Kinetic Free-Surface Flows and Foams with Sharp Interfaces

HAOXIANG WANG, Tsinghua University, China
KUI WU, Lightspeed, USA
HUI QIAO[*], Tsinghua University, China
MATHIEU DESBRUN, Inria/Ecole Polytechnique, France
WEI LI[†], Shanghai Jiao Tong University, China and Lightspeed, China

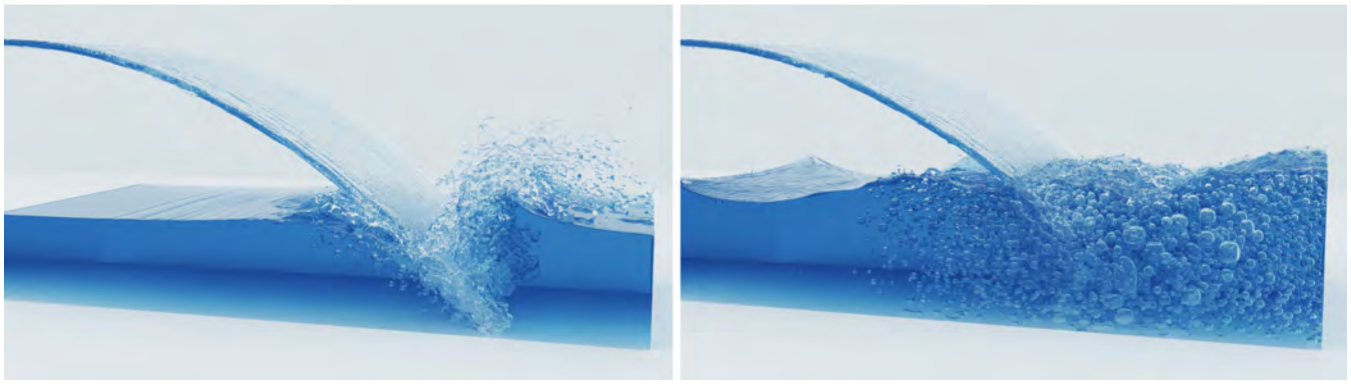


Fig. 1. **HOME-FREE LBM solver.** Our novel fluid solver mixes a kinetic flow simulator and a volume-of-fluid sharp moving interface. For any given grid resolution, our approach captures far more detailed fluid-air interactions than diffuse-interface based solvers. Moreover, we leverage the fact that the ratio of density and viscosity between air and liquid is large enough that simulating the gas phase can be avoided, the resulting free-surface approximation thus saving a big fraction of computational time. In this example, a fast-evolving sheet of water is flowing into a container, creating turbulence and a large amount of bubbles bursting over time, computed at a resolution of 600×300×300 in 58s per frame and with only 8 Gb of memory.

Kinetic multiphase flow solvers have recently demonstrated exquisitely complex and turbulent fluid phenomena involving splashing and bubbling. However, they require full simulation of both the liquid phase and the air to capture a large spectrum of fluid behaviors. Moreover, they rely on diffuse interface tracking to properly account for the interfacial forces involved in fluid-air interactions. Consequently, simulating visually appealing fluids is extremely compute intensive given the required resolution to capture small bubbles, and foam simulation is unattainable with this family of methods. While water simulation involves density and viscosity differences between the two phases so large that one can safely ignore the dynamics of air, so-called kinetic free-surface solvers that only consider the liquid motion have been unable to reproduce the full gamut of turbulent fluid behaviors, being often unstable for even moderately complex scenarios. By revisiting kinetic solvers using sharp interfaces and incorporating recent advances in single-phase and multiphase LBM solvers, we propose a free-surface kinetic solver, which we call HOME-FREE LBM, that not only handles turbulence, glugging, and bubbling, but even foam where bubbles stick to each other through surface tension. We demonstrate that our fluid simulator allows for fast and robust bubble growth, breakup, and coalescence, at a fraction of the computational time that existing CG fluid solvers require.

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

Additional Key Words and Phrases: Lattice Boltzmann method, velocity moments, turbulent flows, free-surface flow, foaming.

ACM Reference Format:
Haoxiang Wang, Kui Wu, Hui Qiao[*], Mathieu Desbrun, and Wei Li[†]. 2025. Kinetic Free-Surface Flows and Foams with Sharp Interfaces. *ACM Trans. Graph.* 44, 6, Article 266 (December 2025), 17 pages. https://doi.org/10.1145/3763288

1 Introduction

With applications ranging from aerodynamics and hydrodynamics to special effects in movies, fluid flows often exhibit fascinating visual complexity. Free-surface flows — involving the motion of the interface between fluid and air — are especially intricate, displaying bubbles rising and bursting, splashing, and even foam. While many efficient techniques for free-surface simulation based on grids or particles have been proposed, they often fail to offer a unified approach to achieving realistic and complex free-surface behaviors. Current solvers capable of simulating free-surface flows face several limitations. Many rely on diffuse interfaces to properly capture fluid-air interactions, requiring high-resolution computations [Song et al. 2005; Zheng et al. 2009] and the entire simulation of the air flow [Li et al. 2021, 2022, 2024]. Others suffer from numerical instability or significant dissipation, limiting their ability to accurately capture air-fluid interactions in high Reynolds number turbulence

[†]Corresponding author; [*] Co-corresponding author.
Authors' addresses: H. Wang (whx22@mails.tsinghua.edu.cn): Department of Automation, Tsinghua University, Beijing, China; K. Wu (walker.kui.wu@gmail.com): LightSpeed Studios, Los Angeles, USA; H. Qiao (qiaohui@mail.tsinghua.edu.cn): Department of Automation, Tsinghua University, Beijing, China; M. Desbrun (mathieu.desbrun@inria.fr): Inria Saclay and LIX at Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France; W. Li (1104720604wei@gmail.com): formerly at LightSpeed Studios, Shanghai, China, now at the John Hopcroft Center for Computer Science, Shanghai Jiao Tong University, Shanghai, China.

[Qu et al. 2023; Deng et al. 2022; Wang et al. 2024]. Moreover, specialized approaches, such as those using bubble particles, are often limited to bubbling and do not extend to other free-surface phenomena [Busaryev et al. 2012; Wretborn et al. 2022]. Simulating complex free-surface flows with a unified and efficient single-phase solver thus remains a major challenge.

In this work, we introduce a high-performance kinetic approach to simulating turbulent free-surface flows that is capable of handling a large variety of air-fluid phenomena such as foaming, bubbling, glugging, and splashing. Our method builds upon the free-surface Lattice Boltzmann Method (FSLBM [Thürey 2007], which models the air-fluid interface as a sharp boundary), integrates recent innovations in computer graphics, and adds key contributions to enable highly turbulent simulations at low computational cost. Compared to current kinetic free-surface solvers, we present various advances:

- we significantly improve upon the existing FSLBM [Thürey 2007] by incorporating a High-Order Moment-Encoded LBM (HOME-LBM [Li et al. 2023b]) encoding of distribution functions, resulting in our more efficient and more stable HOME-FREE LBM formulation supporting high-turbulence free-surface flows;
- we introduce a massively-parallel algorithm to accurately track bubbles and estimate their sizes, enabling detailed simulations of foam and bubbles;
- we incorporate a turbulence model to significantly reduce the possible disappearance of very small bubbles;
- we adapt the double-sided bounce-back approach of Lyu et al. [2023] to our framework and derive an efficient two-way force expression that is valid for both thin-shell and thick objects;
- we also propose a new handling of fresh cells for liquid-gas-solid coupling that prevents the usual sticking or bubbling artifacts of previous methods;
- finally, we employ the D3Q7 central-moment based collision model of [Li et al. 2022] for the advection and diffusion of gas concentration, to which a novel adaptive viscosity model is added in order to stabilize complex membrane structures and foam dynamics at high Reynolds numbers.

The resulting fluid integrator is a unified computational technique to simulate a broad range of free-surface phenomena (from glugging, to bubbles and foam), which offers *significant* improvements in timings, memory usage, and numerical stability over state-of-the-art LBM-based two-phase fluid flow simulators.

2 Related work

Many numerical techniques have tried to emulate free-surface fluid flows. We briefly review related works on free-surface and multiphase fluid flow simulation in order to motivate our approach.

2.1 Navier-Stokes based methods

Many methods rely on numerical approximations of the Navier-Stokes equations, based on particles, (fixed) Eulerian grids, or (moving) Lagrangian grids. All are based on macroscopic discretizations of the fluid motion, but differ depending on whether they simulate only the fluid flow or both the fluid and the air flow.

Free-surface simulation. Particle methods have a long history in fluid simulation. Smoothed Particle Hydrodynamic (SPH), arguably the most popular particle-based approach, has been used early in free-surface simulation [Desbrun and Gascuel 1996; Premžoe et al. 2003], often with a blobby aspect of the induced interface [Desbrun and Cani-Gascuel 1998; Zhu and Bridson 2005]. Realism of SPH-based fluids improved over the years by incorporating incompressibility [Solenthaler and Pajarola 2009; Bender and Koschier 2016], enforcing boundary conditions better [Schechter and Bridson 2012; Koschier and Bender 2017; Band et al. 2018], adding penalty forces [Peer et al. 2015; Ihmsen et al. 2013], or better control over stability [Sun et al. 2018; Chalk et al. 2020; Liu et al. 2024]. Position-based dynamics (PBD) was also shown to be an efficient approach for free-surface simulation with particles [Macklin and Müller 2013; Zhang et al. 2015; Alduán et al. 2017], with recent improvements for the handling of surface tension [Xing et al. 2022]. Particle-In-Cell (PIC) [Foster and Metaxas 1996] and Fluid-Implicit-Particle (FLIP) [Zhu and Bridson 2005; Batty and Bridson 2008; Cornels et al. 2014; Azevedo et al. 2016; Fu et al. 2017] have been shown most useful for viscous fluid simulation, similar to a number of other methods coupling particles and grids for efficient computation [Fei et al. 2018; Hu et al. 2018; Fei et al. 2019; Fang et al. 2020; Fei et al. 2021; Sancho et al. 2024]. Mixing Eulerian and Lagrangian discretizations was also proposed to handle large surface tension, where the interface is simulated as a Lagrangian membrane [Ruan et al. 2021], and faster pressure projections were achieved through algebraic multigrid approaches [Shao et al. 2022]. A major step towards improving the details of free-surface simulation was achieved by adopting the level-set method to animate the interface [Goldade et al. 2016; Aanjaneya et al. 2017; Ando and Batty 2020]. Concurrently, the volume-of-fluid (VOF) method has been employed to design monolithic solvers for fluid-solid coupling [Takahashi and Batty 2020, 2022]. Recently, new techniques were proposed to reproduce complex free-surface phenomena; for instance, a Clebsch-based method [Yang et al. 2021; Xiong et al. 2022] and the Gradient-Augmented Reference-Map Method for Level-Set (GARM-LS) [Li et al. 2023a] were introduced to simulate very detailed free-surface flows. Note that from all these contributions, only two papers showed examples of flows involving bubbles [Xing et al. 2022; Xiong et al. 2022].

Multiphase simulation. While the efficient inclusion of bubbles has been proposed in a few approaches [Thürey et al. 2007; Kim 2010; Goldade et al. 2020; Wretborn et al. 2022, 2025], simulating the full spectrum of multiphase flow behaviors has often required the use of multiphase flow simulation where both fluid and air are fully simulated. First, SPH-based multiphase simulations [Solenthaler and Pajarola 2008; Ren et al. 2014; Yan et al. 2016; Yang et al. 2017] were proposed, but only for viscous flows. Power particles [de Goes et al. 2015; Aanjaneya et al. 2017] were used to preserve exact local volumes and momenta near the interface and improve the treatment of surface tension, before being extended via the material-point method (MPM) to handle bubbles or foam [Qu et al. 2023] — but again, only for fairly viscous fluids. A Moving-Least-Squares Reproducing-Kernel Method (MLSRK) framework to simulate multiphase fluids and solids in a unified manner was also proposed [Chen et al. 2020a], requiring very small timesteps in practice. The level set [Kim 2010] and VOF [Cho and Ko 2013; Langlois et al. 2016] methods were also adapted to multiphase flows

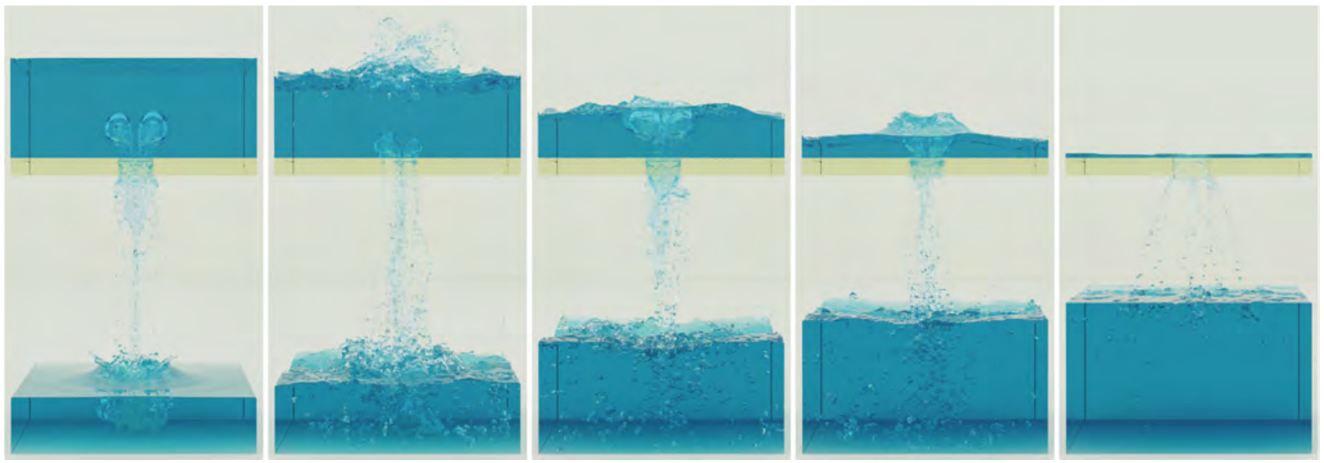


Fig. 2. **Glugging.** Our free-surface LBM-based fluid solver can handle glugging: as water is flowing down from an upper container, large bubbles first appear along with several small bubbles; as the liquid accumulates in the bottom container, many small bubbles are formed, evolving within the turbulent flow.

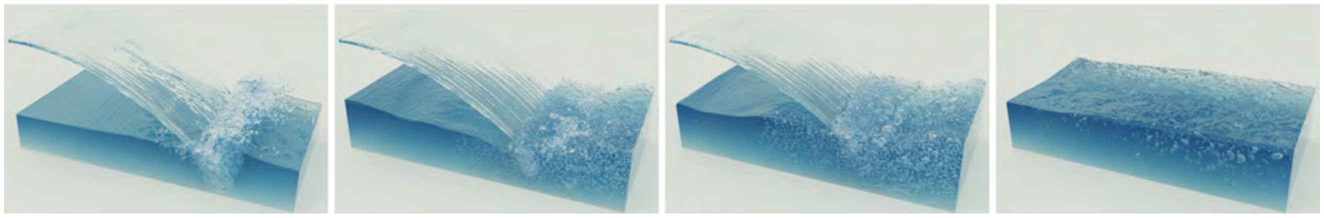


Fig. 3. **Pouring water.** This example, similar to Fig. 1 but with a smaller surface tension, shows a fast-evolving sheet of water that is flowing into a container, creating lots of turbulence and bubble motion, with many bubbles being formed, traveling around, clustering, some of them growing, and eventually bursting.

by improving the treatment of density and pressure jumps at the interface: most related methods [Losasso et al. 2006; Mihalef et al. 2006; Kim et al. 2007; Boyd and Bridson 2012] rely on a variable density pressure projection [Kang et al. 2000] and the ghost fluid method [Hong and Kim 2005] to treat discontinuous jumps. Karnakov et al. [2022] also proposed a multi-VOF method to simulate bubbles and foam, but at the cost of large computational costs. An MPM formulation was introduced in [Su et al. 2021; Tu et al. 2024] to simulate viscoelastic liquids with phase change, while a mesh-based Lagrangian approach to multiphase flows was proposed in [Misztal et al. 2013]; both used only very low Reynolds numbers in their examples. Recent Eulerian-Lagrangian particle approaches have also been successful at simulating bubbles [Deng et al. 2022; Wang et al. 2024], but no turbulent cases were shown. Hybrid solvers using different numerical coupling between fluid velocity, pressure, and interface position [Saye 2016, 2017; Sun et al. 2024] did not fare much better in handling turbulent free-surface flows.

2.2 Kinetic methods

The lattice Boltzmann method (LBM), originating from Computational Fluid Dynamics (CFD), has recently been shown extremely attractive for graphics due to its ability to handle turbulent flows while offering massively-parallel computations as we now review.

Free-surface simulation. While adopted in graphics [Thürey 2003, 2007], free-surface lattice Boltzmann method (FSLBM) was developed in CFD [Körner et al. 2005] by combining LBM with a VOF-based interface-capturing technique for the simulation of incompressible free surface flows. It uses a *sharp interface* between the two phases as the volume-of-fluid field serves the role of an indicator function. Moreover, the dynamics of the gas phase is ignored, and only the pressure influence at the interface is considered while the ratio of density and viscosity between the two fluid phases is assumed infinite. These simplifications lead to high computational efficiency (particularly on GPUs) and low memory usage compared to multiphase flow simulation. A series of improvements to this technique in terms of efficiency and stability has been proposed [Thürey and Rüde 2004; Thürey et al. 2005; Thürey and Rüde 2005; Thürey et al. 2006; Thürey and Rüde 2009; Cao et al. 2020], with open-source libraries now available [Janßen and Krafczyk 2011; Lehmann 2019; Plewinski et al. 2024]. Yet, despite improved (but compute-intensive) boundary treatments [Bogner et al. 2015], FSLBM has not been able to simulate turbulent free-surface flows, bubbles, or even foam.

Multiphase simulation. The last decade has seen rapid progress in the handling of bubbles with FSLBM, starting with [Körner et al. 2005; Thürey et al. 2007], and followed by a number of numerical and algorithmic improvements [Anderl et al. 2014; Bogner 2017] to efficiently handle bubble breakups and coalescence. These results are now available as part of an open-source CFD solver called LB-foam [Ataei et al. 2021]. Alas, we will demonstrate in this paper that the current state-of-the-art methods in FSLBM suffer from instability in turbulent cases and cannot handle complex or thin obstacles, rendering them ill-suited to graphics purposes. Other kinetic multiphase solvers have been able to resolve all these shortcomings in recent years by using a *diffuse interface* instead [Guo et al. 2017; Li et al. 2021, 2022; Li and Desbrun 2023; Li et al. 2024; Ma et al. 2024], as diffusiveness of the interface allows for better integration of the

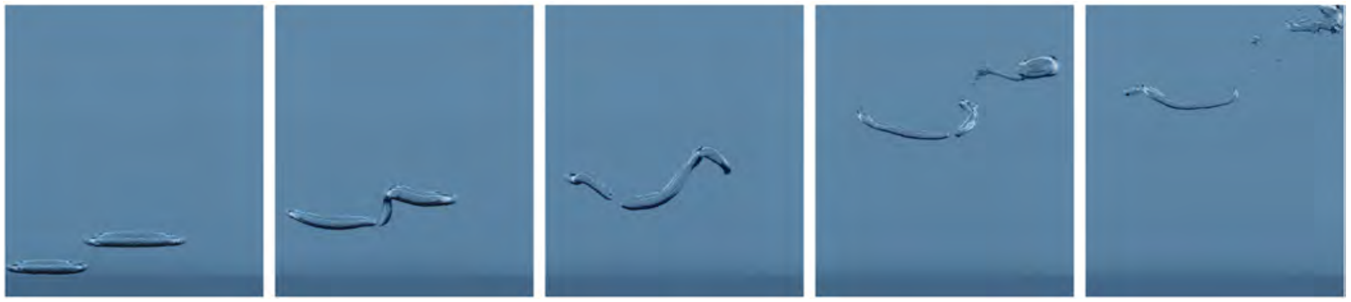


Fig. 4. **Bubble rings.** In this example, we show a time lapse of two bubble rings rising up, created by two strong vorticity loops for the initial velocity of the fluid and a VOF field discretizing two air tori near the bottom of a fluid tank. They first connect, before ejecting a small bubble on their way to the surface.

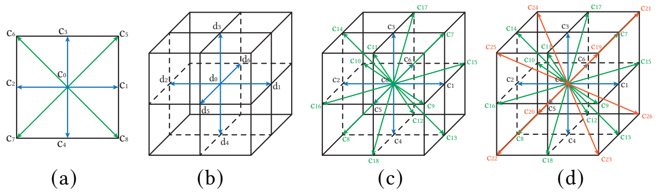


Fig. 5. **Lattice structures.** We use D2Q9 (a) in 2D and either D3Q19 (c) or D3Q27 (d) in 3D for our LBM fluid simulation, where the lattice directions $\boldsymbol{c}_i$ are discretized microscopic velocities. Each discretized distribution function f_i is associated with its corresponding velocity c_i. For the dissolved gas concentration, we use a simpler D3Q7 (b) with lattice directions $\boldsymbol{d}_i$ on which the distribution function g_i is simulated.

interfacial forces across the phases, adding stability compared to a sharp interface treatment. However, this comes at a hefty price: capturing small bubbles and their coalescence or breakups with such a diffuse interface requires extremely large grid sizes, which translates into very high memory requirements and timings. Moreover, foams (where bubbles are tightly packed) can simply not be handled well since a diffuse interface inherently smoothes the interface details, preventing the formation of complex foam structures. Our improved variant of FSLBM will, instead, offer a unified and robust handling of bubbles and foam with much reduced timings and memory usage than current kinetic solvers.

3 Background

Before introducing our new moment-encoded free-surface LBM model, we briefly recap the free-surface LBM method and its current variants, along with its most salient limitations.

3.1 Free surface lattice Boltzmann model (FSLBM)

LBM is a numerical approach that describes the motion of a fluid by the time evolution of a mesoscopic distribution function $\boldsymbol{f}(\boldsymbol{v}, \boldsymbol{x}, t)$, which represents the probability of a particle being present at position $\boldsymbol{x}$, at time t, and with velocity $\boldsymbol{v}$. To model single-phase or free surface fluid dynamics, the governing kinetic equation for the evolution of the distribution function, also called Boltzmann equation [Shan et al. 2006], is expressed as:

$$\frac{\partial \boldsymbol{f}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{f} = \Omega(\boldsymbol{f}) + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} \boldsymbol{f} , \tag{1}$$

where $\boldsymbol{F}$ represents external forces and Ω is a so-called collision operator that relaxes the distribution function towards a local thermodynamic equilibrium state (often denoted as $\boldsymbol{f}^{\text{eq}}$). This Boltzmann equation can be discretized on a lattice in space, velocity, and time, leading to the lattice Boltzmann equations, which read:

$$\forall i, \quad f_i(\boldsymbol{x} + \boldsymbol{c}_i, t + 1) - f_i(\boldsymbol{x}, t) = \Omega_i + F_i , \tag{2}$$

where $f_i(\boldsymbol{x}, t)$ encodes the distribution f in the i-th direction at position $\boldsymbol{x}$ and time t, $\boldsymbol{c}_i$ is the discrete lattice velocity in the i-th direction, Ω_i is the discretized collision operator, and F_i results from external forces projected onto distribution space. The D2Q9 and D3Q19 lattices shown in Fig. 5 are often employed for 2D and 3D simulations, respectively.

Through operator splitting, Eq. (2) can be divided in two steps. The first step is the *streaming step*, evaluating

$$f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x} - \boldsymbol{c}_i, t) , \tag{3}$$

followed by a *collision step* expressed as

$$f_i(\boldsymbol{x}, t + 1) = f_i^*(\boldsymbol{x}, t) + \Omega_i + F_i . \tag{4}$$

where $\Omega_i = -(f_i^*(\boldsymbol{x}, t) - f_i^{\text{eq}}(\boldsymbol{x}, t))/\tau$ is the BGK model [Bhathnagor et al. 1954], where $\tau = 3\nu + 0.5$ (ν being the kinematic viscosity) is the relaxation time. This simple collision model is adopted in LBfoam [Ataei et al. 2021]. Additionally, FSLBM only uses *low-order discretizations of the equilibrium state* $\boldsymbol{f}^{\text{eq}}$ expressed as

$$f_i^{\text{eq}}(\rho, \boldsymbol{u}) = \rho w_i (1 + \frac{\boldsymbol{c}_i \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{c}_i \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u}^2}{2c_s^2}) , \tag{5}$$

and *of the force term in distribution space*:

$$F_i(\boldsymbol{x}, t) = w_i (1 - \frac{1}{2\tau})(\frac{\boldsymbol{c}_i \boldsymbol{u}}{c_s^4} - \frac{\boldsymbol{u} - \boldsymbol{c}_i}{c_s^2}) \cdot \boldsymbol{F} , \tag{6}$$

where w_i are the lattice weights and c_s is the speed of sound ($1/\sqrt{3}$ in 3D). Macroscopic physical quantities are derived from the distribution function through its moments [Li et al. 2023b], yielding

$$\rho = \sum_{i=0}^{q-1} f_i, \quad \rho \boldsymbol{u} = \sum_{i=0}^{q-1} \boldsymbol{c}_i f_i + \frac{1}{2} \boldsymbol{F}, \quad \rho S_{\alpha\beta} = \sum_{i=0}^{q-1} (\boldsymbol{c}_i^2 - \frac{1}{3} \delta_{\alpha\beta}) f_i , \tag{7}$$

where S is the second-order velocity moment, while pressure is simply $p = \rho c_s^2$. Dealing with obstacles in the fluid is often achieved through one of the many variants of the bounce-back approach [Succi 2001] during the streaming step.

In order to track the interface motion between the liquid and the gas (air) phase, the volume-of-fluid (VOF) method [Hirt and Nichols 1981] is used in FSLBM: it consists in advecting a field ϕ in the fluid macroscopic velocity through a transport equation

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \nabla \phi = 0 , \tag{8}$$

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

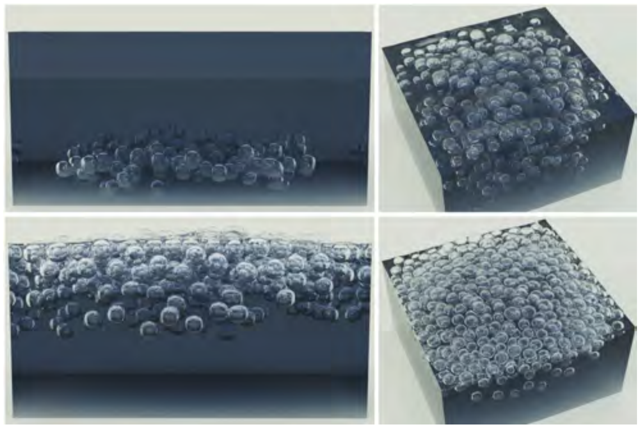
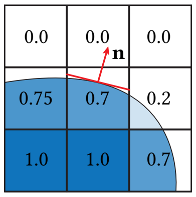


Fig. 6. **Bubbly layer.** After multiple bubbles are nucleated at the bottom of a tank, these bubbles rise and congregate tightly at the top of the liquid, creating a foam layer with a complex free surface.

volume fractions effectively partition the grid nodes into three categories: L representing liquid nodes, G representing air nodes, and I representing the interface nodes where $0 < \phi < 1$, which should always come in between liquid and gas nodes. A fast mass advection algorithm [Körner et al. 2005] is used in FSLBM to discretize Eq. (8), which takes advantage of the evolution of distribution functions to track ϕ in the domain: by calculating the mass exchange between neighboring nodes $\boldsymbol{x}$ and $\boldsymbol{x} + \boldsymbol{c}_i$ based on the distribution functions $\boldsymbol{f}$ and since the VOF (see inset) is equal to the fluid mass divided by its density, we get the new value of ϕ at $\boldsymbol{x}$ as



$$\phi(\boldsymbol{x}, t+1) = \phi(\boldsymbol{x}, t) + \frac{1}{\rho(\boldsymbol{x}, t)} \sum_{i=0}^{q-1} \theta(\boldsymbol{x}) \cdot \left(f_{\bar{i}}(\boldsymbol{x}+\boldsymbol{c}_i, t) - f_i(\boldsymbol{x}, t)\right), \tag{9}$$

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

$$\theta(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in L \\ 0 & \text{if } \boldsymbol{x} \in G \\ \frac{1}{2}(\phi(\boldsymbol{x}, t) + \phi(\boldsymbol{x} + \boldsymbol{c}_i, t)) & \text{if } \boldsymbol{x} \in I. \end{cases} \tag{10}$$

This interface advection approach is particularly convenient as it does not require the reconstruction of the interface or an approximation of its geometric properties such as normals or curvatures. One only needs to update the status of the nodes after advection: if $\phi(\boldsymbol{x}, t+1) \geq 1+\epsilon_\phi$ for $\epsilon_\phi = 1\text{e-}4$ (resp., if $\phi(\boldsymbol{x}, t+1) \leq 0-\epsilon_\phi$), an interface node is converted into a liquid (resp., gas) node and ϕ is clamped to 1 (resp., 0), and the clamped volumes are redistributed to neighboring cells to enforce mass preservation [Lehmann 2019].

While free-surface LBM ignores the dynamics of the gas phase, setting correct pressures is crucial to allow for bubbles not to cavitate. This is achieved through a pressure boundary condition: in the LBM step of Eq. (3), the distribution $\boldsymbol{f}$ from the gas to the interface node is unknown, but approximated via

$$f_i^*(\boldsymbol{x}, t) = f_i^{\text{eq}}(\rho_g, \boldsymbol{u}(\boldsymbol{x}, t)) + f_{\bar{i}}^{\text{eq}}(\rho_g, \boldsymbol{u}(\boldsymbol{x}, t)) - f_{\bar{i}}(\boldsymbol{x}, t), \tag{11}$$

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

$$\rho_g = (p_g - 2\gamma\kappa(\boldsymbol{x}))/c_s^2, \tag{12}$$

which accounts for the surface tension coefficient γ of the interface, the atmospheric pressure p_g [Körner et al. 2005], and the local mean curvature $\kappa(\boldsymbol{x})$ of the interface encoded by the volume fraction ϕ, where κ is evaluated from a local piecewise-linear interface construction (PLIC) of the interface [Youngs 1984; Lehmann 2019].

3.2 Discussion

From this brief recap of the current FSLBM approach, one can already appreciate that some of its commonly-used approximations may lead to numerical instability as soon as the fluid flow contains turbulent parts. For instance, the low-order evaluations of the equilibrium state in Eq. (5) (and similarly for the incorporation of the external forces in Eq. (6)) are ill-advised if the flow is even slightly turbulent. Similarly, the BGK-based collision model has been recognized as low-order accurate as soon as the inertia forces in the fluid are not small. Moreover, only considering a constant atmospheric pressure and the surface tension without accounting for the pressure coming from the lamellae between bubbles is bound to limit the range of bubbling and foaming that our fluid simulator can handle. Boundary handling for obstacles also cannot handle thin or non-closed objects, while this is common practice in single-phase LBM graphics simulators. Finally, recent advancements such as HOME-LBM have allowed for both reduced memory usage in the storage of the distribution functions and improved pressure behavior near obstacles. We now delve into our contributions, describing how the current FSLBM limitations are either completely removed or dramatically reduced and which changes we incorporate in order to result in a simulator with a significantly smaller memory footprint and quite a dramatic drop in computational complexity.

4 Our HOME-FREE LBM Solver

We now detail the various components of our new HOME-FREE LBM solver, starting with the fundamental changes in distribution function representation and collision operator, to our approach to fluid-solid coupling, as well as bubble and foam modeling.

4.1 Foundations of HOME-FREE LBM

Two of the most basic changes to FSLBM we propose have to do with memory efficiency and stability: we adopt a sixth-order collision model and the HOME-LBM encoding of the distribution function.

Collision model. In order to add stability and accuracy to our kinetic free-surface solver, we leverage the non-orthogonal central-moment multiple-relaxation-time model (NOCM-MRT) [De Rosis and Luo 2019] to avoid numerical instability in the presence of turbulent flows. The collision operator $\boldsymbol{\Omega}$ thus becomes

$$\boldsymbol{\Omega} = -\boldsymbol{M}^{-1}\left(\boldsymbol{R}(\boldsymbol{m} - \boldsymbol{m}^{\text{eq}}) + (\boldsymbol{I} - \tfrac{1}{2}\boldsymbol{R})\boldsymbol{K}\right), \tag{13}$$

where $\boldsymbol{M}$ is a matrix known in closed form as a function of the macroscopic velocity $\boldsymbol{u}$ that converts the distribution function into central-moment space through $\boldsymbol{m} = \boldsymbol{M}\boldsymbol{f}$, $\boldsymbol{R}$ is a diagonal matrix containing all individual relaxation rates r_i, and $\boldsymbol{K}$ represents the force terms projected into central-moment space. Following Li et al. [2020] for single-phase fluid simulation based on the lattice structure from Fig. 5 (d), we use a sixth-order Hermite expansion of the equilibrium distribution function, leading to most central-moment

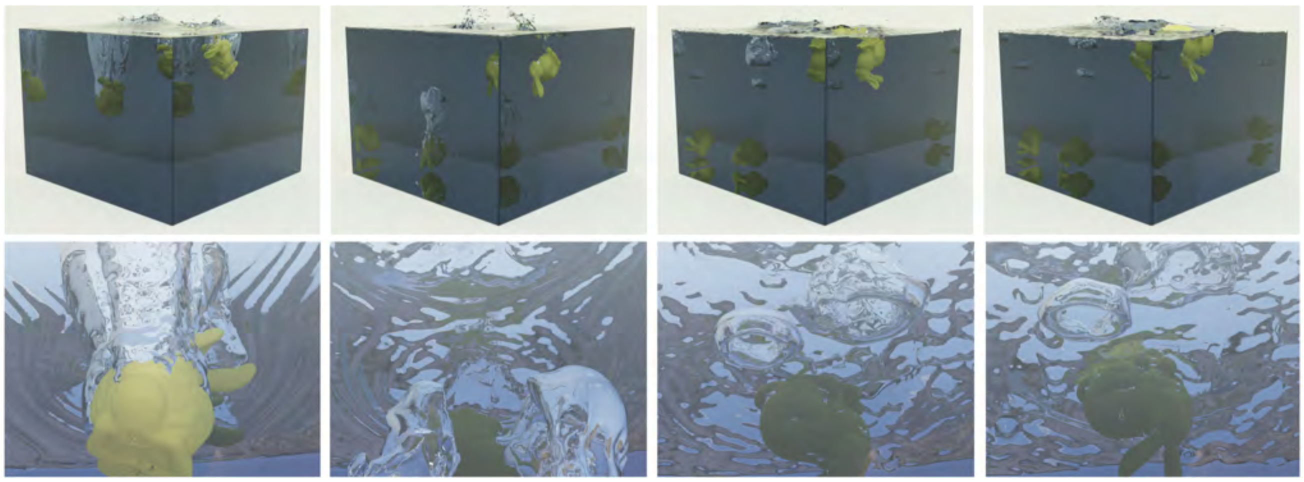


Fig. 7. **Bunny drops.** For a light and heavy bunny being dropped in a water tank (top), many bubbles are formed during impact, and the heavy bunny eventually falls to the bottom while the light one floats; bubble rings even appear, rising up and then bursting on the surface of the water (bottom).

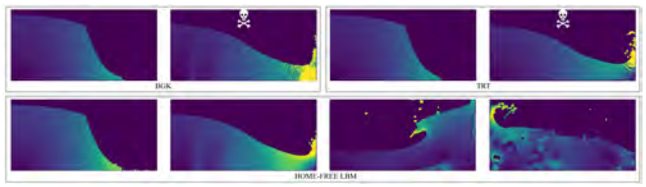


Fig. 8. **Comparisons between BGK, TRT, and HOME-FREE models.** For a simple 2D dam break, an FSLBM simulator using a BGK collision model crashes after 71 frames (top left); the TRT model allows the simulation to go a few frames further but, again, crashes at frame 75 (top right). Our HOME-FREE LBM simulator handles this case easily (bottom), never crashing.

terms vanishing except for:

$$m_0^{\text{eq}} = m_9^{\text{eq}} = \rho,\ m_{17}^{\text{eq}} = \tfrac{1}{3}\rho,\ m_{18}^{\text{eq}} = \tfrac{1}{9}\rho\ ,\ m_{26}^{\text{eq}} = \tfrac{1}{27}\rho, \tag{14}$$

while the forcing terms in the central moments turn into:

$$\begin{array}{lll} K_1 = F_x, & K_2 = F_y, & K_3 = F_z, \\ K_{10} = \tfrac{2}{3}F_x, & K_{11} = \tfrac{2}{3}F_y, & K_{12} = \tfrac{2}{3}F_z, \\ K_{23} = \tfrac{1}{9}F_x, & K_{24} = \tfrac{1}{9}F_y, & K_{25} = \tfrac{1}{9}F_z. \end{array} \tag{15}$$

Moment-encoding of distribution. Motivated by the recent introduction of the HOME-LBM framework [Li et al. 2023b], we replace the storage of the usual D3Q19 or D3Q27 lattice distribution functions f_i by only ρ, $\boldsymbol{u}$, and $\boldsymbol{S}$, for a total of 10 scalar values per grid node. Beyond obvious memory savings, the high-order reconstruction of the distribution functions from these terms and the computationally-simpler expressions of the collision operator bring added numerical stability as demonstrated in [Li et al. 2023b, 2024].

HOME-FREE solver loop at a glance. Our resulting HOME-FREE LBM solver thus proceeds as follows. First, we perform the streaming step just like in FSLBM, but the distribution function f_i from the neighboring node (the right term in Eq. (3)) is evaluated on the fly from the three velocity-moments of this node by a third-order Hermite-based "filtered" reconstruction through

$$f_i = \rho\, w_i \left[1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{\boldsymbol{H}^{[2]}(\boldsymbol{c}_i) : \boldsymbol{S}}{2c_s^4} + \sum_{\alpha\beta\gamma} \frac{H^{[3]}_{\alpha\beta\gamma}(\boldsymbol{c}_i) T_{\alpha\beta\gamma}}{2c_s^6} \right], \tag{16}$$

$$T_{\alpha\beta\gamma} = S_{\alpha\beta}u_\gamma + S_{\alpha\gamma}u_\beta + S_{\beta\gamma}u_\alpha - 2\, u_\alpha u_\beta u_\gamma. \tag{17}$$

where $\boldsymbol{H}$ represents Hermite polynomial bases and $\alpha\beta\gamma \in \{xxy, xyy, xxz, xzz, yzz, yyz, xyz\}$ denotes the Cartesian coordinate indices. We then convert the streamed distributions back into their temporary velocity-moments ρ^*, $\boldsymbol{u}^*$ and $\boldsymbol{S}^*$ via Eq. (7). Finally, the collision step, replacing Eq. (4), evaluates the final velocity-moments as:

$$\rho(\boldsymbol{x}, t+1) = \rho^*, \tag{18}$$

$$u_\alpha(\boldsymbol{x}, t+1) = u_\alpha^* + F_\alpha/(2\rho^*), \tag{19}$$

$$S_{\alpha\beta}(\boldsymbol{x}, t+1) = \left(1 - 1/\tau\right)S^*_{\alpha\beta} + \frac{1}{\tau}\, u^*_\alpha u^*_\beta + \frac{2\tau - 1}{2\tau\rho^*}\left(F_\alpha u^*_\beta + F_\beta u^*_\alpha\right), \tag{20}$$

$$\begin{aligned} S_{\alpha\alpha}(\boldsymbol{x}, t+1) = {}& \frac{\tau - 1}{3\tau}\left(2S^*_{\alpha\alpha} - S^*_{\beta\beta} - S^*_{\gamma\gamma}\right) + \frac{1}{3}\left({u^*_\alpha}^2 + {u^*_\beta}^2 + {u^*_\gamma}^2\right) \\ & + \frac{1}{3\tau}\left(2{u^*_\alpha}^2 - {u^*_\beta}^2 - {u^*_\gamma}^2\right) + \frac{1}{\rho^*}F_\alpha u^*_\alpha \\ & + \frac{\tau - 1}{3\tau\rho^*}\left(2F_\alpha u^*_\alpha - F_\beta u^*_\beta - F_\gamma u^*_\gamma\right), \end{aligned} \tag{21}$$

Incorporating both the HOME-LBM approach and a higher-order collision model into our kinetic free-surface method has already dramatic consequences on the stability of the solver: as Fig. 8 demonstrates, a basic dam break generates instabilities with the BGK or two-relaxation-time (TRT) collision model [Bogner 2017], but not with HOME-FREE LBM. The accuracy of our framework improves several aspects of the free-surface treatment, such as the interface pressure boundary condition from Eq. (11): it uses the filtered distribution function f_i reconstructed using Eq. (16) which offers a more accurate treatment. With these solid foundations in place, we can now focus on the remainder of our free-surface kinetic solver.

4.2 Efficient bubble model for thin shell

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

Existing treatment of bubbles in FSLBM. In traditional FSLBM, gas distribution function values are not simulated, but reconstructed on demand through Eq. (11), and gas pressure inside a bubble is always assumed to be the atmospheric pressure $p_g = p_{atmos} \coloneqq c_s^2$. This is very limiting, as bubble pressure should change due to bubbles growing, shrinking, and/or merging. More involved bubble models [Ataei et al. 2021] have been proposed where the volume change of a bubble b_i (defined as a connected region of gas and interface nodes surrounded by liquid) is recorded before and after one simulation step so that the pressure update can be evaluated based on the ideal gas law (for an isothermal simulation) as:

$$p(b_i,t) = p^{\text{atmos}} \frac{V(b_i,0)}{V(b_i,t)}, \tag{22}$$

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

$$V(b_i,t) = \sum_{\boldsymbol{x}\in b_i} (1-\phi(\boldsymbol{x},t)). \tag{23}$$

Handling bubbles thus needs to store the values used in Eq. (22) for each bubble, and to indicate for every non-liquid node the index of the bubble it corresponds to. Indices are updated at each time step by referring to the previous neighboring indices so as to track the motion of the bubbles. Merging and splitting of bubbles is even more involved: it first requires to detect splits and merges based on the connectedness of nodes of a given index, then to update the initial volumes and pressures of the newly created bubbles. A serial algorithm to achieve this task was proposed by Foley [1996] and Bogner [2017], involving local flood filling algorithms and bipartite graphs between consecutive arrangements of bubble indices. Alas, this approach is not parallelizable and does not scale well with the number of bubbles; moreover, it cannot accommodate thin-shell obstacles in the fluid, which are particularly desirable in graphics.

HOME-FREE bubbles. In order to create an efficient bubble model, we propose a new algorithmic approach to bubble tracking and updates which can be easily implemented on GPUs and which leverages the fact that LBM simulators advect a VOF value by *at most one grid node at a time*, which facilitates bookkeeping of bubble indices. First, we quickly check if some nodes switched from an F type (fluid) to a G or I type (gas or interface) over the last time step as it may mean either that the node just entered a bubble, or multiple bubbles are merging; we determine which case it is by simply storing and checking the indices of the nodes in the immediate vicinity. Similarly, we check if nodes switched from a G or I type to an F type as it may mean that these nodes left a bubble or a bubble is splitting; again, a simple exploration of the 27 indices immediately adjacent to the processed node allows us to find which situation is happening, and the indices of the bubbles at play are stored. The detection of either one of these cases triggers also a *connected component labeling* (CCL) to identify all the bubbles in the fluid — we use the parallel CCL from [Allegretti et al. 2019], which employs a fast block-based union-find data structure to identify the 3D connected interface/gas nodes and mark all connected ones with the same label. But unlike previous methods that go through the (algorithmically painful) exercise of matching old labels with new labels through bipartite graphs to be sure to properly track bubbles, we proceed very differently from there on. For each grid node that is of type G or I, we store both the current index i of the bubble it belongs to, and also the previous index i^{old} (note that this index is set for all G/I nodes, since even nodes freshly covered by a bubble know the previous bubble index of which they are now part due to the earliest part of our algorithm explained above); and we also keep all the (current and initial) volumes and (current) pressures of all the bubbles from the previous time step, denoted as V_j^{old}, $V_j^{0,\text{old}}$, and p_j^{old} respectively. We then update the volumes and pressures of all bubbles in a way that *does not require to know correspondences between old and new indices.* I.e., after initializing all the current and initial volume values to zero for all the identified bubbles, we go through each G/I node $\boldsymbol{x}$ with new and old bubble indices i and i^{old}, and perform the following volume updates (through atomic adds):

$$V_i \mathrel{+}= 1-\phi(\boldsymbol{x},t); \quad V_i^0 \mathrel{+}= (1-\phi(\boldsymbol{x},t))\, p^{\text{old}}_{i^{\text{old}}}/p^{\text{atmos}}. \tag{24}$$

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

$$p_i = p^{\text{atmos}} V_i^0 / V_i. \tag{25}$$

Dealing with cut-cell nodes. While existing FSLBM techniques do not allow for thin-shell obstacles, we can modify our approach slightly to handle this common case in graphics. We follow the cut-cell representation for thin shells in single-phase LBM [Lyu et al. 2021]; that is, a grid node for which one (or more) of its links intersect a boundary is marked as a *cut-cell* node — and we simply ignore cut-cell nodes during the CCL labeling phase. This amounts to considering the gas in a cut-cell region as being air at the atmospheric pressure. While this allows the use of thin-shell obstacles, it changes the treatment of grid nodes touching thick obstacles, as they would be treated in the CCL instead of being skipped. However, we show in Fig. 9 that this small simplification that allows for arbitrary obstacles does not generate visually noticeable differences for thick obstacles like a bunny in a dam break, validating our algorithmic change. Moreover, this pressure approximation near obstacles is actually more numerically robust as LBM is known to generate small spurious fluctuations in pressure near boundaries.

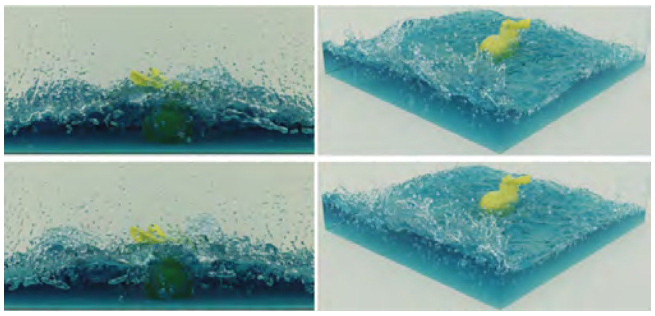


Fig. 9. **Ablation test for cut-cells.** As discussed in Sec. 4.2, we ignore cut-cell nodes during the CCL labeling phase. This simplification (top) only slightly alters the visual results when applied to a thick obstacle like the bunny in a dam break compared to the usual treatment (bottom), but now allows for thin obstacles seamlessly and without computational overhead.

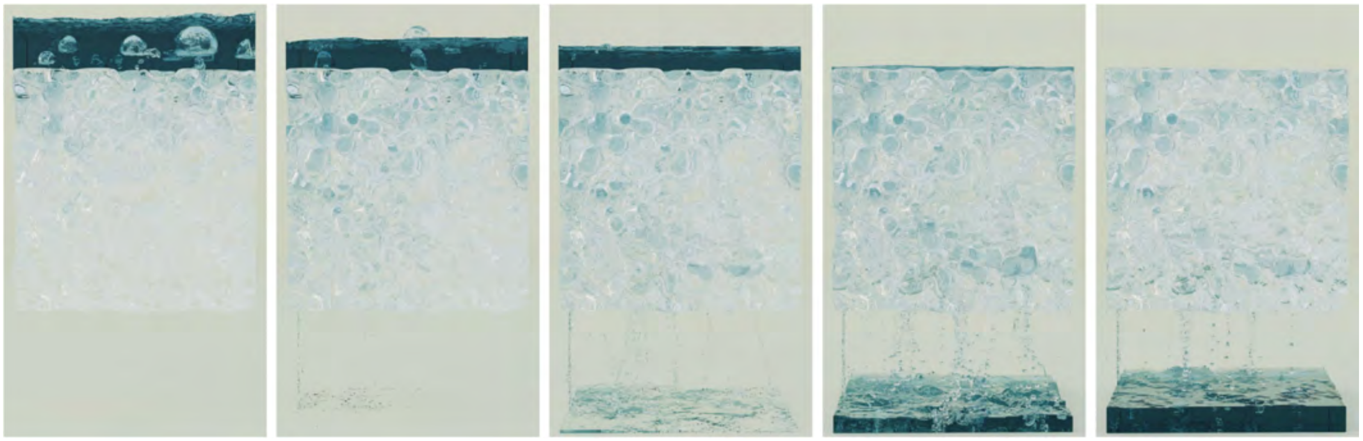


Fig. 10. **Water through porous material.** When water flows through the porous obstacle, bubbles form and rise to the surface as the flow runs through the complex tunnels and cavities, before exiting as liquid filaments and drops at the bottom.

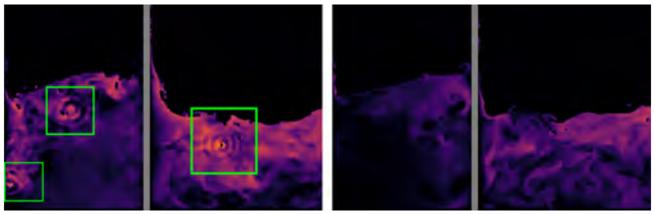


Fig. 11. **Ablation test for turbulence model.** Using a sectional view of the velocity field of Fig. 18 (where the comb is in grey), we see strong ringing artifacts around small bubbles (green boxes) when no eddy viscosity is used (left); the use of a turbulence model (right) removes these issues.

Turbulence model to tame small bubbles. When dealing with turbulent flows, very small bubbles bring additional challenges: as they often impose large pressures locally due to Eq. (25), strong ringing artifacts in the velocity field can appear. Consequently, velocities exceeding the CFL condition introduce errors in the advection of small bubbles, generating erratic displacements and causing some bubbles to disappear. These dispersive errors in the velocity field are often dealt with numerically through the addition of a turbulence model. We thus include *eddy viscosity* [Lesieur 1990] to all nodes less than four cells away from a bubble to prevent these issues. In practice, this means that we add to the usual fluid viscosity ν an eddy viscosity $\nu_e = 4\|S\|_F$, i.e., the eddy viscosity is set proportional to the Frobenius norm of the second velocity-based moment of the fluid to counteract any sharp local changes in the flow. Fig. 11 shows a sectional view for the velocity field from Fig. 18 with and without eddy viscosity: with a turbulence model, ringing artifacts disappear, and small bubbles do not behave spuriously.

4.3 Fluid-solid coupling

Another important part of any flow simulator is fluid-structure interaction, also called coupling.

Existing FSLBM coupling. In FSLBM, fluid-solid coupling is often handled via bounce-back [Ladd 1994]: for lattice edges crossing an obstacle boundary (Fig. 13), streaming is modified to read

$$f_{\bar{i}}^{*}(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) - 6 w_i \rho^s \boldsymbol{u}^s \cdot \boldsymbol{c}_{\bar{i}}. \tag{26}$$

in order to implement a "bounce" of the distribution function against the obstacle and a transfer of inertia due to the motion of the solid obstacle, where $\boldsymbol{u}^s$ is the macroscopic velocity of the solid node and ρ^s is the local fluid density at $\boldsymbol{s}$ intersection node. However, FSLBM uses the original bounce-back approach, which relies on nodes being tagged as inside or outside moving solids at each time-step [Bogner 2017] and thus cannot handle thin structures (Fig. 13(b)). Moreover, its first-order nature leads to spurious oscillations in turbulent flows. To transfer the effect of the fluid motion to the obstacle, FSLBM uses a moment-exchange formulation where a moment Δj_i is evaluated per lattice direction, and the coupling force and torque on the solid is the sum of all these contributions; that is,

$$\boldsymbol{F}_B \equiv \sum_{\boldsymbol{x}\in C_s} \sum_{i\in L_s} \Delta j_i(\boldsymbol{x}), \tag{27}$$

$$\boldsymbol{\tau}_B \equiv \sum_{\boldsymbol{x}\in C_s} (\boldsymbol{x} - \boldsymbol{x}_c) \times \sum_{i\in L_s} \Delta j_i(\boldsymbol{x}), \tag{28}$$

where C_s represents the set of all grid nodes adjacent to the solid $\boldsymbol{s}$ (orange nodes in Fig. 13 (b)), L_s is the lattice direction set towards the solid, while $\boldsymbol{x}_c$ is the barycenter of the solid. Note that FSLBM does not store a distribution function on gas nodes, so the gas pressure estimated from the equilibrium function assumes a zero velocity $\boldsymbol{u}_g = \boldsymbol{0}$. Therefore, if C_s contains gas, interface, and liquid nodes, the lattice moment Δj_i is evaluated through:

$$\Delta j_i(\boldsymbol{x}) = (f_{\bar{i}}^{*}(\boldsymbol{x}, t)\, \boldsymbol{c}_{\bar{i}} - f_i(\boldsymbol{x}, t)\, \boldsymbol{c}_i)\, \phi_{\boldsymbol{x}} + 2 f_i^{\text{eq}}(\rho_g, \boldsymbol{0})\, \boldsymbol{c}_{\bar{i}}\, (1 - \phi_{\boldsymbol{x}}), \tag{29}$$

where a blending between the two phases is achieved through a weight based on the local VOF value $\phi_{\boldsymbol{x}}$. Yet, similar to the bounce-back case, thin obstacles cannot be handled with this approach; moreover, Galilean invariance is not even preserved, which creates spurious artifacts in turbulent flows. We remedy these issues next.

Cut-cell treatment. Inspired by the double-sided bounce-back approach proposed in [Lyu et al. 2021], we first mark the links intersecting a solid boundary surface as *cut-cell links* (Fig. 13(a)), and any node containing a cut-cell link is marked as a cut-cell node (Fig. 13(c) in green). We then apply the bounce-back only on fluid and interface cut-cell nodes — not on gas nodes (nodes covered by the solid obstacle in Fig. 13(c)). This cut-cell boundary treatment naturally supports thin and non-watertight objects. We also incorporate the fluid-solid coupling treatment from HOME-LBM [Li et al.

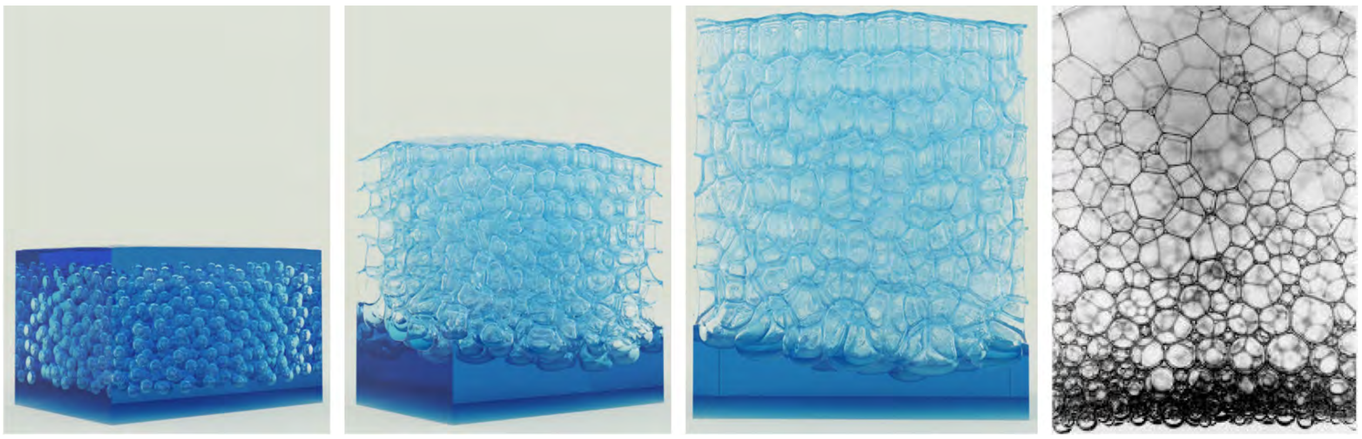


Fig. 12. **3D bubble drainage.** After bubbles are nucleated inside the liquid, foam is formed and bubbles grow as the dissolved gas in the liquid slowly seeps out. The drained liquid ends up at the bottom of the tank while a fairly regular lattice structure appears, as real-world bubble drainage examples (right) exhibit.

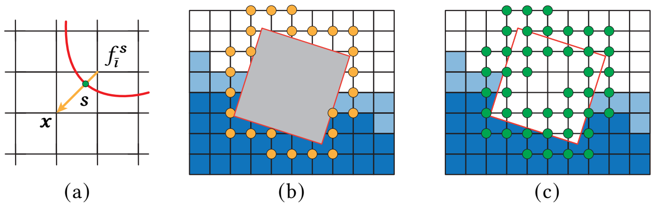


Fig. 13. **Cut-cell treatment of obstacles.** A cut-cell link (a) is a lattice edge (orange) intersecting the boundary of an obstacle, on which we store the intersection point s; cut-cell nodes for the regular bounce-back approach (b) are grid nodes that have cut-cell links (in orange); our double-sided bounce-back treatment uses all the cut-cell nodes (in green) with cut-cell links, even for those inside the obstacle.

2023b, Sec. 4.2]: the approximated distribution function from the solid boundary point $\boldsymbol{s}$ (at the intersection of a cut-cell link and a solid) that is streamed to a grid node $\boldsymbol{x}$ is reconstructed from Eq. (16) using $\rho=\rho_{\boldsymbol{x}}$, $\boldsymbol{u}=\boldsymbol{u}^s$, and $S_{\alpha\beta}=u^s_\alpha u^s_\beta+\left(S^{\boldsymbol{x}}_{\alpha\beta}-u^{\boldsymbol{x}}_\alpha u^{\boldsymbol{x}}_\beta\right)$.

Fluid-to-solid force exchange. To accommodate general obstacles, we modified Eq. (29) with a central moment scheme for F/I nodes to ensure Galilean invariance [Peng et al. 2016] through

$$\begin{aligned}\Delta j_i = {} & \left(f^*_{\bar{i}}(\boldsymbol{x},t)\,(\boldsymbol{c}_{\bar{i}}-\boldsymbol{u}^s)-f_i(\boldsymbol{x},t)\,(\boldsymbol{c}_i-\boldsymbol{u}^s)\right)\phi_{\boldsymbol{x}} \\ & +\left((\boldsymbol{c}_{\bar{i}}-\boldsymbol{u}^s)f^{\mathrm{eq}}_{\bar{i}}(\rho_g,\boldsymbol{0})-(\boldsymbol{c}_i-\boldsymbol{u}^s)f^{\mathrm{eq}}_i(\rho_g,\boldsymbol{0})\right)(1-\phi_{\boldsymbol{x}}),\end{aligned} \tag{30}$$

where $f^*_{\bar{i}}(\boldsymbol{x},t)$ is reconstructed from Eq. (16) with the three velocity-moments derived from the velocity of the obstacle as described above. Finally, the total force remains expressed as Eq. (27), while the total torque is changed to:

$$\boldsymbol{\tau}_B \equiv \sum_{\boldsymbol{x}\in C_s}(\boldsymbol{x}_s-\boldsymbol{x}_c)\times\sum_{i\in L_s}\Delta j_i(\boldsymbol{x}), \tag{31}$$

where C_s represents the set of all cut-set nodes (Fig. 13(c) in green), and $\boldsymbol{x}_s$ is an intersection point between the link and the obstacle boundary (point $\boldsymbol{s}$ in Fig. 13(a)), instead of Eq. (28) which used the current grid node. Note that in our implementation, we re-express Eqs. (27), (30) and (31) by noticing that $f^{\mathrm{eq}}_i(\rho_g,\boldsymbol{0})=w_i=w_{\bar{i}}$ and $\boldsymbol{c}_i=-\boldsymbol{c}_{\bar{i}}$, which yields the following expressions:

$$\begin{aligned}\Delta j_i &= \phi_{\boldsymbol{x}}(f^*_{\bar{i}}(\boldsymbol{x},t)+f_i(\boldsymbol{x},t)-2w_i)\boldsymbol{c}_{\bar{i}}-\phi_{\boldsymbol{x}}(f^*_{\bar{i}}(\boldsymbol{x},t)-f_i(\boldsymbol{x},t))\boldsymbol{u}^s+2\boldsymbol{c}_{\bar{i}}w_i,\\ \Delta\hat{j}_i &= \phi_{\boldsymbol{x}}(f^*_{\bar{i}}(\boldsymbol{x},t)+f_i(\boldsymbol{x},t)-2w_i)\boldsymbol{c}_{\bar{i}}-\phi_{\boldsymbol{x}}(f^*_{\bar{i}}(\boldsymbol{x},t)-f_i(\boldsymbol{x},t))\boldsymbol{u}^s,\\ \boldsymbol{F}_B &= \sum_{\boldsymbol{x}\in C'_s,i\in L_s}\Delta\hat{j}_i,\quad \boldsymbol{\tau}_B=\sum_{\boldsymbol{x}\in C'_s}(\boldsymbol{x}_s-\boldsymbol{x}_c)\times\sum_{i\in L_s}\Delta\hat{j}_i.\end{aligned} \tag{32}$$

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

Fresh/dead nodes. Finally, we follow the approach of Li and Desbrun [2023] to detect "fresh" and "dead" cells, i.e., cut-cell nodes which are suddenly not covered by an obstacle and cut-cell nodes which become covered by an obstacle respectively. For dead nodes, we simply tag the node as a gas node. Fresh nodes require more care. Bogner [2017] proposed a method based on the neighboring non-solid cells: fresh nodes are treated as fluid cells if there are no gas or interface nodes around; otherwise, fresh nodes are tagged as gas nodes. However, our tests show this direct assignment based on neighbors is too drastic: this strategy often causes the liquid to stick to moving obstacles, leading to obvious visual artifacts. We propose a new strategy to "ease in" the fresh nodes into the simulation. When a node is detected as fresh, we first set the local VOF value ϕ by averaging the neighboring nodes' VOF. If the ϕ is lower than a threshold θ, the fresh node is tagged as gas; otherwise, it is tagged as a fluid node, and the density ρ is interpolated from the neighboring fluid and interface nodes, while the velocity $\boldsymbol{u}$ is taken from the nearby solid boundary which triggered the change of this node's status, and the second-order moment $\boldsymbol{S}$ is approximated via $S_{\alpha\beta}=\boldsymbol{u}_\alpha\boldsymbol{u}_\beta$, so that a distribution function can be reconstructed for this new fluid node. Note finally that we make our threshold θ depend linearly on the velocity of the solid objects: if the solid object moves fast, the threshold θ should be set to a large value (0.95) since the fresh nodes cannot be filled quickly with fluid due to the large relative velocity; on the contrary, if the solid object moves slowly, a small threshold θ is used (0.3), to model the fact that the fluid can fill the fresh nodes quickly. As Fig. 22 demonstrates, this treatment removes the usual sticking artifacts of previous methods.

4.4 Foam modeling

Foam, where many gas bubbles are surrounded by thin films (lamellae) of liquid, is a highly recognizable phenomenon often seen on beaches and coffee cups. Capturing this phenomenon is impractical for diffuse phase models such as [Li et al. 2022], as the proper handling of thin films would require an extremely fine grid. Ataei et al. [2021] showed that one can model this complex phenomenon

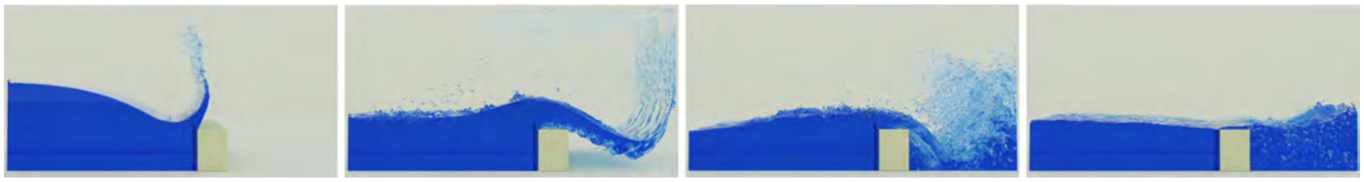


Fig. 14. **Dam break against a small wall.** Initializing a water column next to a small horizontal parallelepiped-shaped obstacle, the resulting dam break flow hits the low wall hard, which generates large splashes and many bubbles until the water comes to rest.

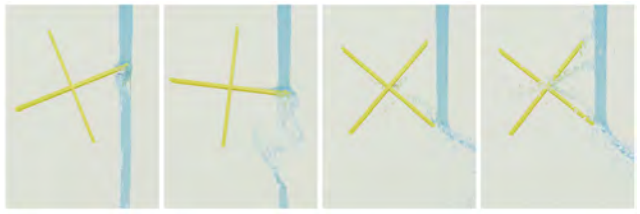


Fig. 15. **Water wheel.** In this two-way coupling example, water flows down onto a wheel, making it spin. Our approach evaluates the forces involved in this interaction five times faster than the work of Bogner [2017].

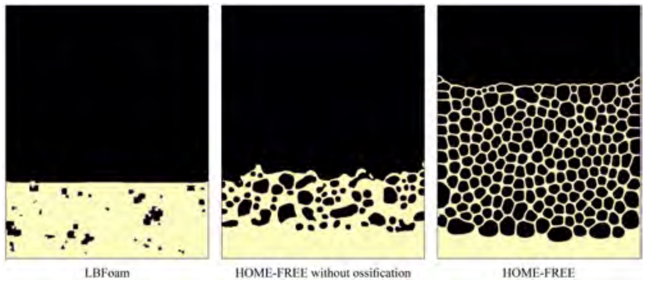


Fig. 16. **Comparisons on 2D foam.** We test our foam modeling on bubble nucleation in 2D at $Re = 100,000$. While [Ataei et al. 2021] crashes quickly (left), our solver generates the expected foaming drainage where the liquid flow within the foam is driven by gravity and capillarity (right). Foam lamellae do not persist without our foam-ossifying viscosity (middle).

with FSLBM by *solving an advection-diffusion equation for the content of dissolved gas in the liquid* and *modeling disjoining pressure* which arises from the attractive interaction between two bubbles. We revisit and adapt this approach to our framework next to make it robust, once again, to more turbulent cases.

Advection-diffusion equation. Ataei et al. [2021] proposed to model advection and diffusion of a field representing the dissolved gas concentration φ (in terms of mass fraction) using

$$\frac{\partial \varphi}{\partial t} + \nabla \cdot (\varphi \boldsymbol{u}) = \nabla \cdot (\mu_g \nabla \varphi) + q, \tag{33}$$

where q is a source term responsible for gas generation due to, for example, chemical reactions, and μ_g is the diffusion coefficient. This PDE is solved through LBM discretization as well, using another distribution function $\boldsymbol{g}$ on a D3Q7 lattice structure this time (with lattice directions $\boldsymbol{d}_i$, weights $w_0^g = 1/4$ and $w_i^g = 1/8$ for $i \in \{1, ..., 6\}$, with speed of sound $d_s = \frac{1}{2}$, see Fig. 5) as it does not require the same accuracy as for the fluid. The lattice Boltzmann equations for the distribution function values g_i then read

$$g_i(x + \boldsymbol{d}_i, t + 1) - g_i(x, t) = \Omega_i^g + q_i, \tag{34}$$

where $q_i = w_i^g q$ and $\Omega_i^g = -(\boldsymbol{g}_i - \boldsymbol{g}_i^{\text{eq}})/\tau_g$ for the BGK collision model (sufficiently accurate for this advection-diffusion simulation) where the relaxation time is linked to the diffusion constant via $\tau_g = 3\mu_g + \frac{1}{2}$. As usual, the zeroth-order moment of the distribution function g allows for the reconstruction of the gas concentration φ, i.e.,

$$\varphi(x, t) = \sum_{i=0}^{6} g_i(x, t). \tag{35}$$

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

$$\boldsymbol{g}_i^{\text{eq}}(x, t) = w_i \varphi(x, t)(1 + \boldsymbol{d}_i \cdot \boldsymbol{u}/d_s^2). \tag{36}$$

Ataei et al. [2021] also proposes to account for the gas diffusing into the bubble volumes by adding to all bubble initial volumes:

$$V_i^0 \mathrel{+}= \frac{\rho}{p^{\text{atmos}}} \sum_{x \in b_i} \Bigg(\sum_{x + c_i \in F} [g_{\bar{i}}(x + c_i, t) - g_i(x, t)] \\ - \varphi(x, t)[\phi(\boldsymbol{x}, t) - \phi(\boldsymbol{x}, t - 1)] \Bigg), \tag{37}$$

where V_i^0 is the initial volume of bubble b_i as discussed in Sec. 4.2. Finally, the concentration of gas in a bubble interface is known to obey Henry's law [Henry 1832] in foams, which expresses a linear relationship between gas concentration and bubble pressure:

$$\varphi(x, t) = k_H p_g. \quad \forall x \in I, \tag{38}$$

where k_H is Henry's law constant and p_g is the bubble pressure. One can thus use this property as a way to set boundary conditions similar to Eq. (11) for the free surface treatment through:

$$g_i^*(x, t) = g_i^{\text{eq}}(\varphi(x, t), \boldsymbol{u}) + g_{\bar{i}}^{\text{eq}}(\varphi(x, t), \boldsymbol{u}) - g_i(x, t). \tag{39}$$

Disjoining pressure. A disjoining pressure Π has also been proposed to stabilize the lamellae between bubbles, based on the theory of thin liquid films, in order to allow the formation of foam [Derjaguin and Churaev 1978]. This disjoining pressure shows a linear dependence in the distance d between two interfaces of bubbles and starts acting from the maximum distance d_{max} (typically set to 4 grid-cell widths) to minimize the variation of Gibbs free energy for two interfaces [Huber et al. 2014], yielding:

$$\Pi = \begin{cases} 0 & \text{if } d > d_{max} \\ k_\pi(1 - d/d_{max}) & \text{if } d \leq d_{max}, \end{cases} \tag{40}$$

where k_π is the disjoining pressure strength. This disjoining term gets added to the pressure and surface tension in Eq. (12) to become:

$$\rho_g = (p_g - 2\gamma\kappa(x) - \Pi)/c_s^2. \tag{41}$$

New foam-ossifying treatment. However, the foam model above can only handle low Reynolds number foam cases and will crash in turbulent cases, as Fig. 16 (right) at $Re = 100,000$ demonstrates. Increasing surface tension is sometimes advocated to further stabilize bubble shapes, but we found in our experiments that high surface tension has a tendency to drastically slow down small bubbles due to the large surface tension forces overpowering the bubble's motion — thus adversely affecting the simulation away from foams.

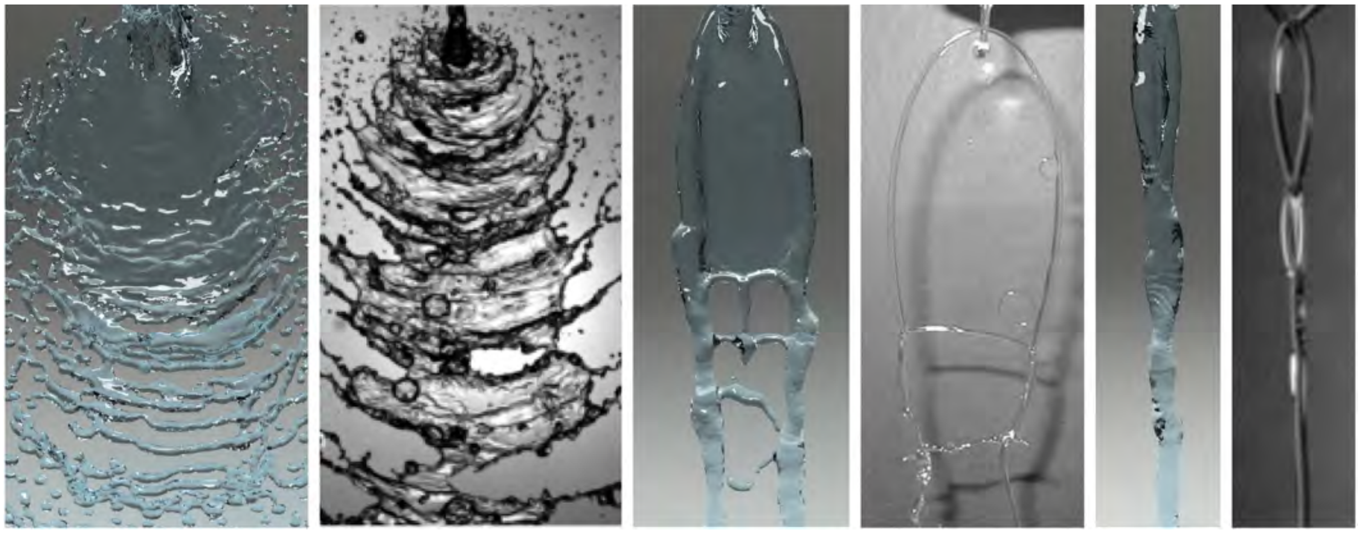


Fig. 17. **Impinging jets.** Our HOME-FREE solver exhibits similar patterns of thin films and chain-link shaped streams for impinging jets (here, for three decreasing initial velocities) when compared to real-world impinging jets (black-and-white photos from Bush and Hasha [2004] and Pruitt et al. [2024]).

Thus, we make specific alterations to our solver to improve foam modeling without affecting bubble dynamics. First, we employ the D3Q7 Central-Moment-Relaxation (CMR) collision model from [Li et al. 2022] to improve the accuracy and stability of Eq. (34), where the collision operator is expressed as:

$$\Omega^g = -(\boldsymbol{M}^g)^{-1}\boldsymbol{R}^g\boldsymbol{M}^g(\boldsymbol{g}-\boldsymbol{g}^{\text{eq}}) + (\boldsymbol{M}^g)^{-1}(\boldsymbol{I}-\tfrac{1}{2}\boldsymbol{R}^g)\text{Q}, \quad (42)$$

where the matrix $\mathbf{M}^g$ transforms the distribution function $\boldsymbol{g}$ and the source term q_i into central-moment space with $\boldsymbol{Q}=\mathbf{M}^g\boldsymbol{q}$, which enforces Galilean invariance. The diagonal relaxation matrix, written $\mathbf{R}^g=\text{diag}(1, 1/(4\mu_g+½), 1/(4\mu_g+½), 1/(4\mu_g+½), 1, 1, 1)$, has separate relaxation rates of each moment, reducing the instabilities that the BGK model creates. The closed-form expressions of the coefficients of $\mathbf{M}^g=(\text{M}_0^g, \text{M}_1^g, \text{M}_2^g, \text{M}_3^g, \text{M}_4^g, \text{M}_5^g, \text{M}_6^g)^T$ were given in [Li et al. 2022], and are expressed as

$$\text{M}_0^g = (1,1,1,1,1,1,1)^T,$$
$$\text{M}_1^g = \overline{d}_{i,x}, \quad \text{M}_2^g = \overline{d}_{i,y}, \quad \text{M}_3^g = \overline{d}_{i,z}, \quad \text{M}_4^g = \overline{d}_{i,x}^2 - \overline{d}_{i,y}^2,$$
$$\text{M}_5^g = \overline{d}_{i,x}^2 - \overline{d}_{i,z}^2, \quad \text{M}_6^g = \overline{d}_{i,x}^2 + \overline{d}_{i,y}^2 + \overline{d}_{i,z}^2.$$

where $\overline{d}_{i,\delta} = (\boldsymbol{d}_i - \boldsymbol{u})_\delta$, with $i \in \{0,...,6\}$ being the lattice index and $\delta \in \{x,y,z\}$ the coordinate. Second, we introduce a local viscosity ν_f for all nodes less than six cells away from bubble interfaces. This helps with the ossification of bubble lamellae but does not affect the rest of the flow. Without this foam-ossifying viscosity, foams will not keep their structure, collapsing quickly at such high Reynolds numbers (Fig. 16). Finally, we locally adjust the surface tension γ based on the disjoint force value Π through

$$\gamma = (\Pi{=}{=}0)\,?\,\gamma \,:\, \gamma_{\text{oss}}, \quad (43)$$

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

ALGORITHM 1: Pseudocode of our HOME-FREE solver

```
Data: $\phi$, $\rho$, $\boldsymbol{u}$, $S$, $\varphi$, $q$, $\boldsymbol{g}$, flag
Result: Simulation sequence
$t \leftarrow 0$;
Provided $\phi$, $\varphi$ $\rho$, $\boldsymbol{u}$, $q$ and flag, initialize the velocity moments $\boldsymbol{S}$ and
  the distributions $\boldsymbol{g}$ in advection-diffusion;
while $t < T$ do
    1: ResetCutCell() [Sec. 4.3];
    2: ComputeDisjoinPressure() [Sec. 4.4, Eq. (40)];
    3: Streaming and apply free-surface boundary conditions
       [Sec. 3.1, Eq. (11)];
    4: Compute two-way force [Sec. 4.3, Eq. (32)];
    5: HOME collison() [Sec. 4.1, Eqs. (18)-(??)];
    6: HOME-FREE bubble update [Sec. 4.2, Eqs. (24), (25)];
    7: Fresh/dead nodes update [Sec. 4.3];
    8: Advection-Diffusion() [Sec. 4.4, Eqs. (34)-(39)];
    $t \leftarrow t + 1$;
end
```

5 Results

We now go through our tests and results to demonstrate the multiple benefits of our approach. We cover a number of simulation experiments with our solver, including bubbles, foam, and interactions with solid objects, as well as comparisons with existing methods and real-world experiments. We also provide various ablation studies to experimentally demonstrate the efficacy of our contributions. Note that our implementation is publicly available at https://github.com/qingxu-thu/Home-FSLBM.

5.1 Implementation details

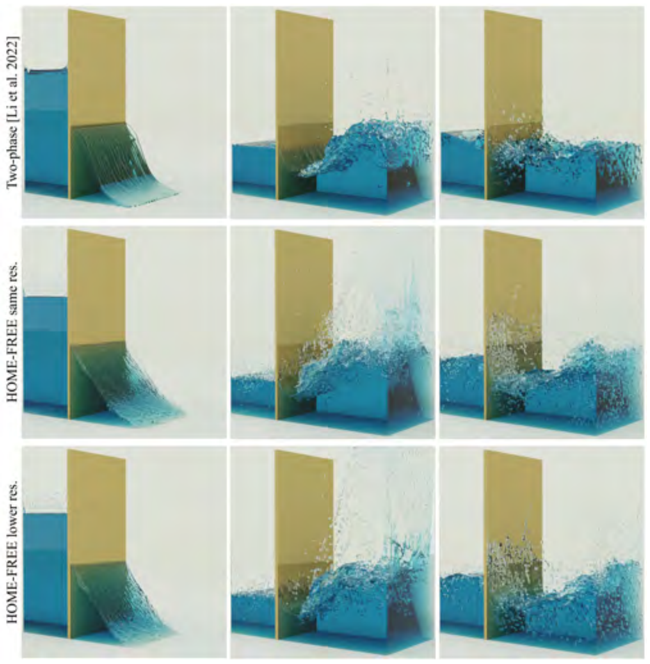
We implemented our kinetic free-surface solver in C++ and CUDA, using a structure-of-arrays (SoA) data structure to store two copies of 10 variables (the HOME velocity-moments) per grid node in order to allow for time integration (see Alg. 1). For curvature estimation of the interface, we follow the PLIC implementation of [Lehmann

2019], which relies on a Parker-Youngs approximation for the normal [Pohl 2008]. We leverage the CCL implementation from [Allegretti et al. 2019; Bolelli et al. 2020; Bolelli 2023] with a block size kernel of 8×8×4 and mark the labels in a sequential order. Bubble volumes are computed in double precision to ensure accuracy; but the amount of bubbles remains smaller than the number of grid nodes, so this format does not incur a large memory burden. For intersection detection through cut-cell links, we follow the approach of [Li and Desbrun 2023] by first constructing a bounding-volume hierarchy tree structure for the 3D mesh model on the GPU; we also adopt cut-cell flags and bounding boxes to accelerate link-mesh intersection. For the disjoining pressure, we use the tracing method mentioned in [Ataei et al. 2021] with a Parker-Youngs approximated normal; we then trace along the normal to search and identify whether there is another bubble around. Conversion between physical units and LBM units follows the procedure described in the supplementary materials of Li et al. [2020], with the iteration counts chosen to correspond to a frame rate of $\frac{1}{30}$s. All results were run on NVidia GeForce RTX 3090 GPU cards with 24GB of memory. We also used GPUs to render the resulting interface mesh using [van Bergen 2023] and Cinema 4D; note that after extracting the sharp VOF interface, we perform a slight Laplacian smoothing and remesh it with Blender [Blender Online Community 2018] to reduce aliasing artifacts. Detailed statistics, including timings profiled with Nsight, parameters, and other setups are presented in Tab. 1. It is also worth noting in Tab. 1 that we only utilize the foam-ossifying viscosity ν_f in Figs. 6 and 12 to better stabilize complex membrane structures and foam dynamics at high Reynolds numbers.

5.2 Comparisons

We first provide comparisons to current state-of-the-art solvers, before providing further validation of our approach through comparisons with real-life experiments.

Comparison with diffuse-interface LBM solvers. In order to evaluate how our sharp-interface free-surface solver compares to existing diffuse-interface multiphase LBM solvers, we compare our approach to [Li et al. 2022] on their example of the water dam break through a thin "comb" structure in Fig. 18. with the same resolution, gravity and viscosity parameters. The water flows through the interstices of the comb, causing large splashing and many bubbles. We observe that our free-surface simulation flows more freely through the thin structures while the fluid from [Li et al. 2022] is slowed down due to its non-sharp interface. Moreover, our method generates far more bubbles and splashing with more turbulent details, while the multiphase solver seems to introduce numerical viscosity in comparison due to its diffuse encoding of the phase. Perhaps more importantly, our solver performs this animation sequence two times faster and uses only 60% of the memory size required by the multiphase solver, proving its efficiency. For a smaller resolution, our solver still captures more turbulent details and bubbles than [Li et al. 2022], this time nearly 4.4 times faster and for only 25% of the memory usage. We can also compare to *non-kinetic* solvers, as this same example was used in [Chen et al. 2020b]: while they report 90 sec/frame for a resolution of 64×64×32 with a GeForce GTX 1080Ti card, we achieve 31 sec./frame on a RTX 3090 card for a resolution of 400×400×200.



Memory usage	$(\rho,\boldsymbol{u},S)$	cutcell	flag	ϕ	bubble info	mass	$\boldsymbol{f}$	$\boldsymbol{n}$	$\boldsymbol{F}$	total
Ours	20	1	1	2	3	2	0	0	0	29
[Li et al. 2022]	5	1	1	1	0	0	34	3	3	48

Fig. 18. **Comb dam.** Compared with the two-phase solver [Li et al. 2022] (top) which is twice slower and uses 1.7 times as much memory as ours for the same resolution (bottom), our solver captures much smaller bubbles due to its sharp-interface representation. For a lower resolution (middle), our solver also shows finer fluid details while being now 4.4 times faster for 4 times less memory use than [Li et al. 2022]. Note that *bubble info* represents the amount of data for bubble indexing, CCL, and merge detection.

The use of a kinetic solver thus brings orders of magnitude speed-up, allowing for obviously far more fluid details in the simulation.

Comparison with real-world impinging jet. To better characterize the performance of our method, we also compare the results of our HOME-FREE solver against real-world impinging jets for both viscous and turbulent scenarios. We simulate two jet streams colliding with different surface tension and viscosity coefficients to explore the different shapes of the resulting jet as suggested in [Bush and Hasha 2004]. For low Reynolds number (Re=80) and low Weber number (We=57.6) as in Fig. 17 (right), we observe a chain link forming, where the stream is twisted due to the influence of surface tension and high viscosity. If we increase the Weber number (We=136.5) and the Reynolds number (Re=150), we observe in Fig. 17 (middle) an open rim with a thin layer of liquid that breaks up under the influence of gravity and surface tension. In Fig. 17 (left) showing even higher Reynolds and Weber numbers (Re=1,280 and We=20,480), we now observe a very turbulent flapping sheet phenomenon, with a periodic impact wave and a periodic splashing pattern. For each example, a snapshot of our result is compared with a real-world experiment for similar physical parameters taken from [Bush and Hasha 2004; Pruitt et al. 2024]. Compared to other published numerical methods such as [Da et al. 2016], our results exhibit liquid

Table 1. **Statistics.** All examples timed on a NVIDIA RTX 3090 for a frame representing 1/30s.

Figure	Resolution	Sec./frame	Iters/frame	ν	$\gamma(,\gamma_{oss})$	k_π	ν_e	ν_f	q
Fig. 1	600 × 300 × 300	58.1	320	0.0001	0.0002,0.004	0.032	$4\|S\|_F$	N/A	0
Fig. 2	200 × 400 × 200	16.4	320	0.0001	0	0	N/A	N/A	N/A
Fig. 3	600 × 300 × 300	58.1	320	0.0001	0.00003,0.0002	0.032	$4\|S\|_F$	N/A	0
Fig. 4	400 × 300 × 400	80.6	320	0.0003	0	0	$0.8\|S\|_F$	N/A	N/A
Fig. 6	400 × 400 × 400	130.6	240	0.0001	0.005	0.032	N/A	0.13	0.00002
Fig. 7	250 × 300 × 250	13.2	100	0.0001	0	0	N/A	N/A	N/A
Fig. 9	400 × 200 × 400	26.1	320	0.0001	0	0	N/A	N/A	N/A
Fig. 10	200 × 400 × 200	9.5	320	0.0001	0	0	N/A	N/A	N/A
Fig. 12	300 × 600 × 200	46.8	320	0.0001	0.005	0.04	N/A	0.2	0.00003
Fig. 14	400 × 200 × 200	14.5	320	0.0001	0	0	$4\|S\|_F$	N/A	N/A
Fig. 15	250 × 300 × 250	8.4	333	0.0001	0	0	N/A	N/A	N/A
Fig. 17 (left)	200 × 400 × 400	3.5	320	0.003	0.0001	0	N/A	N/A	N/A
Fig. 17 (median)	200 × 400 × 400	3.2	320	0.012	0.001	0	N/A	N/A	N/A
Fig. 17 (right)	200 × 400 × 400	2.9	320	0.017	0.003	0	N/A	N/A	N/A
Fig. 18 ([Li et al. 2022])	400 × 400 × 200	57.6	320	0.0001	0	N/A	N/A	N/A	N/A
Fig. 18 (same res.)	400 × 400 × 200	31.2	320	0.0001	0	0	$4\|S\|_F$	N/A	N/A
Fig. 18 (lower res.)	300 × 300 × 150	13.0	320	0.0001	0	0	$4\|S\|_F$	N/A	N/A
Fig. 19	512 × 120 × 120	0.04	20	0.1	0.1	0	N/A	N/A	N/A
Fig. 20	400 × 300 × 400	78.6	320	0.001	0	0	N/A	N/A	N/A
Fig. 21	400 × 120 × 1200	36.2	133	0.0001	0	0	N/A	N/A	N/A
Fig. 22	250 × 300 × 250	11.3	100	0.0001	0	0	N/A	N/A	N/A
Fig. 23	300 × 300 × 300	30.1	250	0.0001	0	0	N/A	N/A	N/A

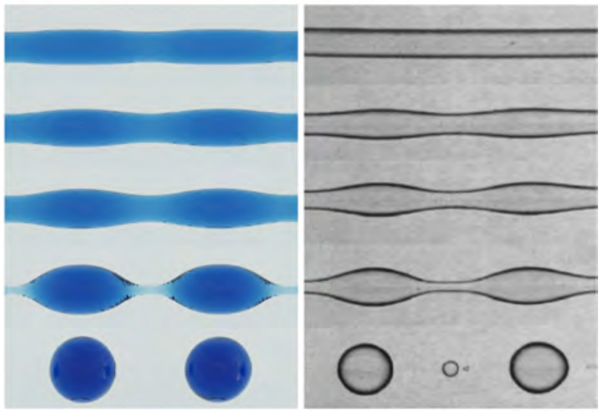


Fig. 19. **Plateau-Rayleigh instability.** Our HOME-FREE solver simulates a 3D Plateau-Rayleigh instability for a thin stream of fluid (top) getting thinner, before breaking into droplets (bottom). Ours (left) reproduces the real-world experiments provided by Hagedorn et al. [2004].

sheets, ligaments and splashing more in line with the experiments, especially for high Reynolds numbers. Note also that the original FSLBM is unable to simulate the most turbulent case without crashing. This demonstrates the experimental accuracy of our method at different regimes of viscosity and surface tension. Finally, the non-kinetic approach of [Chen et al. 2013] (which used adaptively refined grids) required timings of the order of two months on 48 3-Ghz processors, while ours only take 3 seconds per frame.

Comparison with real-world bubble drainage. We also provide in Fig. 12 a 3D version of Fig. 16, showing foam growth as the water is drained towards the bottom of the tank. The foam shows a regular lattice structure, exhibiting very close similarities with real-world drainage structures [Ataei et al. 2021; Saint-Jalmes 2006]. We set viscosity at 1e-4 (LBM unit), with a local viscosity ν_f around bubble structures set to 2e-1. We perform bubble nucleation through Poisson sampling [Ataei et al. 2021] and use a constant source q=3e-5 in the advection-diffusion equation to force foam growth.

Comparison for Plateau-Rayleigh instability. We test our approach on the surface-tension driven Plateau-Rayleigh instability: a thin stream of water gets increasingly thinner and, at some point, breaks up into individual droplets. We use a cylinder of fluid (9 times as long as its radius) with an initial sine wave perturbation of magnitude 0.05 times the radius [Breslouer 2010]. We use a viscosity of 0.1 in LBM units and a surface tension coefficient of 0.1, with no initial velocity and no gravity. After a few timesteps, the perturbation grows before the cylinder breaks up into several droplets, in agreement with the experiment results found in [Hagedorn et al. 2004].

5.3 Bubbles and foam phenomena

Bubbles and foams in free-surface simulation are particularly challenging at high Reynolds numbers due to the fast motion of the complex structure of the free surface. We now review various examples of bubbles and foam phenomena, including foam, bubbles, and bubble rings, to show the performance of our solver in these difficult scenarios that FSLBM cannot handle.

Glugging. Fig. 2 shows that our solver captures glugging using the typical two-chamber setup with an opening in between. The water from the top chamber falls into the bottom chamber, and air from the bottom container rises up through the opening to create the usual bubbling and glugging seen in water coolers. Even with a fairly low resolution, splashing shows many details and bubbles travel around for a while. Again, this turbulent flow with bubbles could not be performed with regular FSLBM.

Pouring and foaming. Fig. 3 shows water being poured into a tank with a high velocity, which stirs the water already in the tank and generates numerous bubbles and foam. Note that this case uses a high Reynolds number (Re=300,000), impossible to capture with FSLBM. A small surface tension coefficient is used as well as a disjoining pressure, while the advection-diffusion equation uses no source term. This sharp-interface simulation captures fine details of the foam and associated splashing. A similar setup was also used in [Karnakov et al. 2022, Fig. 6], and their simulation took about

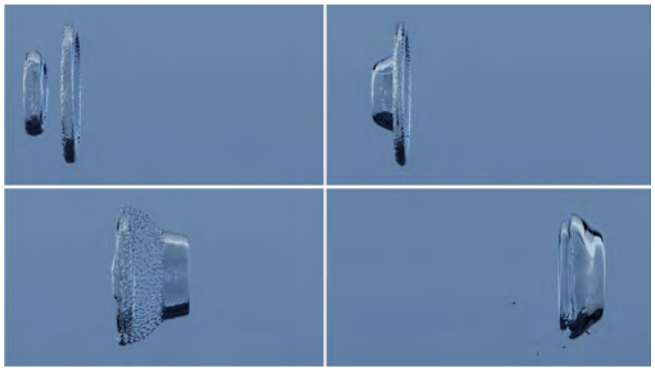


Fig. 20. **Leapfrogging bubble rings.** HOME-FREE succeeds in capturing bubble ring leapfrogging effects, creating at times a veil of bubbles.

24 hours using 13,824 cores on the Piz Daint supercomputer for a resolution of 768×384×384 grid; instead, ours took 36 mins for a resolution of 600×300×300 to get similar visual results.

Bubble rising. Fig. 6 shows bubbles rising in a tank of water. The bubbles are generated at the bottom of the tank, rising and growing until they end up floating on top of the water. The bubbles form a foam-like structure, with many fine lamellae which stay robustly in place due to our locally increased viscosity and surface tension. Here again, we use a rather high Reynolds number (Re = 120,000) to simulate a realistic bubble rising speed. A surface tension with Weber number We = 72 is also used to keep a circular shape, together with a disjoining pressure to repulse the bubbles from each other. The advection-diffusion equation uses an extra chemical source term (q = 2e-5) to induce slow bubble growth.

Bubble ring. We also demonstrate the ability of our method to deal with complex bubble phenomena in Fig. 4 by simulating the evolution of a bubble ring in a tank of water. The rings rise up, deform, split, and reconnect. After connecting, a small bubble is ejected due to the rapid motion of the resulting bubble ring. Compared to previous bubble ring simulations in [Padilla et al. 2019; Xiong et al. 2022] for instance, we use a much higher Reynolds number (Re=26,000). We simply initialize the flow with a curl-free velocity field tangent to a torus with air inside the torus; our free-surface solver manages to maintain this vortical structure in time.

Bubble leapfrogging. We also show an example of bubble leapfrogging in Fig. 20. Two bubble rings are initialized, and the bubbles end up leapfrogging each other twice, before finally merging into a larger one. Interestingly, the first ring splits into a veil of tiny bubbles before merging into the second ring. Similar to the bubble ring case, we use a high Reynolds number (Re = 12,000), with no foam-ossifying viscosity.

5.4 Fluid interaction with static objects

We also tested our approach in the presence of static obstacles.

Water through porous material. To test our method on complex geometry, we simulate the motion of water as it flows through a porous structure in Fig. 10. Initially, bubbles from the porous material rise up as in the glugging example while the water begins to make its way through the various openings; then the water runs through the complex tunnels and cavities, before exiting as liquid filaments and drops at the bottom, as expected.

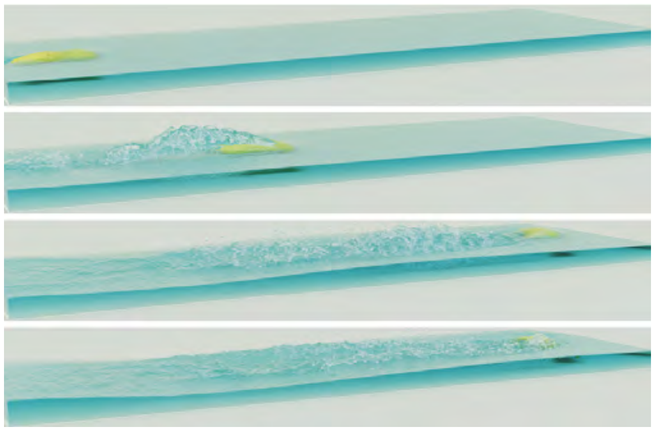


Fig. 21. **Car splashing.** As a car moves through a large body of water fast, it generates bubbles and splashes in front of the car, while the wake formed behind the car exhibits turbulence.

Dam break on obstacle. We tested the effects of water on an obstacle in the usual dam break flow problem. We initialize a horizontal parallelepiped obstacle next to a water column, so that the dam break flow hits the obstacle and generates large splashes and bubbles. Fig. 14 shows the water accumulating behind the obstacle leading to complex turbulence with small bubbles forming and breaking before the water slowly comes to rest.

Water drop. We also provide a real-time test for our HOME-FREE solver in the accompanying video: we simulate a ball of water falling into a tank of water, generating splashes. Our simulation runs at 20 fps (50 ms/frame) for a resolution of 200 × 100 × 200. We do not include bubble modeling, and viscosity is set to ν = 5e-5 to exhibit turbulent effects. This result shows promise for real-time applications on consumer-grade GPU cards.

5.5 Fluid interaction with moving object

We demonstrate one-way and two-way interactions next.

Car splashing. Fig. 21 shows a car moving through a large body of water. In this one-way coupling simulation, bubbles and splashes are clearly visible in front of the car, and turbulence forms behind the car. We can also view splashing caused by the car's skidding.

Bunny drops. We also demonstrate two-way coupling, where two bunnies (one heavy, one light) are dropped in a tank of water in Fig. 7. The bunnies hit the water with a splash, after which the heavy bunny sinks to the bottom of the tank, surrounded by bubbles trapped by the water being displaced rapidly during the fall, while the light bunny bobs about as it floats on the water surface. Note that the bubbles generated from the heavy bunny form two bubble rings, which gradually enlarge and rise up to the surface.

5.6 Ablation study

Finally, we perform ablation studies to further illustrate the effects of our several alterations to FSLBM. We previously showed the effectiveness of our HOME-based solver over a traditional FSLBM solver in handling turbulence (Fig. 11), in foam growth (Fig. 16), and demonstrated in Fig. 9 that our simplified bubble treatment is effective. We now discuss the advantages of our cut-cell based two-way force computation and fresh-node update in more details.

Two-way force computation. As explained in Sec. 4.3, we adopt Eq. (32) instead of Eq. (29) to compute two-way forces. Our approach will thus be (at times, much) more efficient than [Bogner 2017] when there is a small amount of fluid in the computational domain and a high cut-cell node proportion. For instance, in the case of the water wheel fall in Fig. 15, we profiled our two-way simulation kernel (used in steps 3, 4, 5 of Alg. 1): our kernel requires 2.4 ms, while our implementation of [Bogner 2017] takes 11.9 ms, about 5 times slower than our approach. Even for scenes with high water occupancy such as the single bunny drop in Fig. 22 (bottom), our method is more efficient, with only a 6% improvement. We note that our approach is also more stable and less sensitive to the intersection truncation error due to fewer link-mesh intersection operations.

Handling fresh nodes. Our treatment of fresh nodes drastically improves simulations. Compared to [Bogner 2017] which sets fresh nodes as fluid cells, our solver does not exhibit any “sticking” artifacts on moving objects as demonstrated in Fig. 23. Compared to [Li and Desbrun 2023] which sets fresh nodes as gas cells this time, our solver does not generate spurious bubbles as shown in Fig. 22 where a bunny drops into water: in the last frame, there are fewer than 10 bubbles in our result (which quickly disappear afterwards), but setting fresh nodes as gas cells ends up with over 70 small bubbles right against the bunny, making the simulation overly bubbly.

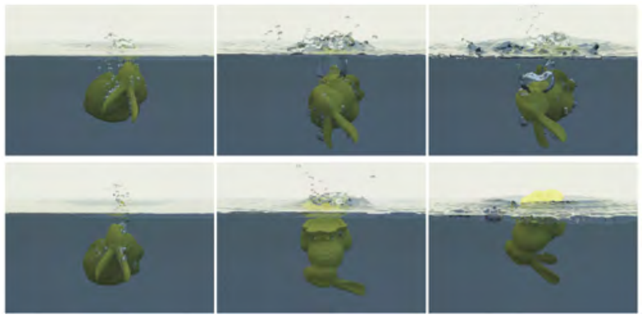


Fig. 22. **Ablation test for fresh cell treatment I.** If we replace our fresh cell handling (bottom) with [Li and Desbrun 2023] (top) in a splashing bunny simulation, spurious bubbles appear since fresh cells are set as gas cells.

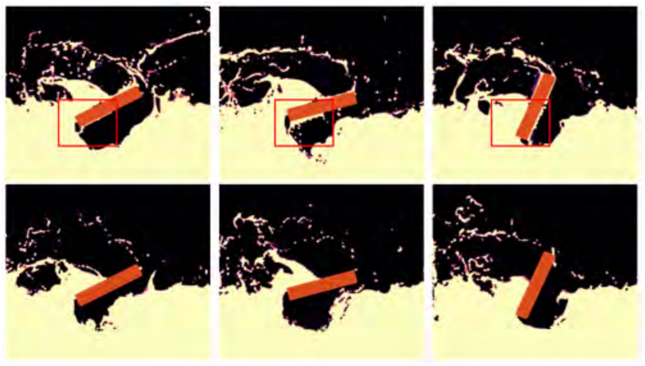


Fig. 23. **Ablation test for fresh cell treatment II.** If we replace our fresh cell handling (bottom) with [Bogner 2017] (top) in a rotating propeller simulation, we observe a large amount of water adhering to the solid object (red boxes) since fresh cells are initialized as fluid cells.

6 Conclusion

In this paper, we developed a kinetic free-surface fluid solver using a VOF-based sharp-interface encoding. By fixing a series of shortcomings of the original FSLBM, our resulting HOME-FREE LBM approach ends up rivaling with state-of-the-art kinetic diffuse-interface based fluid solvers in accuracy and stability, but allows for much finer fluid interface details for a given grid resolution. Our approach is thus at least two times more efficient than prior art, and can now exhibit not just more bubbles, but even the formation of persistent bubbles that stick to each other through surface tension or other complex foam behavior with very thin lamellae.

Limitations and future work. Given the complexity of bubble and foam simulation, our method is certainly not without limitations. First and foremost, wetting has not been treated in this work. We believe that incorporating wetting boundary conditions by changing the curvature estimation based on methods such as [Popinet 2009; Bogner et al. 2016] is possible, but requires further investigation. While we managed to handle small bubbles more robustly than the original FSLBM, our VOF-based approach cannot capture lots of tiny bubbles (being limited by the grid cell size); augmenting our approach with Lagrangian particles, potentially inspired by [Wretborn et al. 2025], could be a practical solution. Since our approach offers control over the size and persistence of bubbles, it would also be valuable to provide an intuitive interface to guide a user towards a desired behavior of the simulated flow (e.g., for the water pouring example in Fig. 1 vs. Fig. 3). Finally, we may explore different mesh extraction techniques to extract the sharp interface for rendering purposes, like Poisson reconstruction or dual contouring instead of relying on marching cubes followed by smoothing and remeshing.

Acknowledgments

We thank the reviewers for their help with improving our paper. The bunny mesh in Figs. 7,9, and 22 is from of the Stanford Computer Graphics Laboratory. This work was supported by the National Key R&D Program of China (2023YFB3209700), and the National Natural Science Foundation of China (62322110). MD acknowledges the generous support of an Inria chair, of the MediTwin consortium (funded by the French government as part of France 2030), and of Ansys, while Wei benefited from SJTU’s startup funds.

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), 1–12.

Iván Alduán, Angel Tena, and Miguel A. Otaduy. 2017. DYVERSO: A Versatile Multi-Phase Position-Based Fluids Solution for VFX. *Comput. Graph. Forum* 36, 8 (2017), 32–44.

Stefano Allegretti, Federico Bolelli, and Costantino Grana. 2019. Optimized block-based algorithms to label connected components on GPUs. *IEEE Trans. Parallel and Distributed Systems* 31, 2 (2019), 423–438.

Daniela Anderl, Simon Bogner, Cornelia Rauh, Ulrich Rüde, and Antonio Delgado. 2014. Free surface lattice Boltzmann with enhanced bubble model. *Computers & Mathematics with Applications* 67, 2 (2014), 331–339.

Ryoichi Ando and Christopher Batty. 2020. A practical octree liquid simulator with adaptive surface resolution. *ACM Trans. Graph.* 39, 4 (2020), 32–1.

Mohammadmehdi Ataei, Vahid Shaayegan, Franco Costa, Sejin Han, Chul B Park, and Markus Bussmann. 2021. LBfoam. *Comp. Phys. Comm.* 259 (2021), 107698.

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 (2016).

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.

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 Bhathnagor, Eugene P. Gross, and Max Krook. 1954. A model for collision processes in gases. *Physical Review* 94, 3 (1954), 511.

Blender Online Community. 2018. *Blender – a 3D modelling and rendering package.* Blender Foundation. http://www.blender.org

Simon Bogner. 2017. *Direct numerical simulation of liquid-gas-solid flows based on the lattice Boltzmann method.* Ph. D. Dissertation. Erlangen-Nürnberg University.

Simon Bogner, Regina Ammer, and Ulrich Rüde. 2015. Boundary conditions for free interfaces with the lattice Boltzmann method. *J. Comput. Phys.* 297 (2015), 1–12.

Simon Bogner, Ulrich Rüde, and Jens Harting. 2016. Curvature estimation from a volume-of-fluid indicator function for the simulation of surface tension and wetting with a free-surface lattice Boltzmann method. *Physical Review E* 93, 4 (2016), 043302.

Federico Bolelli. 2023. YACCLAB: Yet Another Convolutional Classification Algorithm Benchmarks. https://github.com/prittt/YACCLAB Accessed: 2025-01-11.

Federico Bolelli, Michele Cancilla, Lorenzo Baraldi, and Costantino Grana. 2020. Toward reliable experiments on the performance of connected components labeling algorithms. *Journal of Real-Time Image Processing* 17 (2020), 229–244.

Landon Boyd and Robert Bridson. 2012. MultiFLIP for Energetic Two-Phase Fluid Simulation. *ACM Trans. Graph.* 31, 2, Article 16 (2012).

Oren Breslouer. 2010. Rayleigh-plateau instability: falling jet. *Project Report* (2010).

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

John WM Bush and Alexander E Hasha. 2004. On the collision of laminar jets: fluid chains and fishbones. *Journal of fluid mechanics* 511 (2004), 285–310.

Wenjin Cao, Zhe Li, Xuhui Li, and David Le Touzé. 2020. A regularized single-phase lattice Boltzmann method for free-surface flows. *Computers & Mathematics with Applications* 80, 10 (2020), 2194–2211.

Caitlin M. Chalk, Manuel Pastor, Jeff Peakall, Duncan J. Borman, Andrew Sleigh, William Murphy, and Raul Fuentes. 2020. Stress-Particle Smoothed Particle Hydrodynamics: An application to the failure and post-failure behaviour of slopes. *Comput. Methods in Appl. Mech. Eng.* 366 (2020), 113034.

Xiaodong Chen, Dongjun Ma, Vigor Yang, and Stephane Popinet. 2013. High-fidelity simulations of impinging jet atomization. *Atomization and sprays* 23, 12 (2013).

Xiao-Song Chen, Chen-Feng Li, Geng-Chen Cao, Yun-Tao Jiang, and Shi-Min Hu. 2020a. A moving least square reproducing kernel particle method for unified multiphase continuum simulation. *ACM Trans. Graph.* 39, 6 (2020), 1–15.

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

Junghyun Cho and Hyeong-Seok Ko. 2013. Geometry-Aware Volume-of-Fluid Method. *Comput. Graph. Forum* 32, 2 (2013), 379–388.

Jens Cornels, Markus Ihmsen, Andreas Peer, and Matthias Teschner. 2014. IISPH-FLIP for incompressible fluids. *Comput. Graph. 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).

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, Article 50 (2015).

Alessandro De Rosis and Kai H Luo. 2019. Role of higher-order Hermite polynomials in the central-moments-based lattice Boltzmann framework. *Phys. Rev. E* 99, 1 (2019), 013301.

Yitong Deng, Mengdi Wang, Xiangxin Kong, Shiying Xiong, Zangyueyang Xian, and Bo Zhu. 2022. A moving eulerian-lagrangian particle method for thin film and foam simulation. *ACM Trans. Graph.* 41, 4 (2022).

BV Derjaguin and NV Churaev. 1978. On the question of determining the concept of disjoining pressure and its role in the equilibrium and flow of thin films. *Journal of Colloid and Interface Science* 66, 3 (1978), 389–398.

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 *EG Workshop on Computer Animation and Simulation.* 61–76.

Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. 2020. IQ-MPM: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Trans. Graph.* 39, 4, Article 51 (2020).

Yun Fei, Qi Guo, Rundong Wu, Li Huang, and Ming Gao. 2021. Revisiting integration in the material point method: a scheme for easier separation and less dissipation. *ACM Trans. Graph.* 40, 4 (2021), 1–16.

Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2018. A multi-scale model for simulating liquid-fabric interactions. *ACM Trans. Graph.* 37, 4, Article 51 (2018).

Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2019. A multi-scale model for coupling strands with shear-dependent liquid. *ACM Trans. Graph.* 38, 6, Article 190 (2019).

James D. Foley. 1996. *Computer graphics: principles and practice.* Addison-Wesley.

Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graph. Mod. Image Process.* 58, 5 (1996), 471–483.

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

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, Article 43 (2020).

Ryan Goldade, Christopher Batty, and Chris Wojtan. 2016. A practical method for high-resolution embedded liquid surfaces. In *Eurographics.* 233–242.

Yulong Guo, Xiaopei Liu, and Xuemiao Xu. 2017. A Unified Detail-Preserving Liquid Simulation by Two-Phase Lattice Boltzmann Modeling. *IEEE Trans. Vis. Comp. Graph.* 23, 5 (2017), 1479–1491.

John G Hagedorn, Nicos S Martys, and Jack F Douglas. 2004. Breakup of a fluid thread in a confined geometry: droplet-plug transition, perturbation sensitivity, and kinetic stabilization with confinement. *Phys. Rev. E* 69, 5 (2004), 056312.

William Henry. 1832. Experiments on the quantity of gases absorbed by water at different temperatures and under different pressures. In *Phil. Trans. R. Soc.* 103–104.

Cyril W Hirt and Billy D Nichols. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comput. Phys.* 39, 1 (1981), 201–225.

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

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.* 37, 4 (2018).

C Huber, Y Su, CT Nguyen, Andrea Parmigiani, Helge M Gonnermann, and J Dufek. 2014. A new bubble dynamics model to study bubble growth, deformation, and coalescence. *Journal of Geophysical Research: Solid Earth* 119, 1 (2014), 216–239.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2013. Implicit incompressible SPH. *IEEE Trans. Vis. Comp. Graph.* 20, 3 (2013), 426–435.

Christian Janßen and Manfred Krafczyk. 2011. Free surface flow simulations on GPGPUs using the LBM. *Computers & Mathematics with Applications* 61, 12 (2011), 3549–3563.

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.

Petr Karnakov, Sergey Litvinov, and Petros Koumoutsakos. 2022. Computing foaming flows across scales: From breaking waves to microfluidics. *Science Advances* 8, 5 (2022), eabm0590.

Byungmoon Kim. 2010. Multi-phase fluid simulations using regional level sets. *ACM Trans. Graph.* 29, 6 (2010), 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, Article 98 (2007).

Carolin Körner, Michael Thies, Torsten Hofmann, Nils Thürey, and Ulrich Rüde. 2005. Lattice Boltzmann model for free surface flow for modeling foaming. *Journal of Statistical Physics* 121 (2005), 179–196.

Dan Koschier and Jan Bender. 2017. Density maps for improved SPH boundary handling. In *Symposium on Computer Animation.*

Anthony J. C. Ladd. 1994. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. *J. Fluid Mech.* 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).

Moritz Lehmann. 2019. *High Performance Free Surface LBM on GPUs.* Ph. D. Dissertation. University of Bayreuth.

Marcel Lesieur. 1990. *Turbulence in Fluids: Stochastic and Numerical Modelling.* Kluwer Academic Publishers.

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, Article 47 (2020).

Wei Li and Mathieu Desbrun. 2023. Fluid-Solid Coupling in Kinetic Two-Phase Flow Simulation. *ACM Trans. Graph.* 42, 4, Article 123 (2023).

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.

Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient Kinetic Simulation of Two-Phase Flows. *ACM Trans. Graph.* 41, 4, Article 114 (2022).

Wei Li, Tongtong Wang, Zherong Pan, Xifeng Gao, Kui Wu, and Mathieu Desbrun. 2023b. High-Order Moment-Encoded Kinetic Simulation of Turbulent Flows. *ACM Trans. Graph.* 42, 6 (2023), 1–13.

Wei Li, Kui Wu, and Mathieu Desbrun. 2024. Kinetic Simulation of Turbulent Multifluid Flows. *ACM Trans. Graph.* 43, 4, Article 55 (2024).

Xingqiao Li, Xingyu Ni, Bo Zhu, Bin Wang, and Baoquan Chen. 2023a. GARM-LS: A Gradient-Augmented Reference-Map Method for Level-Set Fluid Simulation. *ACM Trans. Graph.* 42, 6 (2023).

Shusen Liu, Xiaowei He, Yuzhong Guo, Yue Chang, and Wencheng Wang. 2024. A Dual-Particle Approach for Incompressible SPH Fluids. *ACM Trans. Graph.* 43, 3 (2024).

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

Chaoyang Lyu, Kai Bai, Yiheng Wu, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2023. Building a Virtual Weakly-Compressible Wind Tunnel Testing Facility. *ACM Trans. Graph.* 42, 4 (2023).

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, Article 201 (2021).

Yihui Ma, Xiaoyu Xiao, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2024. Hybrid LBM-FVM solver for two-phase flow simulation. *J. Comput. Phys.* 506 (2024), 112920.

Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Trans. Graph.* 32, 4 (2013), 1–12.

V. Mihalef, B. Unlusu, D. Metaxas, M. Sussman, and M. Y. 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.

Marcel Padilla, Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2019. On bubble rings and ink chandeliers. *ACM Trans. Graph.* 38, 4 (2019).

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An implicit viscosity formulation for SPH fluids. *ACM Trans. Graph.* 34, 4, Article 114 (2015).

Cheng Peng, Yihua Teng, Brian Hwang, Zhaoli Guo, and Lian-Ping Wang. 2016. Implementation issues and benchmarking of lattice Boltzmann method for moving rigid particle simulations in a viscous flow. *Math. Comput. Simul.* 72, 2 (2016), 349–374.

Jonas Plewinski, Christoph Alt, Harald Köstler, and Ulrich Rüde. 2024. Performance analysis of the free surface lattice Boltzmann implementation in waLBerla. *PAMM* 24, 3 (2024), e202400196.

Thomas Pohl. 2008. *High performance simulation of free surface flows using the lattice Boltzmann method.* Ph. D. Dissertation. Erlangen-Nürnberg University.

Stéphane Popinet. 2009. An accurate adaptive solver for surface-tension-driven interfacial flows. *J. Comput. Phys.* 228, 16 (2009), 5838–5866.

Simon Premžoe, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker. 2003. Particle-Based Simulation of Fluids. *Comput. Graph. Forum* 22, 3 (2003), 401–410.

Evan Pruitt, William Markiewicz, Carlos Gonzalez, and Xiaofeng Liu. 2024. Visualization and feature tracking of the atomization of impinging jets. *Physical Review Fluids* 9, 11 (2024), 110513.

Ziyin Qu, Minchen Li, Yin Yang, Chenfanfu Jiang, and Fernando De Goes. 2023. Power Plastics: A Hybrid Lagrangian/Eulerian Solver for Mesoscale Inelastic Flows. *ACM Trans. Graph.* 42, 6 (2023).

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), 1–11.

Liangwang Ruan, Jinyuan Liu, Bo Zhu, Shinjiro Sueda, Bin Wang, and Baoquan Chen. 2021. Solid-fluid interaction with surface-tension-dominant contact. *ACM Trans. Graph.* 40, 4 (2021).

Arnaud Saint-Jalmes. 2006. Physical chemistry in foam drainage and coarsening. *Soft Matter* 2, 10 (2006), 836–849.

Sergio Sancho, Jingwei Tang, Christopher Batty, and Vinicius C. Azevedo. 2024. The Impulse Particle-In-Cell Method. *Comput. Graph. Forum* 43, 2 (2024), e15022.

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.

Hagit Schechter and Robert Bridson. 2012. Ghost SPH for animating water. *ACM Trans. Graph.* 31, 4 (2012), 61.

Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. 2006. Kinetic theory representation of hydrodynamics: a way beyond the Navier–Stokes equation. *J. Fluid Mech.* 550 (2006), 413–441.

Han Shao, Libo Huang, and Dominik L. Michels. 2022. A fast unsmoothed aggregation algebraic multigrid framework for the large-scale simulation of incompressible flow. *ACM Trans. Graph.* 41, 4, Article 49 (2022).

B. Solenthaler and R. Pajarola. 2008. Density contrast SPH interfaces. In *Symposium on Computer Animation.* 211–218.

B. Solenthaler and R. Pajarola. 2009. Predictive-corrective incompressible SPH. *ACM Trans. Graph.* 28, 3, Article 40 (2009).

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.

S Succi. 2001. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond.* Oxford University Press.

Peng-Nan Sun, Andrea Colagrossi, Salvatore Marrone, Matteo Antuono, and A-Man Zhang. 2018. Multi-resolution Delta-plus-SPH with tensile instability control: Towards high Reynolds number flows. *Comp. Phys. Comm.* 224 (2018), 63–80.

Yuchen Sun, Linglai Chen, Weiyuan Zeng, Tao Du, Shiying Xiong, and Bo Zhu. 2024. An Impulse Ghost Fluid Method for Simulating Two-Phase Flows. *ACM Trans. Graph.* 43, 6 (2024).

Tetsuya Takahashi and Christopher Batty. 2020. Monolith: a monolithic pressure-viscosity-contact solver for strong two-way rigid-rigid rigid-fluid coupling. *ACM Trans. Graph.* 39, 6 (2020), 1–16.

Tetsuya Takahashi and Christopher Batty. 2022. ElastoMonolith: A monolithic optimization-based liquid solver for contact-aware elastic-solid coupling. *ACM Trans. Graph.* 41, 6 (2022), 1–19.

Nils Thürey. 2003. *A single-phase free-surface Lattice-Boltzmann Method.* Master's thesis. Erlangen-Nürnberg University.

Nils Thürey. 2007. *Physically based Animation of Free Surface Flows with the Lattice Boltzmann Method.* Ph. D. Dissertation. Erlangen-Nürnberg University.

Nils Thürey, C Körner, and U Rüde. 2005. *Interactive free surface fluids with the lattice Boltzmann method.* Technical Report 05-4. University of Erlangen-Nuremberg.

Nils Thürey, Thomas Pohl, Ulrich Rüde, Markus Oechsner, and Carolin Körner. 2006. Optimization and stabilization of LBM free surface flow simulations using adaptive parameterization. *Computers & fluids* 35, 8-9 (2006), 934–939.

Nils Thürey and Ulrich Rüde. 2004. Free Surface Lattice-Boltzmann fluid simulations with and without level sets.. In *Int. Symp. Vision, Modeling, & Visualization.* 199–207.

Nils Thürey and Ulrich Rüde. 2005. Optimized free surface fluids on adaptive grids with the lattice Boltzmann method. In *ACM SIGGRAPH 2005 Posters.* 112–es.

Nils Thürey and Ulrich Rüde. 2009. Stable free surface flows with the lattice Boltzmann method on adaptively coarsened grids. *Comp. Vis. Sci.* 12 (2009), 247–263.

N. Thürey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Gross. 2007. Real-time simulations of bubbles and foam within a shallow water framework. In *Symposium on Computer Animation.* 191–198.

Zaili Tu, Chen Li, Zipeng Zhao, Long Liu, Chenhui Wang, Changbo Wang, and Hong Qin. 2024. A Unified MPM Framework Supporting Phase-field Models and Elastic-viscoplastic Phase Transition. *ACM Trans. Graph.* 43, 2 (2024).

Jan van Bergen. 2023. GPU Raytracer. https://github.com/jan-van-bergen/GPU-Raytracer Accessed: 2025-01-11.

Hui Wang, Zhi Wang, Shulin Hong, Xubo Yang, and Bo Zhu. 2024. A Moving Least-Squares/Level-Set Particle Method for Bubble and Foam Simulation. *IEEE Trans. Vis. Comp. Graph.* (2024), 1–15.

Joel Wretborn, Sean Flynn, and Alexey Stomakhin. 2022. Guided bubbles and wet foam for realistic whitewater simulation. *ACM Trans. Graph.* 41, 4 (2022).

Joel Wretborn, Alexey Stomakhin, and Christopher Batty. 2025. A unified multi-scale method for simulating immersed bubbles. *Comput. Graph. Forum* (2025), e70033.

Jingrui Xing, Liangwang Ruan, Bin Wang, Bo Zhu, and Baoquan Chen. 2022. Position-based surface tension flow. *ACM Trans. Graph.* 41, 6 (2022).

Shiying Xiong, Zhecheng Wang, Mengdi Wang, and Bo Zhu. 2022. A Clebsch method for free-surface vortical flow simulation. *ACM Trans. Graph.* 41, 4 (2022).

Xiao Yan, Yun-Tao Jiang, Chen-Feng Li, Ralph R Martin, and Shi-Min Hu. 2016. Multiphase SPH simulation for interactive fluids and solids. *ACM Trans. Graph.* 35, 4 (2016), 1–11.

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), 1–13.

David L Youngs. 1984. An interface tracking method for a 3D Eulerian hydrodynamics code. *Atomic Weapons Research Establishment* 44-92 (1984).

Shuai Zhang, Xubo Yang, Ziqi Wu, and Haibo Liu. 2015. Position-based fluid control. In *Symposium on Interactive 3D Graphics and Games.* 61–68.

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.