained at $t = 0$, $t = T/4$, $t = T/2$, $t = 3T/4$, and $t = T$, from left to right.

3.4.4 Controlling norms. Another safety measure to enforce stability concerns the norms of the gradient of ϕ and of forces. In the previous paragraph, we described how the phase gradient is computed so as to account for the boundary presence and the extended Neumann condition for wetting. To further combat possible numerical inaccuracies, we leverage the property that the profile of the phase field should remain close to the form of Eq. 14. Thus, given the macroscopic phase field value ϕ at a cut-cell node, we know that its ideal gradient norm should thus be $|\nabla\phi^*|=4\phi(1-\phi)/\xi$. If we find out that the actual vector value of $\nabla\phi$ has a norm larger than this ideal norm by 60%, we simply rescale the gradient vector. In practice, our tests show that this rescaling is triggered only in less than 5% of the cut-cell nodes on average — but this normalization often prevents blow-ups. A last modification we perform is that if one of the force magnitudes is more than 60 times larger than gravity and that the density of the light fluid is very low, we threshold forces where the phase ϕ is below 0.1. In this specific case (which happens for strong splashes like in Fig. 4), instead of using the adaptive temporal method proposed in [Li et al. 2020], we can reduce the magnitude of the pressure, viscosity, and surface tension forces by 40% without affecting the motion: thresholding large "impulse" forces does not affect the visual behavior significantly since it affects only the light fluid; this last safety measure improves stability without having to reduce time steps.

3.5 Discussion

Now that we have reviewed the details of our approach, we can discuss its most salient differences with the recent work of [Li et al. 2021]. First, our LBM formulation is different: we use a velocity-based lattice Boltzmann formulation instead of their moment-based formulation. This has already important consequences: since density is discontinuous in multiphase flows, the moment $\rho\mathbf{u}$ is much more prone to face instability at the interface [Kumar et al. 2019]; moreover, the forces $\mathbf{F}_p$ and $\mathbf{F}_\mu$ are evaluated calculated directly instead of being included implicitly in the distribution function, making the coupling between flow and interface more explicit. Additionally, the higher-order collision model we use for both g and h brings numerical accuracy whose induced stability allows for more turbulent multiphase simulation, with density ratios of up to 1000 for a thinner phase interface than in [Li et al. 2021]. Another key difference is our treatment of boundaries which can handle complex obstacles that are either thick or thin, without suffering from possible leakages due to our proper one-sided approach to bounce-back. Memory requirements are also drastically reduced by our use of D3Q7 lattice for the phase field vs. the original D3Q27 lattice — we will also note in the implementation details listed in Sec. 4.2 that we employ the "swap" strategy from [Latt 2007a], bringing a memory usage reduction to around 60%. Moreover, we will leverage

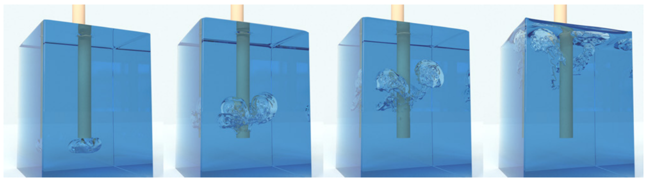

Fig. 11. **Blowing air into a viscous liquid.** Using the same setup as Fig. 2 but with a forty times lower Reynolds number and using a single burst of air, the air forms larger and rounder bubbles as expected.

the recent GPU parallel optimizations of [Chen et al. 2021], which will result in a typical four-fold speedup for resolutions around $400 \times 200 \times 200$. Finally, all the safety measures listed in Sec. 3.4 to improve stability allow for larger effective time steps (typically, 50% larger), thus far improved efficacy; note that a reduced physical time step would remove the need for these numerical measures, but for computer graphics applications, efficiency (even at the cost of minor inaccuracy) is paramount.

4 RESULTS AND LIMITATIONS

We now cover the numerical experiments we conducted with our solver, to illustrate its accuracy, efficiency, and robustness via benchmark tests, complex scenarios, and comparisons with existing solvers or real-life fluid motions.

4.1 Benchmark tests

We begin our tests with three classical benchmark examples.

Advection of phase field in prescribed velocity field. In order to test our kinetic phase field solver alone, we perform a passive advection of spherical shape in a prescribed vector field forming a typical shear flow as proposed in various previous works [Geier et al. 2015; Liang et al. 2018; Zu and He 2013]. The phase field (with $M = 0.0035$) is initialized on a regular grid of size 100×100×100 so that the interface forms a sphere of radius $R = 20$ centered at $\mathbf{x}_0 = (30, 30, 50)$. The predefined vector field is defined on a grid node (i, j, k) through:

$$\mathbf{u} = 0.02 \begin{pmatrix} \cos\left[\frac{i-50}{100}\pi\right]\left(\sin\left[\frac{k-50}{100}\pi\right] - \sin\left[\frac{j-50}{100}\pi\right]\right)\cos\left(\pi\frac{t}{T}\right) \\ \cos\left[\frac{j-50}{100}\pi\right]\left(\sin\left[\frac{i-50}{100}\pi\right] - \sin\left[\frac{k-50}{100}\pi\right]\right)\cos\left(\pi\frac{t}{T}\right) \\ \cos\left[\frac{k-50}{100}\pi\right]\left(\sin\left[\frac{j-50}{100}\pi\right] - \sin\left[\frac{i-50}{100}\pi\right]\right)\cos\left(\pi\frac{t}{T}\right) \end{pmatrix}. \tag{29}$$

We visualize the phase field evolution by extracting its isovalue ½ in Fig. 10 Note that the sphere is supposed to be most deformed at $t = T/2$, before returning to its initial position after a period T. The L_2 relative error ε with respect to the initial value of the phase field after a full period, defined as

$$\varepsilon = \sqrt{\frac{\sum_i [\phi(\mathbf{x}_i, T) - \phi(\mathbf{x}_i, 0)]^2}{\sum_i \phi^2(\mathbf{x}_i, 0)}}, \tag{30}$$

is listed in Tab. 1 along with other D3Q7-based [Fakhari et al. 2019] and D3Q19-based [De Rosis and Enan 2021] methods to evaluate its accuracy. We see that we improve upon the existing D3Q7 phase field by 33%, and reach an error equivalent to the 27-link lattice — but with the reduced memory size of the 7-link lattice.

Droplet oscillation. Fig. 12 shows a three-dimensional elliptical droplet (density ratio 1000) which, due to the surface tension of the interface, oscillates around its spherical equilibrium shape. The

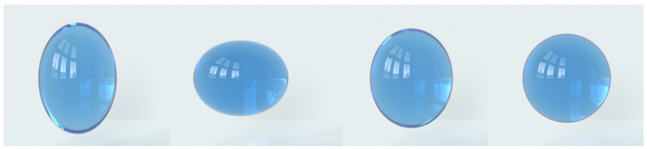

Fig. 12. **Droplet oscillation.** Under the effect of surface tension, the shape of a droplet simulated with our LBM-based phase field oscillates (left to right) around its circular equilibrium shape, eventually coming to rest.

Table 1. **Relative error** of our D3Q7 LBM for the experiment of Fig. 10, compared to [Fakhari et al. 2019] and [De Rosis and Enan 2021].

Model	ε
D3Q7 [Fakhari et al. 2019]	0.0754,
D3Q19 [De Rosis and Enan 2021]	**0.0490**
D3Q7 (Ours)	0.0500

fluids inside and outside the droplet are slightly viscid, so the droplet eventually converges to a sphere.

Three-dimensional Rayleigh Taylor instability. We also tested our solver on the classical Rayleigh-Taylor instability, where a heavy fluid is placed on top of a lighter fluid. The interface between the immiscible fluids should deform and exhibit Kelvin–Helmholtz instabilities as the heavier fluid moves down and the lighter material is displaced upwards. We simulated this canonical case on a grid of size 64×256×64, and for an interface width set to $\xi=5$. In order to compare our results to existing published examples, we initialize the two immiscible fluids as follows:

$$\phi(\mathbf{x}) = \begin{cases} 1, & \text{if } y > 128 + 3.2\left[\cos\left(2\pi\frac{x}{64}\right) + \cos\left(2\pi\frac{z}{64}\right)\right], \\ 0, & \text{otherwise,} \end{cases} \tag{31}$$

and set the various dimensionless governing parameters as follows: Atwood number $At = 0.5$, capillarity number $Ca = 960$ and Péclet number $Pe = 256$. The downwards vertical gravity $\mathbf{g}$ is set to have a magnitude $|\mathbf{g}| = U_0^2/64$ for a reference velocity $U_0 = 0.04$, and the gravitational force is defined as $\mathbf{F_g} = (\rho - \bar{\rho})\mathbf{g}$. Finally, the viscosity of both fluids is set to $\nu = 64\sqrt{64|\mathbf{g}|}/Re$. We run this experiment for three different Reynolds numbers (Re = 256, 3,000, and 30,000) as shown in Fig. 13, where the vertical walls are set to have periodic boundary conditions. As time goes on, the heavy and light fluids intertwine, forming mushroom-shaped downward spikes as the heavy fluid grows into the light fluid, and bubbles as the light fluid grows into the heavy fluid. Using time scale $T = \sqrt{64/|\mathbf{g}|}$, a first roll-up of the heavy fluid occurs around $t = 2T$ at the four edges of the fluid field box. This roll-up interface becomes stronger in time. While there are no analytical solutions with which to compare, our results are consistent with published data for $Re = 256$ [He et al. 1999]: the accuracy of our results, measured by the vertical positions of the lowest point of the downward spike at different times, is shown in Tab. 2 and compared to previous works [De Rosis and Coreixas 2020; Lee and Kim 2013]. Note that with larger Reynolds numbers, we observe more vortical features near the four sides of the box as expected; but we were unable to compare our spike heights on these more inviscid cases despite results recently published in [De Rosis and Enan 2021] as we realized that their code is missing a division by ρ for the gravitational acceleration, rendering their heights in this turbulent case inconsistent.

Table 2. **Three-dimensional Rayleigh–Taylor instability:** vertical positions of the interface spike normalized by the domain width at representative times compared to two other multiphase flow LBM solvers.

t/T	Ours	[De Rosis and Coreixas 2020]	[Lee and Kim 2013]
0.0	1.875	1.897	1.904
0.5	1.843	1.897	1.869
1.0	1.757	1.753	1.776
1.5	1.617	1.592	1.618
2.0	1.421	1.381	1.396

4.2 Implementation details

While the overall structure of our solver is relatively simple (see the pseudocode in Alg. 1), a number of important implementation points deserve further discussions.

Memory usage. Compared to recent LBM-based works in computer graphics, we reduce memory usage significantly. First, we employ the "swap" approach proposed in [Latt 2007b] and extend it to our GPU implementation. The basic idea is that the neighboring data copy in the streaming step is replaced by a swap operation with another existing variable, where no temporary memory is needed — thus reducing the amount of memory used for the distribution functions g and h by a factor two. Moreover, using a D3Q7 lattice further reduces memory compared to the approach of [Li et al. 2021]. In the end, we save 73 variables per grid node in 3D compared to this latest work, for a total memory usage reduction of around 60%, see Tab. 3. Furthermore, we use a cut-cell buffer array to store precomputed closet points and intersection link flags as described in Sec. 3.4.1. When dealing with a cut-cell node, we access the necessary values from this cut-cell buffer array directly through an index array that saves the cut-cell node index in the cut-cell buffer array. Hence, memory consumption is limited to a size proportional to the area of the obstacle boundaries, which remain small in all of our examples.

GPU-based implementation. Our multiphase flow solver uses only local computations, and can thus be easily implemented in a massively-parallel fashion on a GPU. We adopt the structure-of-arrays (SoA) memory layout to improve cache utilization — and thus improve performance — as recommended for single-phase flow simulation in [Chen et al. 2021]. Moreover, we also employ the LU decomposition of M_c^{-1} to speed up projections in and out of central-moment space as proposed in [Lyu et al. 2021]. For g^{eq} in Eq. (4), we use an analytical expression that we derived directly in central-moment space (see Sec. F of the supplemental material) which saves further collision computations. Finally, prefetching distribution functions g_i from global to GPU shared memory significantly speeds up performance as threads can repeatedly access data much faster.

Efficiency. While [Li et al. 2021] proposed a parallel implementation of their approach, we leverage all code accelerations mentioned in [Chen et al. 2021] which drastically improved efficiency. Due to our improved stability, our new approach also supports 25% larger

Table 3. **Memory usage.** Number of 32-bit floats per grid node in 3D.

Method	$(\rho, \mathbf{u}, p)$	cut-cell	flag	$\mathbf{g}$	$\mathbf{h}$	ϕ	$\mathbf{F}$	$\mathbf{n}$	total
Our method	5	1	1	27	7	1	3	3	48
[Li et al. 2021]	5	—	1	54	54	1	3	3	121

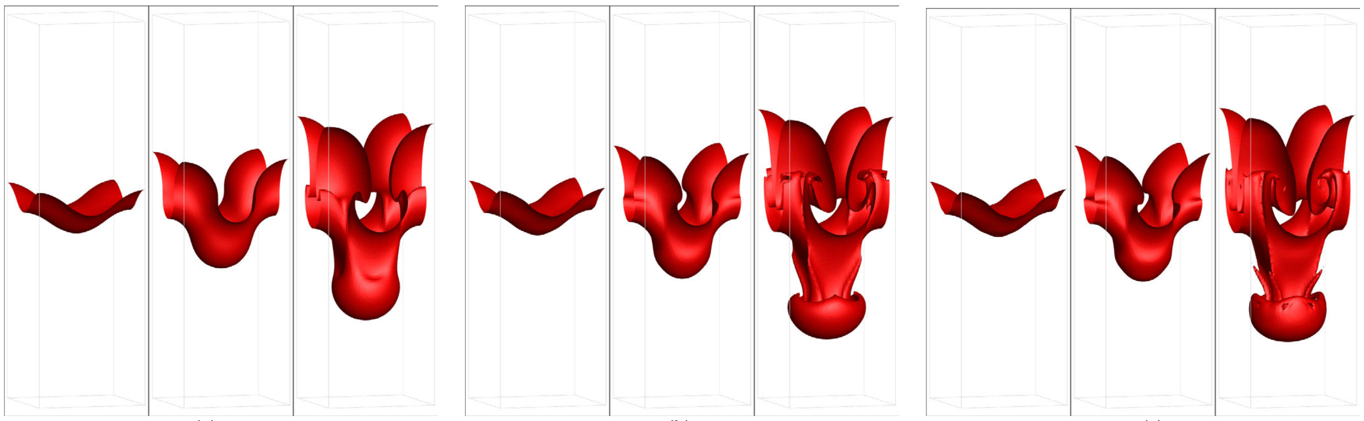

Fig. 13. **Three-dimensional Rayleigh–Taylor instability:** Evolution of the interface between a heavy fluid on top and a light fluid at the bottom for various Reynold numbers, (a) Re = 256, (b) Re = 3, 000, and (c) Re = 30, 000. Each example is shown at times at $t = T$, $t = 2T$, and $t = 3T$.

time steps, meaning that our solver can use fewer iteration numbers for the same simulation sequence. As a consequence of all the differences we summarized in Sec. 3.5, we show in Fig. 14 that our approach is faster: based on a basic 3D dam break with obstacle from Fig. 8 and using the same GPU card, we improve upon their recent solver by almost a factor seven for low resolutions, and nearly four-fold for resolutions above 400×200×200.

4.3 Simulation results

All the examples of multiphase flows in the paper were run on a workstation equipped with a 16-core Intel(R) Xeon(R) Silver CPU, 128 GB of memory, and a NVIDIA RTX A6000 GPU. We used V-Ray [Chaos 2022] to render Figs. 1, 3, 22, and 17, while the other results were generated using our own renderer. We now discuss our results for various liquid simulations with complex, thin and/or thick static solids in order to illustrate the various properties of our solver. All of our simulation sequences involve a transparent container, and water splashes reaching these transparent walls thus form droplets that slide and merge, creating wetting patterns — the only exception being Fig. 17 where we set open boundary conditions for the phase field on the box to allow the water drops to fly off the domain in order not to obstruct the view. Note also that we use the Smagorinsky turbulence model to resolve flow details due to the limited grid resolution we use: an eddy viscosity ν' is evaluated

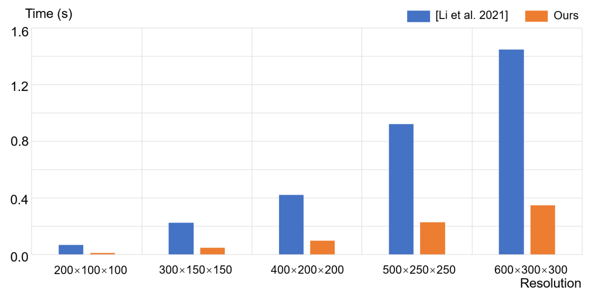

Fig. 14. **Computational times on dam break.** Compared to the recent work of [Li et al. 2021] (in blue), our solver (in orange) performs systematically faster — here, for the average iteration time of the dam break in Fig. 8.

according to [Geller et al. 2013] at each fluid node and added to the fluid viscosity ν.

ALGORITHM 1: Pseudocode of our kinetic two-phase flow solver.

```
CutCellsPreprocessing();                          ▷ Secs. 3.4 and 4.2
$t \leftarrow 0$;
while $t < T$ do
    PerformCollisionForFlow();                    ▷ Sec. 3.2
    PerformCollisionForPhase();                   ▷ Sec. 3.3
    StreamingInRegular&CutCells();                ▷ Secs. 3.2 & 3.4.2
    ComputeLow-OrderMoments();                    ▷ Sec.3.2
    CalculateForces();                            ▷ Secs. 3.2 & 3.4.3
    UpdateMacroscopicVariablesWithForces();       ▷ Eqs. (7)
    $t \leftarrow t + 1$;
end
```

Dam breaks. Testing the fall of a column of water and the subsequent breaking wave is a common test used in single- and multiphase flow solvers alike. We tested a simple 3D dam break with a rectangular box obstacle in Fig 5, showing that a low-order collision model like MRT creates spurious velocity artifacts (visualized via a 2D cut showing color-coded velocity magnitude), while the weighted MRT model used in [Li et al. 2021] leads to slightly noisy velocity field and does not capture very fine instability around the interface; our improved treatment using the central-moment space to compute the collision operator removes all these numerical issues. The same example is used again in Fig. 8 to demonstrate that the usual bounce-back approach to deal with streaming near obstacle boundaries can create severe artifacts, again in the form of oscillatory patterns which can make the simulation crash rather easily. Instead, our hybrid bounce-back prevents such artifacts. For this example, we also note that our new solver can support higher Reynolds numbers (by more than a factor six) and larger density ratios (by 25%) compared to [Li et al. 2021]. We also tested a dam break where a thin plate with a 30-degree incline is placed along the path; it exhibits a strong crown splash as the water hits the obstacle, see Fig. 4. This sort of thin crown water sheet cannot be captured by solvers with small to moderate numerical viscosity or

Table 4. **Profiling.** Timings of each key LBM step for [Li et al. 2021] and our algorithm. *Collision* refers to the computation of the collision operator for velocity and phase field, *Streaming* refers to the streaming update, *Moments* refers to the computation of the low-order moments, *Forces* represents the time needed to evaluate the various forces, and *Updates* refers to the macroscopic variables updates using forces.

Grid	**[Li et al. 2021]**				**Our solver**				
resolution	*Collision*	*Streaming*	*Moments*	*Forces*	*Collision*	*Streaming*	*Moments*	*Forces*	*Updates*
200×100×100	9.53ms	16.25ms	0.59ms	12.44ms	3.23ms	1.81ms	0.37ms	4.65ms	0.63ms
300×150×150	42.7ms	74.23ms	2.58ms	55.99ms	14.58ms	7.57ms	1.71ms	21.68ms	2.75ms
400×200×200	86.12ms	142.78ms	5.14ms	109.21ms	30.44ms	16ms	3.44ms	43.03ms	5.6ms
500×250×250	192.24ms	324.99ms	12.21ms	237.86ms	68.68ms	37.03ms	8.44ms	96.42ms	13.13ms
600×300×300	281.71ms	492.71ms	18.56ms	376.28ms	108.41ms	58.13ms	11.99ms	147.35ms	22.92ms

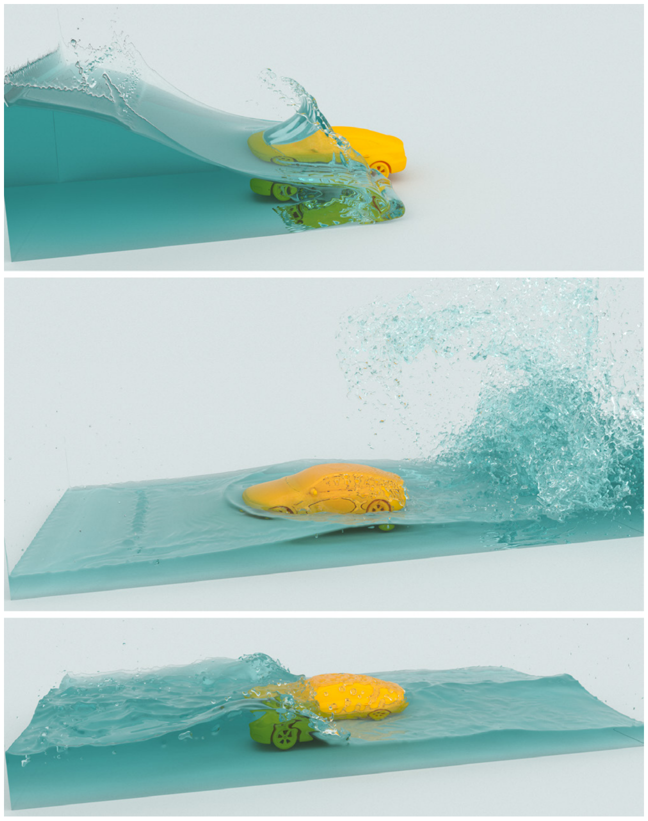
Fig. 15. **Dam break with thick obstacle.** A dam-break wave splashes into a car-shaped obstacle, forming an obvious crown splash and bubbles before subsiding. Wetting patterns on the car are also visible.

without a very accurate treatment of the collision operator. Bubbles and splashing foam form when the water reaches the right wall, while wetting patterns are also appearing as water spills onto the four transparent walls of the container. Note that our boundary treatment allows for such thin obstacles without instability issues, and does not suffer from the leakage problems that are typical of the immersed boundary method used in common LBM solvers. A different type of 3D dam break, this time with a thicker obstacle in the shape of a car, is demonstrated in Fig. 15. Clear wetting is visible on the car surface after the initial splash, while bubbles are also present as the flow subsides. Finally, we provide a last dam break example where we imitate the setup proposed in [Chen et al. 2020], where a dam break is set up to force the water to go through a fixed comb-like structure with narrow slits: we show in Fig. 16 that the liquid flows through the slits realistically, not only on the way down but also after the wave reflects back and forth in the container. We also demonstrate in Fig. 21 that the grid resolution for this simulation has obvious effects on how small the details of a flow can be captured; however, the overall motion is well captured even with five times fewer degrees of freedom. Chen et al. [2020] reported a computational time per frame of 90s. using a NVIDIA GeForce GTX 1080Ti; a slightly higher resolution takes 18s. for our solver on the same card, and only 7.5s. on our current machine.

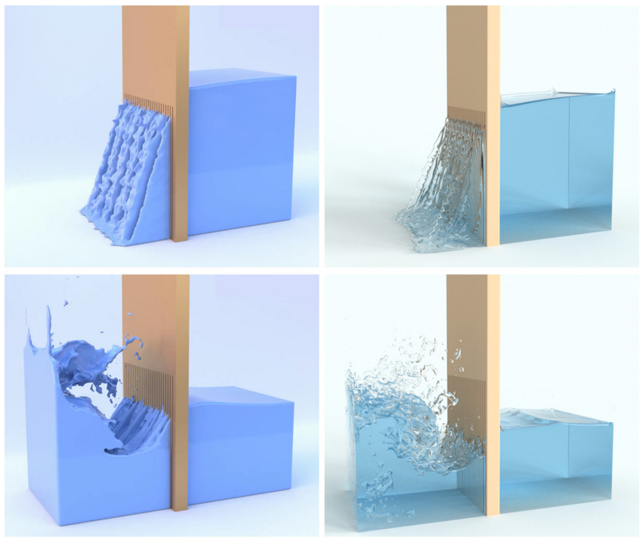
Fig. 16. **Comparison with [Chen et al. 2020].** While current single-phase flow solvers have demonstrated the ability to capture liquids flowing through narrow slits (left), our approach can not only simulate similar scenarios, but exhibits also bubbles and wetting.

Multiphase flows with complex obstacles. We also tested our solver robustness by simulating very complex obstacles as well. Fig. 3 shows a flow through a pumice-like porous rock, which is a very challenging scenario to simulate due to the large number of cut-cells. The water goes through all the irregular holes and tunnels, at times forming bubbles, while water filaments and drops appear at the bottom of the rock as the water exits. We also show an example of water going through a colander in Fig. 1: water is dropped suddenly inside the colander, and goes through the little interstices at high speed, creating a crown-like structure; some of the water hits the inside and regroups before going through the same interstices in a second wave. We also demonstrate robustness to complex objects in Fig. 18, where a water column comes crashing down on a loosely woven basket. Here again, the complexity of the shape and topology of the basket made out of thin structures is a great stress test for

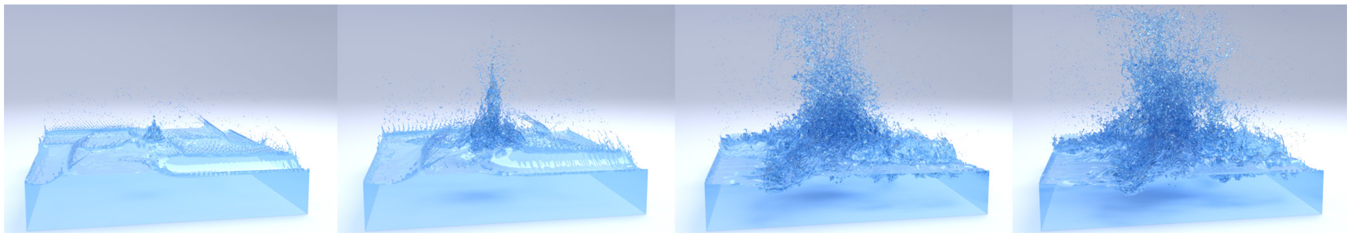
Fig. 17. **Air-driven flow.** Starting from a still pool of water, four horizontal strong jets (positioned near the top of the water surface) blow air, eventually driving the water up and forming a tornado-like behavior with significant splashing and spraying. For this example, we use an open boundary condition for the phase field (thus water droplets can fly off the domain) to avoid the visual obstruction that droplets on the walls would create.

Table 5. **Statistics for our results (FPS = 60).**

Figure	Resolution	Min./frame	ν_H	ν_L	ρ_H/ρ_L
1	600×500×600	2.3	0.0015	0.04	1000
2	400×800×400	5.3	0.0020	0.03	800
3	300×600×300	1.1	0.0015	0.06	1000
4	800×400×400	1.3	0.0010	0.08	1000
11	300×600×300	2.32	0.060	0.09	800
12	300×300×300	0.42	0.0080	0.01	1000
15	880×440×400	1.6	0.0010	0.08	1000
17	650×500×650	1.5	0.0019	0.03	1000
18	800×400×400	1.3	0.0010	0.08	1000
19 [Li et al. 2021]	600×300×300	1.84	0.0015	0.015	800
19 [Li et al. 2021]	800×400×400	5.83	0.0020	0.02	800
19 ours (middle)	600×300×300	0.47	0.0015	0.015	800
19 ours (middle)	800×400×400	1.36	0.0020	0.02	800
19 ours (bottom)	800×400×400	1.36	0.0080	0.02	800
19 ours (bottom)	800×400×400	1.36	0.0005	0.02	800
21 low-res	306×306×153	0.15	0.0003	0.01	1000
21 hi-res	512×512×256	1.05	0.0002	0.03	1000
22	800×320×400	3.6	0.0005	0.02	1000
23	750×420×400	3.0	0.0008	0.02	1000

boundary handling, and our solver is able to handle the simulation both robustly and efficiently.

Turbulent air-driven flows. Finally, we show a few examples of flows entirely driven by air motion, another stress test for any multiphase flow solver. Fig. 2 shows air being blown through a straw immersed in a water container. We impose a sine-wave speed on the inlet at the top of the straw; bubbles form below the straw as the air makes its way down the straw, then go up before bursting on top of the liquid surface. Fig. 17 shows, instead, a still pool over which four very strong air jets blow horizontally, lifting water up in the air, and generating a very turbulent motion. Note that this case uses open boundaries for the phase field, so that droplets that fly off are simply disappearing instead of crashing onto the transparent walls: this allows for a much more obvious visualization, unobstructed by a multitude of wetting patterns.

4.4 Comparisons

Finally, we provide a few comparisons, both with previous work and with real video-recorded flows. Fig. 16 was inspired by an example found in [Chen et al. 2020], where water is flowing through narrow slits. Not only we are able to handle this challenging scenario at a similar resolution, but we provide a full two-phase simulation instead of their single-phase result: the presence of bubbles and more realistic behavior of the flow ensues. We also compared our solver to the recent multiphase solver of [Li et al. 2021] in Fig. 5 as discussed earlier: while oscillations can be easily triggered when strong interaction with boundaries occur in their results, our approach has no such behavior — even for Reynolds numbers six times higher than

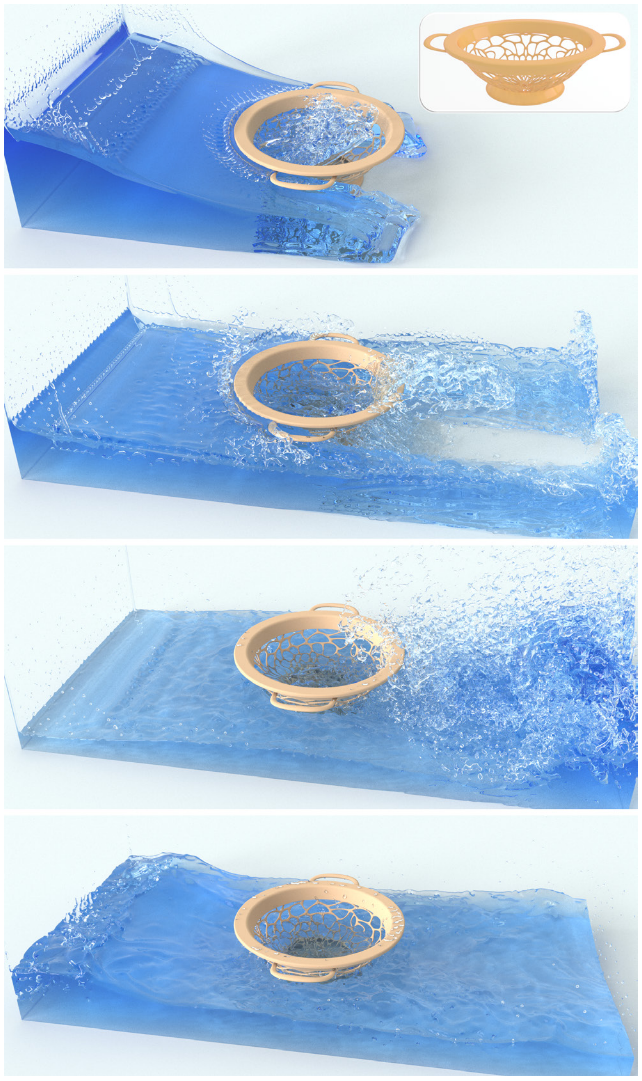
Fig. 18. **Basket case.** A basket, made out of complex, thin-walled structures (see top inset), is being immersed in water; as a water column comes crashing down on it, complex interactions between the flow and the object occur.

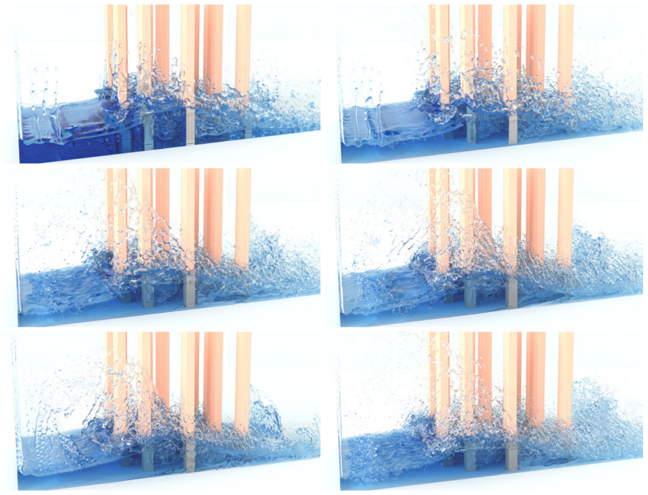

Fig. 19. **Comparison with [Li et al. 2021].** For a dam break flow against several pillars, (top) the approach of [Li et al. 2021] at 600×300×300 (left) and 800×400×400 (right) exhibits clear differences between the two resolutions due to excessive numerical viscosity. (middle) Our method, instead, shows better agreement and finer details. (bottom) Moreover, our approach can handle a far wider range of viscosity, from four times more (left) to four times less viscosity (right) which [Li et al. 2021] cannot simulate.

what they can support — and yet, as discussed in Sec. 4.2, our solver requires less memory and less time. Fig. 19 also demonstrates that the increased accuracy of our higher-order collision model leads to simulations much less sensitive to grid resolution, and allows for much less viscous fluids. We note additionally that the complex obstacles like Fig. 3 or thin shells like Figs. 1, 4, and 18 would not be possible to handle with their solver either: their boundary treatment prevents the use of thin shells, and obstacles have to be voxelized. As a last example, we show in Fig. 20 that a real colander exhibits a very similar behavior to what we showed in Fig. 1: while the shape of the strainer is a bit different (in particular, its curvature at the bottom) and it is placed on top of a glass, one can see a number of qualitative similarities, for instance, the same two-crown splashing behavior as the water falls through the interstices.

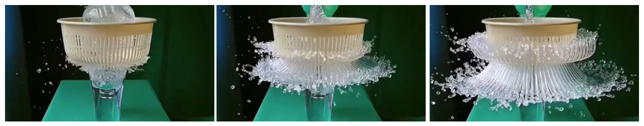

Fig. 20. **Real colander.** A real experiment of a water balloon being popped over a colander standing on a glass exhibits very similar features to what can be seen in Fig. 1 despite a different shape for the strainer: a two-crown splash appears as the water rushes through the interstices.

4.5 Limitations and future work

While our work overcomes a number of numerical hurdles in the simulation of binary fluids and allows for the simulation of everyday's phenomena where two fluids commingle, it is not without limitations. First, we remain limited in the highest Reynold number we can handle: extremely turbulent fluids would require higher accuracy to be captured precisely; thankfully, this limitation is not quite stringent for graphics as it is less relevant to our typical applications. Second, the ratio of fluid density is also limited — but here again, the fact that we can, at last, capture typical intricate air-water interactions is largely sufficient in most cases. Third, like other purely Eulerian methods, phase fields on very coarse grids can lead to disappearing droplets. Our conservative treatment limits this issue quite drastically, but rapid motions can still generate numerical diffusion resulting in a spurious shift of the fluid/air interface; a mix of phase field and particles (à la particle-levelset) could prevent this issue at the cost of constant reseeding of particles near the interface, offering a robust solution for coarse grids. Fourth, applying adaptive mesh refinement in 3D would also allow for a better capture of boundary layers than our current use of a fixed regular grid; an approach based on multiple grid resolutions as described in [Li et al. 2019] or [Fakhari et al. 2017a] could lead to much finer details, at the cost of memory and computational time increase. Fifth, it could be useful to extend our approach to many fluids instead of just two, as was done for Navier-Stokes solvers with an implicit encoding of the interface [Kim et al. 2010]; the recent work of Hu et al. [2020] could help in this regard. Sixth, our smeared interface is not able to properly handle bubble foam; mixing our work with an ad-hoc representation of foam [Kim et al. 2007; Busaryev et al. 2012] would be an interesting improvement. Finally, we only demonstrated the motion of binary fluids interacting with complex, but *static* obstacles. Offering stable one-way and two-way dynamic coupling in this multiphase flow context would open the door to many additional simulation scenarios. Note that it is, in fact, relatively easy to handle dynamic immersed solids: one needs to update the list of cut cells and cut-cell nodes at each time step, and could use the kernel-based penalty force (à la immersed boundary method) and dimension scaling proposed in [Li et al. 2020] to couple fluids and solids. This obviously incurs additional overhead, but allows us to seamlessly extend our work for non-trivial cases as shown in Figs. 22 and 23, where we show propellers inducing bubbles and waves, as well as a rotating tire splashing water. However, the typical "leakage" issue of the immersed boundary method renders this approach limited and

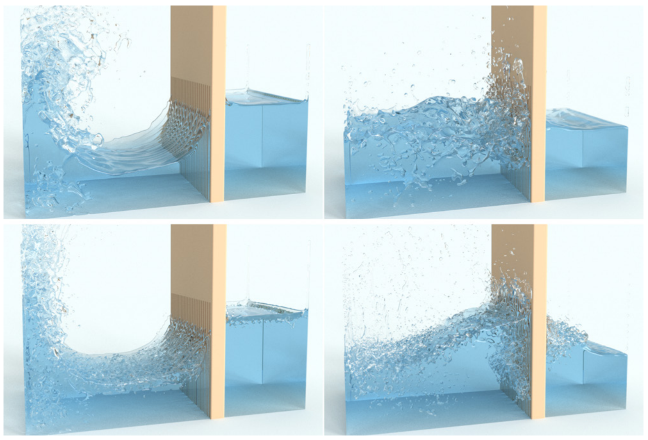

Fig. 21. **Low vs. high-resolution.** For the dam break through narrow slits from Fig. 16, using a 306×306×153 grid resolution (top) cannot capture as fine bubbles and droplets as a 512×512×256 resolution (bottom), but the overall motion is still well approximated as evidenced by these two frames.

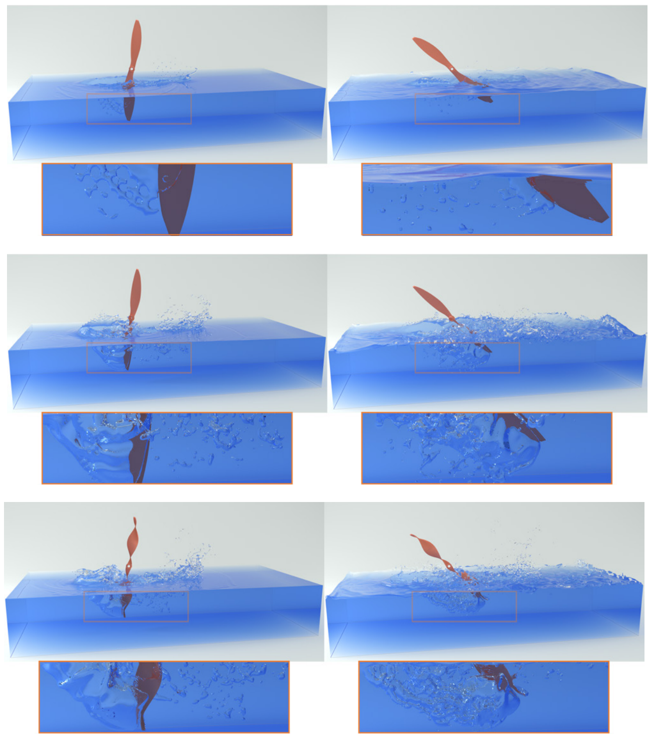
Fig. 22. **Propellers** While a two-blade propeller rotating between air and water generates small bubbles and minor perturbation of the free surface (top), offsetting the axis of rotation (center) or changing the blade shape (bottom) clearly increase both the underwater and free surface effects.

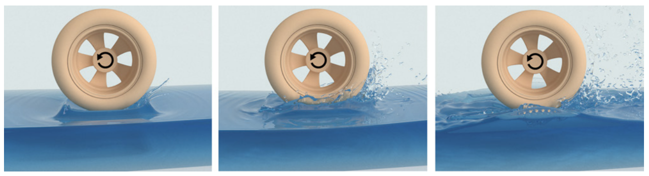
Fig. 23. **Rotating wheel.** A smooth, spinning wheel barely touching the surface of a water body generates splashing.

brittle. Moreover, the cases we show do not use high velocities: such a simple one-way coupling treatment is *nowhere near as stable as what we proposed in this paper for static obstacles*: our contributions to ensure robustness for static obstacles do not suffice in the case of fast-evolving solid objects. We leave a proper extension of one- or two-way coupling for future work, as the same level of stability for full coupling in LBM-based multiphase flow simulation would offer an exciting framework for more advanced simulation scenarios.

5 CONCLUSION

While computer animation has generated exquisite simulations of viscous fluids (from dough to syrupy fluids) using the (macroscopic) Navier-Stokes equations, simulating nearly inviscid incompressible fluids such as water remains a major challenge. Moreover, capturing the fine interaction between water and air that we encounter in our daily lives requires a solver that can handle large density ratios, adding more numerical difficulty. In this paper, we proposed a kinetic solver that offers an effective and unified computational framework to capture most of the salient multiphase flow behaviors seen in everyday life. Through a higher-order collision operator and a number of contributions to handle boundary conditions robustly, our two-phase immiscible fluid solver can simulate real-life density ratios (800 for air-water) and Reynolds numbers ($Re > 4000$ for interesting turbulent flows) both faithfully and efficiently, for complex obstacles — be they thin or thick. Its statistical mechanics nature allows for a massively-parallel implementation on GPU, ensuring an efficiency superior to existing methods, while reducing the memory usage compared to other kinetic solvers. For future work, we plan on removing the limitations listed in Sec. 4.5. Improving the resolution of the phase field through the use of adaptive grids is a particularly interesting research direction. Finally, addressing the case of coupling fluids with deformable solids would bring a whole slew of interesting applications such as blood flow simulation.

ACKNOWLEDGMENTS

Meshes are courtesy of Technische Universität München (Figs. 15 and 23), Dhanasekar Vinayagamoorthy (Fig. 22), and kathzerrato (Fig. 18). The colander video is from Chu Huarong (Fig. 20). The authors would like to thank the reviewers for their constructive comments. XPL acknowledges grant no. 62072310 from the National Natural Science Foundation of China; MD acknowledges the generous support of a Choose France Inria chair, and of Ansys, Inc.

REFERENCES

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power Diagrams and Sparse Paged Grids for High Resolution Adaptive Liquids. *ACM Trans. Graph.* 36, 4 (2017), 140:1–12.

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2015. A stream function solver for liquid simulations. *ACM Trans. Graph.* 34, 4 (2015), 53:1–9.

Vinicius C. Azevedo, Christopher Batty, and Manuel M. Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Trans. Graph.* 35, 4, Article 97 (2016), 97:1–12 pages.

Yan Ba, Haihu Liu, Qing Li, Qinjun Kang, and Jinju Sun. 2016. Multiple-relaxation-time color-gradient lattice Boltzmann model for simulating two-phase flows with high density ratio. *Physical Review E* 94, 2 (2016), 023310:1–15.

Vittorio E Badalassi, Hector D Ceniceros, and Sanjoy Banerjee. 2003. Computation of multiphase systems with phase field models. *J. Comput. Phys.* 190, 2 (2003), 371–397.

Stefan Band, Christoph Gissler, Markus Ihmsen, Jens Cornelis, Andreas Peer, and Matthias Teschner. 2018. Pressure Boundaries for Implicit Incompressible SPH. *ACM Trans. Graph.* 37, 2 (2018), 14:1–11.

Christopher Batty and Robert Bridson. 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Symposium on Computer Animation*. 219–228.

Jan Bender and Dan Koschier. 2016. Divergence-free SPH for incompressible and viscous fluids. *IEEE Trans. Vis. & Comp. Graph.* 23, 3 (2016), 1193–1206.

Prabhu Lal Bhatnagar, Eugene P. Gross, and Max Krook. 1954. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Phys. Rev.* 94, 3 (1954), 511–525.

Morten Bojsen-Hansen and Chris Wojtan. 2013. Liquid surface tracking with error compensation. *ACM Trans. Graph.* 32, 4 (2013), 79:1–79:10.

Morten Bojsen-Hansen and Chris Wojtan. 2016. Generalized Non-reflecting Boundaries for Fluid Re-simulation. *ACM Trans. Graph.* 35, 4 (July 2016), 96:1–96:7.

Landon Boyd and Robert Bridson. 2012. MultiFLIP for energetic two-phase fluid simulation. *ACM Trans. Graph.* 31, 2, Article 16 (2012).

Oleksiy Busaryev, Tamal K. Dey, Huamin Wang, and Zhong Ren. 2012. Animating bubble interactions in a liquid foam. *ACM Trans. Graph.* 31, 4 (2012), 1–8.

Chaos. 2022. V-Ray renderer. (2022). https://www.chaos.com/

Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. 2021. GPU Optimization for High-Quality Kinetic Fluid Simulation. *IEEE Trans. Vis. & Comp. Graph.* Preprint (2021).

Yi-Lu Chen, Jonathan Meier, Barbara Solenthaler, and Vinicius C Azevedo. 2020. An extended cut-cell method for sub-grid liquids tracking with surface tension. *ACM Trans. Graph.* 39, 6 (2020), 1–13.

Nuttapong Chentanez, Matthias Müller, Miles Macklin, and Tae-Yong Kim. 2015. Fast grid-free surface tracking. *ACM Trans. Graph.* 34, 4 (2015), 48:1–48:11.

Sergio Chibbaro, Giacomo Falcucci, Giancarlo Chiatti, Hudong Chen, Xiaowen Shan, and Sauro Succi. 2008. Lattice Boltzmann models for nonideal fluids with arrested phase-separation. *Physical Review E* 77, 3 (2008), 036705.

Jonathan Chin. 2002. Lattice Boltzmann simulation of the flow of binary immiscible fluids with different viscosities using the Shan-Chen microscopic interaction model. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 360, 1792 (2002), 547–558.

Junghyun Cho and Hyeong-Seok Ko. 2013. Geometry-Aware Volume-of-Fluid Method. In *Computer Graphics Forum*, Vol. 32. 379–388.

Jens Cornels, Markus Ihmsen, Andreas Peer, and Matthias Teschner. 2014. IISPH-FLIP for incompressible fluids. *Computer Graphics Forum* 33, 2 (2014), 255–262.

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-only liquids. *ACM Trans. Graph.* 35, 4 (2016), 78:1–12.

Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power particles: an incompressible fluid solver based on power diagrams. *ACM Trans. Graph.* 34, 4 (2015), 50:1–11.

Alessandro De Rosis and Christophe Coreixas. 2020. Multiphysics flow simulations using D3Q19 lattice Boltzmann methods based on central moments. *Physics of Fluids* 32, 11 (2020), 117101.

Alessandro De Rosis and Enatri Enan. 2021. A three-dimensional phase-field lattice Boltzmann method for incompressible two-components flows. *Physics of Fluids* 33, 4 (2021), 043315.

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

Mathieu Desbrun and Marie-Paule Cani-Gascuel. 1998. Active Implicit Surface for Animation. In *Graphics Interface.* 143–150.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *Comp. Anim. and Sim.* 61–76.

Douglas Enright, Frank Losasso, and Ron Fedkiw. 2005. A fast and accurate semi-Lagrangian particle level set method. *Comput. & Structures* 83, 6-7 (2005), 479–490.

Douglas Enright, Steve Marschner, and Ron Fedkiw. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph.* 21, 3 (2002), 736–744.

Abbas Fakhari and Diogo Bolster. 2017. Diffuse interface modeling of three-phase contact line dynamics on curved boundaries: A lattice Boltzmann model for large density and viscosity ratios. *J. Comput. Phys.* 334 (2017), 620–638.

Abbas Fakhari, Diogo Bolster, and Li-Shi Luo. 2017a. A weighted multiple-relaxation-time lattice Boltzmann method for multiphase flows and its application to partial coalescence cascades. *J. Comput. Phys.* 341 (2017), 22–43.

Abbas Fakhari, Martin Geier, and Diogo Bolster. 2019. A simple phase-field model for interface tracking in three dimensions. *Computers & Mathematics with Applications* 78, 4 (2019), 1154–1165.

Abbas Fakhari, Travis Mitchell, Christopher Leonardi, and Diogo Bolster. 2017b. Improved locality of the phase-field lattice-Boltzmann model for immiscible fluids at high density ratios. *Physical Review E* 96, 5 (2017), 053301.

Giacomo Falcucci, Gino Bella, Giancarlo Chiatti, Sergio Chibbaro, Mauro Sbragaglia, Sauro Succi, et al. 2007. Lattice Boltzmann models with mid-range interactions. *Communications in Computational Physics* 2, 6 (2007), 1071–1084.

Giacomo Falcucci, Stefano Ubertini, Chiara Biscarini, Silvia Di Francesco, Daniele Chiappini, Silvia Palpacelli, Alessandro De Maio, and Sauro Succi. 2011. Lattice Boltzmann methods for multiphase flow simulations across scales. *Communications in Computational Physics* 9, 2 (2011), 269–296.

Olga Filippova and Dieter Hänel. 1998. Grid refinement for lattice-BGK models. *Journal of Computational physics* 147, 1 (1998), 219–228.

Nick Foster and Ronald Fedkiw. 2001. Practical animation of liquids. In *Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH).* 23–30.

Nick Foster and Demetri Metaxas. 1996. Realistic animation of liquids. *Graphical Models and Image Processing* 58 (1996), 471–493.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial Particle-In-Cell method. *ACM Trans. Graph.* 36, 6 (2017), 222.

Ming Gao, Andre Pradhana, Xuchen Han, Qi Guo, Grant Kot, Eftychios Sifakis, and Chenfanfu Jiang. 2018. Animating fluid sediment mixture in particle-laden flows. *ACM Trans. Graph.* 37, 4 (2018), 149:1–11.

Martin Geier, Abbas Fakhari, and Taehun Lee. 2015. Conservative phase-field lattice Boltzmann model for interface tracking equation. *Physical Review E* 91, 6 (2015), 063309:1–11.

Sebastian Geller, Sonja Uphoff, and Manfred Krafczyk. 2013. Turbulent jet computations based on MRT and Cascaded Lattice Boltzmann models. *Computers & Mathematics with Applications* 65, 12 (2013), 1956–1966.

Ryan Goldade, Mridul Aanjaneya, and Christopher Batty. 2020. Constraint bubbles and affine regions: reduced fluid models for efficient immersed bubbles and flexible spatial coarsening. *ACM Trans. Graph.* 39, 4 (2020), 43–1.

Daryl Grunau, Shiyi Chen, and Kenneth Eggert. 1993. A lattice Boltzmann model for multiphase fluid flows. *Physics of Fluids A: Fluid Dynamics* 5, 10 (1993), 2557–2562.

G Gruszczyński, Travis Mitchell, Christopher Leonardi, and Tracie Barber. 2020. A cascaded phase-field lattice Boltzmann model for the simulation of incompressible, immiscible fluids with high density contrast. *Computers & Mathematics with Applications* 79, 4 (2020), 1049–1071.

Andrew K. Gunstensen, Daniel H. Rothman, Stéphane Zaleski, and Gianluigi Zanetti. 1991. Lattice Boltzmann model of immiscible fluids. *Physical Review A* 43, 8 (1991), 4320–4327.

Yulong Guo, Xiaopei Liu, and Xuemao Xu. 2017. A unified detail-preserving liquid simulation by two-phase lattice Boltzmann modeling. *IEEE Trans. Vis. & Comp.Graph.* 23, 5 (2017), 1479–1491.

Xiaoyi He, Shiyi Chen, and Gary D Doolen. 1998. A novel thermal model for the lattice Boltzmann method in incompressible limit. *J. Comput. Phys.* 146, 1 (1998), 282–300.

Xiaoyi He, Raoyang Zhang, Shiyi Chen, and Gary D Doolen. 1999. On the three-dimensional Rayleigh–Taylor instability. *Physics of Fluids* 11, 5 (1999), 1143–1152.

Nambin Heo and Hyeong-Seok Ko. 2010. Detail-preserving fully-Eulerian interface tracking framework. *ACM Trans. Graph.* 29, 6 (2010), 176:1–176:8.

David J. Holdych, Dimitrios Rovas, John G. Georgiadis, and Richard O. Buckius. 1998. An improved hydrodynamics formulation for multiphase flow lattice-Boltzmann models. *International Journal of Modern Physics C* 9, 08 (1998), 1393–1404.

Jeong-Mo Hong and Chang-Hun Kim. 2005. Discontinuous fluids. *ACM Trans. Graph.* 24, 3 (2005), 915–920.

Yang Hu, Decai Li, and Qiang He. 2020. Generalized conservative phase field model and its lattice Boltzmann scheme for multicomponent multiphase flows. *International Journal of Multiphase Flow* 132 (2020), 103432.

Takaji Inamuro, Nobuharu Konishi, and Fumimaru Ogino. 2000. A Galilean invariant model of the lattice Boltzmann method for multiphase fluid flows using free-energy approach. *Computer Physics Communications* 129, 1-3 (2000), 32–45.

David Jacqmin. 1999. Calculation of Two-Phase Navier-Stokes Flows Using Phase-Field Modeling. *J. Comput. Phys.* 155, 1 (1999), 96–127.

Alexandros N. Kalarakis, Vasilis N. Burganos, and Alkiviades C. Payatakes. 2002. Galilean-invariant lattice-Boltzmann simulation of liquid-vapor interface dynamics. *Physical Review E* 65, 5 (2002), 056702.

Myungjoo Kang, Ronald P Fedkiw, and Xu-Dong Liu. 2000. A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing* 15, 3 (2000), 323–360.

Qinjun Kang, Dongxiao Zhang, and Shiyi Chen. 2004. Immiscible displacement in a channel: simulations of fingering in two dimensions. *Advances in Water Resources* 27, 1 (2004), 13–22.

Po-Hao Kao and Ruey-Jen Yang. 2008. An investigation into curved and moving boundary treatments in the lattice Boltzmann method. *J. Comput. Phys.* 227, 11 (2008), 5671–5690.

Vivien M Kendon, Michael E Cates, Ignacio Pagonabarraga, J-C Desplat, and Peter Bladon. 2001. Inertial effects in three-dimensional spinodal decomposition of a symmetric binary fluid mixture: a lattice Boltzmann study. *Journal of Fluid Mechanics* 440 (2001), 147–203.

Byungmoon Kim. 2010. Multi-phase fluid simulations using regional level sets. *ACM Trans. Graph.* 29, 6 (2010), 175:1–8.

Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. 2007. Simulation of bubbles in foam with the volume control method. *ACM Trans. Graph.* 26, 3 (2007), 98.

Doyub Kim, Oh-Young Song, and Hyeong-Seok Ko. 2010. A Practical Simulation of Dispersed Bubble Flow. *ACM Trans. Graph.* 29, 4 (2010), 70:1–5.

Theodore Kim, Jerry Tessendorf, and Jils Thurey. 2013. Closest point turbulence for liquid surfaces. *ACM Trans. Graph.* 32, 2 (2013), 15:1–15:13.

Dan Koschier and Jan Bender. 2017. Density Maps for Improved SPH Boundary Handling. In *Symposium on Computer Animation.* Article 1.

E. Dinesh Kumar, Sannasi Annamalaisamy Sannasiraj, and Vallam Sundar. 2019. Phase field lattice Boltzmann model for air-water two phase flows. *Physics of Fluids* 31, 7 (2019), 072103.

Anthony J. C. Ladd. 1994. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics* 271 (1994), 285–309.

Timothy R. Langlois, Changxi Zheng, and Doug L. James. 2016. Toward Animating Water with Complex Acoustic Bubbles. *ACM Trans. Graph.* 35, 4 (2016), 95:1–13.

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

Jonas Latt. 2007b. Technical report: How to implement your DdQq dynamics with only q variables per node (instead of $2q$). *Tufts University* (2007), 1–8.

Sébastien Leclaire, Marcelo Reggio, and Jean-Yves Trépanier. 2011. Isotropic color gradient for simulating very high-density ratios with a two-phase flow lattice Boltzmann model. *Computers & Fluids* 48, 1 (2011), 98–112.

Hyun Geun Lee and Junseok Kim. 2013. Numerical simulation of the three-dimensional Rayleigh–Taylor instability. *Computers & Mathematics with Applications* 66, 8 (2013), 1466–1474.

Taehun Lee and Ching-Long Lin. 2005. A stable discretization of the lattice Boltzmann equation for simulation of incompressible two-phase flows at high density ratio. *J.*

Comput. Phys. 206, 1 (2005), 16–47.

Qing Li, Kaihong Luo, YJ Gao, and YL He. 2012. Additional interfacial force in lattice Boltzmann models for incompressible multiphase flows. *Physical Review E* 85, 2 (2012), 026704.

Wei Li, Kai Bai, and Xiaopei Liu. 2019. Continuous-Scale Kinetic Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* 25, 9 (2019), 2694–2709. https://doi.org/10.1109/TVCG.2018.2859931

Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling. *ACM Trans. Graph.* 39, 4 (2020).

Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. 2021. Kinetic-Based Multiphase Flow Simulation. *IEEE Trans. Vis. & Comp. Graph.* 27, 7 (2021), 3318–3334.

Hong Liang, Jiangrong Xu, Jiangxing Chen, Huili Wang, Zhenhua Chai, and Baochang Shi. 2018. Phase-field-based lattice Boltzmann modeling of large-density-ratio two-phase flows. *Physical Review E* 97, 3 (2018), 033309.

Frank Losasso, Frederic Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM Trans. Graph.* 457–462.

Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple interacting liquids. *ACM Trans. Graph.* 25, 3 (2006), 812–819.

Daniel Lycett-Brown, Kai H. Luo, Ronghou Liu, and Pengmei Lv. 2014. Binary droplet collision simulations by a multiphase cascaded lattice Boltzmann method. *Physics of Fluids* 26 (2014), 023303:1–26.

Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Trans. Graph.* 40, 6 (2021), 1–18.

Michael E McCracken and John Abraham. 2005. Multiple-relaxation-time lattice-Boltzmann model for multiphase flow. *Physical Review E* 71, 3 (2005), 036701.

Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. 2004. Animation and Control of Breaking Waves. In *Symposium on Computer Animation.* 315–324.

Viorel Mihalef, Betul Unlusu, Dimitris Metaxas, Mark Sussman, and M Yousuff Hussaini. 2006. Physics based boiling simulation. In *Symposium on Computer Animation.* 317–324.

Marek Krzysztof Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. 2013. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE Trans. Vis. & Comp. Graph.* 20, 1 (2013), 4–16.

Shiladitya Mukherjee and John Abraham. 2007a. Lattice Boltzmann simulations of two-phase flow with high density ratio in axially symmetric geometry. *Physical Review E* 75, 2 (2007), 026701.

Shiladitya Mukherjee and John Abraham. 2007b. A pressure-evolution-based multi-relaxation-time high-density-ratio two-phase lattice-Boltzmann model. *Computers & Fluids* 36, 6 (2007), 1149–1158.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Symposium on Computer Animation.* 154–159.

Seyed Nabavizadeh, Mohsen Eshraghi, and Sergio Felicelli. 2018. A Comparative Study of Multiphase Lattice Boltzmann Methods for Bubble-Dendrite Interaction during Solidification of Alloys. *Appl. Sci.* 9, 1 (2018), 57:1–24.

Xiao-Dong Niu, You Li, Yi-Ren Ma, Mu-Feng Chen, Xiang Li, and Qiao-Zhong Li. 2018. A mass-conserving multiphase lattice Boltzmann model for simulation of multiphase flows. *Physics of Fluids* 30, 1 (2018), 013302:1–13.

Enzo Orlandini, Michael R Swift, and JM Yeomans. 1995. A lattice Boltzmann model of binary-fluid mixtures. *Euro-Physics Letters* 32, 6 (1995), 463.

Saket Patkar, Mridul Aanjaneya, Dmitriy Karpman, and Ronald Fedkiw. 2013. A hybrid Lagrangian-Eulerian formulation for bubble generation and dynamics. In *Symp. Comp. Anim.* 105–114.

Saket Patkar and Parag Chaudhuri. 2013. Wetting of Porous Solids. *IEEE Trans. Vis. & Comp. Graph.* 19, 9 (2013), 1592–1604.

Bo Ren, Chenfeng Li, Xiao Yan, Ming C Lin, Javier Bonet, and Shi-Min Hu. 2014. Multiple-fluid SPH simulation using a mixture model. *ACM Trans. Graph.* 33, 5 (2014), 171:1–11.

Shimpei Saito, Alessandro De Rosis, Alessio Festuccia, Akiko Kaneko, Yutaka Abe, and Kazuya Koyama. 2018. Color-gradient lattice Boltzmann model with nonorthogonal central moments: Hydrodynamic melt-jet breakup simulations. *Physical Review E* 98, 1 (2018), 013305.

Robert Saye. 2016. Interfacial gauge methods for incompressible fluid dynamics. *Science Advances* 2, 6 (2016), e1501869.

Robert Saye. 2017. Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part I. *J. Comput. Phys.* 344 (2017), 647–682.

Mauro Sbragaglia, Roberto Benzi, Luca Biferale, Sauro Succi, Kazu Sugiyama, and Federico Toschi. 2007. Generalized lattice Boltzmann method with multirange pseudopotential. *Physical Review E* 75, 2 (2007), 026702.

Hagit Schechter and Robert Bridson. 2012. Ghost SPH for Animating Water. *ACM Trans. Graph.* 31, 4 (2012).

Xiaowen Shan and Hudong Chen. 1993. Lattice Boltzmann model for simulating flows with multiple phases and components. *Physical review E* 47, 3 (1993), 1815.

Xiaowen Shan and Hudong Chen. 1994. Simulation of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation. *Physical Review E* 49, 4 (1994), 2941.

J.Y. Shao, C. Shu, H.B. Huang, and Y.T. Chew. 2014. Free-energy-based lattice Boltzmann model for the simulation of multiphase flows with density contrast. *Physical Review E* 89, 3 (2014), 033309:1–14.

Barbara Solenthaler and Renato Pajarola. 2008. Density Contrast SPH Interfaces. In *Symposium on Computer Animation.* 211–218.

Barbara Solenthaler and Renato Pajarola. 2009. Predictive-Corrective incompressible SPH. *ACM Trans. Graph.* 28, 3 (2009), 40:1–6.

Oh-Young Song, Hyuncheol Shin, and Hyeong-Seok Ko. 2005. Stable but nondissipative water. *ACM Trans. Graph.* 24, 1 (2005), 81–97.

Haozhe Su, Tao Xue, Chengguizi Han, Chenfanfu Jiang, and Mridul Aanjaneya. 2021. A unified second-order accurate in time MPM formulation for simulating viscoelastic liquids with phase change. *ACM Trans. Graph.* 40, 4 (2021), 1–18.

Kazuhiko Suga, Yoshiaki Kuwata, K Takashima, and R Chikasue. 2015. A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows. *Computers & Mathematics with Applications* 69, 6 (2015), 518–529.

Michael R Swift, E Orlandini, WR Osborn, and JM Yeomans. 1996. Lattice Boltzmann simulations of liquid-gas and binary fluid systems. *Physical Review E* 54, 5 (1996), 5041.

Michael R Swift, WR Osborn, and JM Yeomans. 1995. Lattice Boltzmann simulation of nonideal fluids. *Physical Review Letters* 75, 5 (1995), 830–833.

Nils Thuerey, Filip Sadlo, Simon Schirm, Matthias Müller-Fischer, and Markus Gross. 2007. Real-time simulations of bubbles and foam within a shallow water framework. In *Symposium on Computer Animation.* 191–198.

H.L. Wang, Z.H. Chai, B.C. Shi, and H. Liang. 2016. Comparative study of the lattice Boltzmann models for Allen-Cahn and Cahn-Hilliard equations. *Physical Review E* 94, 3 (2016), 033304.

Chris Wojtan, Matthias Müller-Fischer, and Tyson Brochu. 2011. Liquid simulation with mesh-based surface tracking. *ACM SIGGRAPH Course Notes* (2011).

Xiao Yan, Yun-Tao Jiang, Chenfeng Li, Ralph R. Martin, and Shi-Min Hu. 2016. Multiphase SPH simulation for interactive fluids and solids. *ACM Trans. Graph.* 35, 4, Article 79 (2016).

Shuqi Yang, Shiying Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. 2021. Clebsch gauge fluid. *ACM Trans. Graph.* 40, 4 (2021), 1–11.

Tao Yang, Jian Chang, Ming C. Lin, Ralph R. Martin, Jian J. Zhang, and Shi min Hu. 2017. A unified particle system framework for multi-phase, multi-material visual simulations. *ACM Trans. Graph.* 36, 6 (2017), 224:1–13.

Xuewen Yin and Junfeng Zhang. 2012. An improved bounce-back scheme for complex boundary conditions in lattice Boltzmann method. *J. Comput. Phys.* 231, 11 (2012), 4295–4303.

Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2015. Continuum foam: A material point method for shear-dependent flows. *ACM Trans. Graph.* 34, 5 (2015), 160:1–20.

Fan Zhang, Xiong Zhang, Kam Yim Sze, Yanping Lian, and Yan Liu. 2017. Incompressible material point method for free surface flow. *J. Comput. Phys.* 330 (2017), 92–110.

Yizhong Zhang, Huamin Wang, Shuai Wang, Yiying Tong, and Kun Zhou. 2012. A Deformable Surface Model for Real-Time Water Drop Animation. *IEEE Trans. Vis. & Comp. Graph.* 18, 8 (2012), 1281–1289.

HW Zheng, Chang Shu, and Yong-Tian Chew. 2006. A lattice Boltzmann model for multiphase flows with large density ratio. *J. Comput. Phys.* 218, 1 (2006), 353–371.

Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. 2009. Simulation of bubbles. *Graphical Models* 71, 6 (2009), 229–239.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Trans. Graph.* 24, 3 (2005), 965–972.

Yingqing Zu and Shuisheng He. 2013. Phase-field-based lattice Boltzmann model for incompressible binary fluid systems with density and viscosity contrasts. *Physical Review E* 87, 4 (2013), 043301.