Fluid-Solid Coupling in Kinetic Two-Phase Flow Simulation

Wei Li, Mathieu Desbrun

Fluid-Solid Coupling in Kinetic Two-Phase Flow Simulation

WEI LI, Inria, France and Tencent Lightspeed Studios, China
MATHIEU DESBRUN, Inria / Ecole Polytechnique, France

Fig. 1. **Key Drop.** In this paper, we propose a stable and efficient kinetic two-phase flow simulator which can handle complex fluid-solid coupling, like a skeleton key being dropped in water. The dynamics of the bubbles entrapped by the fall is well captured as the insets of a real experiment demonstrate.

Real-life flows exhibit complex and visually appealing behaviors such as bubbling, splashing, glugging and wetting that simulation techniques in graphics have attempted to capture for years. While early approaches were not capable of reproducing multiphase flow phenomena due to their excessive numerical viscosity and low accuracy, kinetic solvers based on the lattice Boltzmann method have recently demonstrated the ability to simulate water-air interaction at high Reynolds numbers in a massively-parallel fashion. However, robust and accurate handling of fluid-solid coupling has remained elusive: be it for CG or CFD solvers, as soon as the motion of immersed objects is too fast or too sudden, pressures near boundaries and interfacial forces exhibit spurious oscillations leading to blowups. Built upon a phase-field and velocity-distribution based lattice-Boltzmann solver for multiphase flows, this paper spells out a series of numerical improvements in momentum exchange, interfacial forces, and two-way coupling to drastically reduce these typical artifacts, thus significantly expanding the types of fluid-solid coupling that we can efficiently simulate. We highlight the numerical benefits of our solver through various challenging simulation results, including comparisons to previous work and real footage.

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

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

ACM Reference Format:
Wei Li and Mathieu Desbrun. 2023. Fluid-Solid Coupling in Kinetic Two-Phase Flow Simulation. *ACM Trans. Graph.* 42, 4 (August 2023), 14 pages. https://doi.org/10.1145/3592138

1 INTRODUCTION

Fluid flows are all around us in our daily lives. Their complex dynamics and visual appeal are a source of awe for many, and an enduring challenge for computer graphics (CG) and computational fluid dynamics (CFD) researchers. While early fluid simulation offered only limited realism, the ever-more sophisticated techniques used in graphics for the animation of immiscible multiphase flows have successfully demonstrated the ability to produce typical complex behaviors such as bubbling or splashing realistically. To achieve this feat, various physically based simulation methods based on grids and/or particles have been developed to capture both the individual phases *and* their detailed interactions. More recently, the use of kinetic solvers based on lattice Boltzmann methods (LBM) have proved capable to handle large air-to-liquid density ratios and generate very turbulent flows, which had been out of reach for Navier-Stokes solvers in graphics. Yet, even if state-of-the-art fluid simulation of complex water-air interactions has reached cinematic realism at very reasonable computational cost, *fluid-solid coupling* is still limited to low-speed motions, preventing the simulation of common natural occurrences like the thrust of a propeller underwater or a stone skipping over a lake.

Reasons for the lack of robust and efficient coupling in multiphase flows are manifold. First, most graphics fluid simulators achieve efficiency through large time steps; although the resulting lack of accuracy is acceptable for visual purposes, large time steps are simply unable to capture the subtle interactions between two phases or the coupling between phases and obstacles, particularly for fast motions and turbulent flows. Second, enforcing a strong coupling (via a monolithic solver) is inefficient for turbulent multiphase flows, so partitioned coupling (for which a fluid solver and a deformable body solver are run separately, yet interfaced so as to exchange forces) is often favored in CG; but it often results in oscillating pressure near boundaries and inaccurate interfacial forces which eventually lead to blow-ups. Even the use of the immersed boundary method in CFD is known to fail for fairly turbulent flows and when dealing with thin shell models due to its inherent leakage through boundaries, making it poorly adapted to CG.

In this paper, we extend the phase-field and velocity-distribution based LBM approach of [Li et al. 2022] to offer two-way coupling and significantly improved robustness. We show that more accurate interface normal evaluations and phase field gradients dramatically

Authors' addresses: W. Li, eduardli@tencent.com, Tencent Lightspeed Studios, 139 Longai Road, Xuhui District, Shanghai; M. Desbrun, mathieu.desbrun@inria.fr, Inria Saclay, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France.

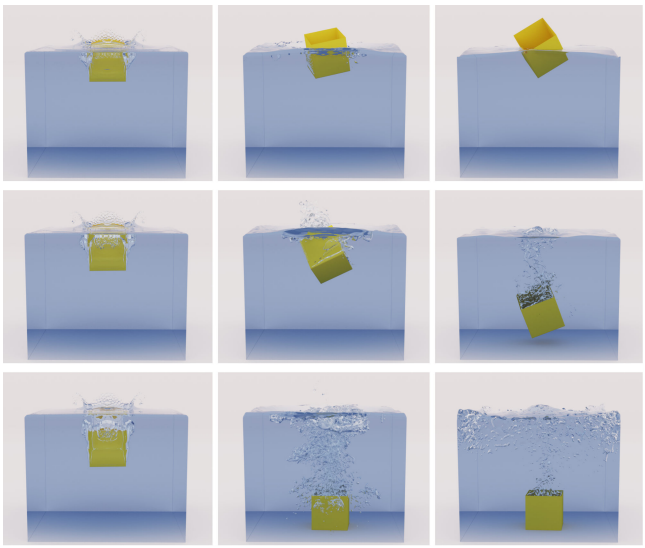

Fig. 2. **Cups.** Depending on its mass, a thin-shell cup dropped in a water tank will either float (top), or first float before water gushes in and sinks the cup (middle), or directly drop to the bottom (bottom).

enhance stability of fluid-solid coupling. By using a twice-finer phase-field discretization than the grid size of the fluid distribution function, we also significantly improve boundary handling and offer a simple strategy for dealing with dead and fresh cells. Our contributions bring robustness to multiphase flow simulation without impairing accuracy or efficiency, allowing us to simulate a far improved range of motions as shown for example in Figs. 2, 8, and 17. We also demonstrate that our solver reproduces well-known coupling behaviors, such as the fluttering or tumbling of coins in water (Fig. 16), or stone skipping over an air-water interface (Fig. 19) as it can handle the real-life density ratio between air and water as well as high Reynolds numbers flows very efficiently due to its massively-parallel implementation.

2 PREVIOUS WORK

Dealing with fluid-structure interactions has drawn significant attention in both CG and CFD due to its key role in capturing the fine interplay between obstacles and flows. We briefly review previous work on coupling of single- and two-phase fluid and solids.

2.1 Single-phase and free-surface solvers

Single-phase fluid-solid coupling. Early works handled fluid-solid coupling explicitly, semi-implicitly or fully implicitly [Yngve et al. 2000; Foster and Fedkiw 2001; Carlson et al. 2004; Takahashi et al. 2002; Génevaux et al. 2003; Klingner et al. 2006; Chentanez et al. 2006; Batty et al. 2007] by taking the Lagrangian solid velocity as a Neumann boundary condition for the fluid and integrating pressure forces at the solid surface. A fully Eulerian fluid-solid treatment was later introduced in [Teng et al. 2016]. To deal with thin-shell solids, [Guendelman et al. 2005] proposed ray-casting to prevent leakage, while [Robinson-Mosher et al. 2008, 2009] improved on this approach using ghost cells and more accurate momentum transfer. Eulerian solvers with boundary-conforming meshes [Feldman et al. 2005a,b; Dai and Schmidt 2005; Klingner et al. 2006; Elcott et al. 2007] have also been suggested, followed by Lagrangian methods [Misztal et al. 2010; Clausen et al. 2013]. More recently, the use of cut-cells using sub-cell information [Roble et al. 2005; Batty et al. 2007; Ng et al. 2009; Gibou and Min 2012; Weber et al. 2015; Edwards and Bridson 2014; Liu et al. 2015; Azevedo et al. 2016; Tao et al. 2022] has found wide adoption due to its reduced computational complexity and absence of fluid leakage for thin objects.

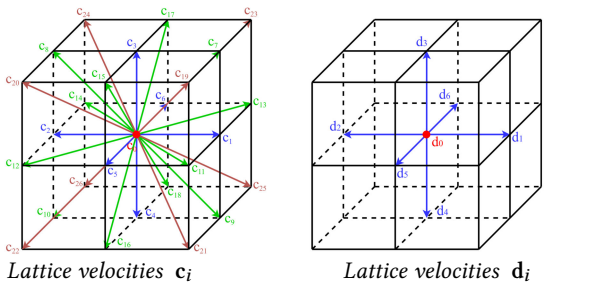

Fig. 3. **Lattice structures.** Flow velocities are stored and updated in time on a D3Q27 lattice (left) for high-accuracy with lattice weights w_i^c, while a simpler (and more memory-efficient) D3Q7 lattice structure (right) with lattice weights w_i^d suffices to encode the phase field.

Free-surface fluid-solid coupling. A Lagrangian discretization of fluids through smoothed particle hydrodynamics (SPH) was first proposed by [Desbrun and Gascuel 1996; Müller et al. 2003], but its inherent visual blobbiness near the interface started a number of palliative measures [Desbrun and Cani-Gascuel 1998; Zhu and Bridson 2005]. After the incompressibility of SPH was improved in [Solenthaler and Pajarola 2009; Bender and Koschier 2016], free-surface and solid boundary conditions were handled for particles more generally in [Schechter and Bridson 2012; Koschier and Bender 2017; Band et al. 2018], sometimes through penalty forces [Peer et al. 2015; Ihmsen et al. 2013]. To generate uniform particle distributions and precise volume control, power particles [de Goes et al. 2015] and hybrid particle-grid transfers [Qu et al. 2022] were also introduced. Hybrid particle-grid methods have been favored in recent years for coupling [Fei et al. 2018; Hu et al. 2018; Fei et al. 2019; Fang et al. 2020; Fei et al. 2021], but they are limited to fairly viscous flows. A mix of FLIP and boundary element method (BEM) [Huang et al. 2021] was even formulated to simulate larger-scale coupling, while Shao et al. [2022] presented an unsmoothed aggregation algebraic multigrid (UAAMG) method to support large-scale simulation, but bubbles and two-way coupling were not handled. The use of levelset-based free-surface liquid and solid one-way coupling has also demonstrated success when used with adaptive octrees for high-resolution surface discretizations [Aanjaneya et al. 2017; Ando and Batty 2020] or cut-cells to handle static thin objects [Chen et al. 2020b]. The volume-of-fluid (VOF) representation was also leveraged to design monolithic solvers [Takahashi and Batty 2020, 2022] for fluid-solid coupling, but such solvers have not been extended yet to multiphase flows and are currently rather inefficient. The lack of versatile and fast coupling techniques have also spawned a number of specialized solvers: [Ruan et al. 2021] proposed a three-way coupling method via a hybrid Eulerian-Lagrangian framework to simulate large surface tension flow in which liquid and solid are

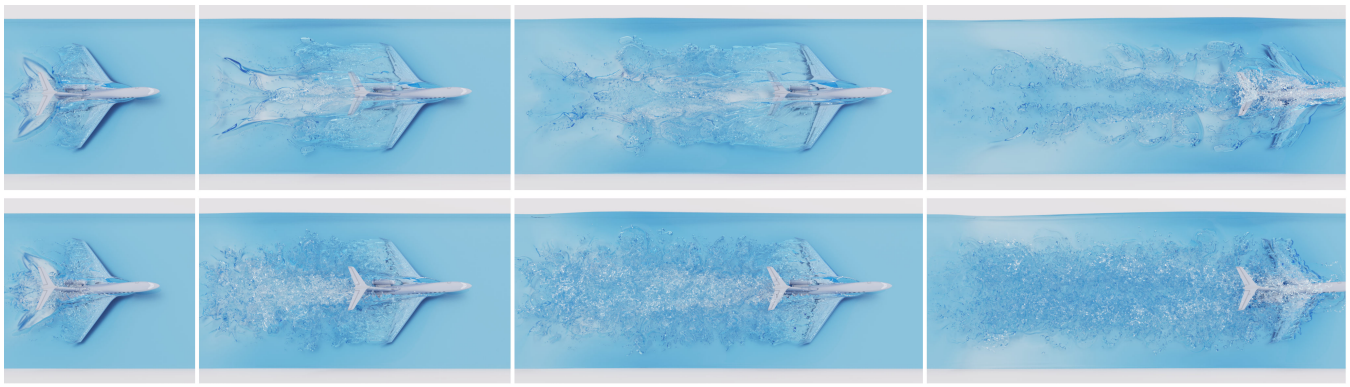

Fig. 4. **Airplane ditching.** An airplane comes to a stop in a pool of water. Depending on the Reynolds number (top: $Re = 4,000$; bottom: $Re = 200,000$) chosen for the simulation, one can witness very different coupling behaviors, with bubbles and smooth waves (top) or much greater turbulence (bottom).

coupled by a Lagrangian membrane; [Xiong et al. 2022] proposed a Clebsch method to simulate free-surface vortical flow; [Liu et al. 2022] introduced a Lagrangian model for viscous solid-fluid interaction; and position-based dynamics (PBD) for surface tension [Xing et al. 2022] was even proposed, but it does not support turbulent multiphase flows.

2.2 Navier-Stokes-based multiphase fluids

While ad hoc techniques proposed simply to add bubbles in the flow [Thuerey et al. 2007; Kim et al. 2010; Goldade et al. 2020], SPH-based methods [Solenthaler and Pajarola 2008; Ren et al. 2014; Yan et al. 2016; Yang et al. 2017] were shown capable of simulating multiple immiscible fluids of different densities, but only for viscous flows. Multiphase flows with a levelset [Kim 2010] or VOF [Cho and Ko 2013; Langlois et al. 2016] encoding of the interface have improved their treatment of discontinuous jumps in fluid density and pressure at the interface [Boyd and Bridson 2012; Kim et al. 2007; Losasso et al. 2006; Mihalef et al. 2006] through a variable density pressure projection [Kang et al. 2000] or the ghost fluid method [Hong and Kim 2005]; but again, most approaches along these lines focus on viscous fluids. There are also a few approaches relying on a diffuse-interface model [Song et al. 2005; Zheng et al. 2009] to avoid having to deal with a sharp interface. Hybrid grid-particle methods [Saye 2016, 2017; Yang et al. 2021; Su et al. 2021] have also been used for multiphase flows. A Moving Least Square Reproducing Kernel Method (MLSRK) framework to simulate multiphase fluids and solids in a unified manner was also formulated [Chen et al. 2020a]; however, the need for small time steps renders the solver impractical for large scenes. A mesh-based Lagrangian approach to simulate multiphase flows was also proposed in [Misztal et al. 2013], but only examples of very low Reynolds numbers were demonstrated.

2.3 Kinetic fluid solvers

Recently, kinetic solvers based on statistical mechanics have been introduced in graphics [Guo et al. 2017; Li et al. 2019], building upon the significant improvements over the original lattice Boltzmann method (LBM) used in computational physics over the last ten years. LBM-based kinetic solvers have emerged as a great alternative to the traditional incompressible Navier-Stokes fluid integrators from graphics both for single- [Li et al. 2020; Lyu et al. 2021] and multiphase flows [Li et al. 2021, 2022] due, by and large, to their massively-parallel nature: their high computational efficiency when implemented on GPU [Chen et al. 2021] is extremely attractive, offering the computation of realistic flows in a matter of minutes on a single GPU; moreover, progress in LBM collision modeling has resulted in simulation accuracy superior to Navier-Stokes solvers. While coupling in single-phase incompressible flows has already reached a maturity which is arguably sufficient for many graphics applications [Li et al. 2020; Lyu et al. 2021], this is far from true in the case of *multiphase flows*. In fact, developing a numerical scheme able to simulate the type of high Reynolds numbers and large density ratios that water-air interactions require has always been a challenge — even in CFD where the use of the immersed boundary method [Xiao et al. 2022; De Rosis and Tafuni 2022] can only handle viscous multiphase flows. The first fluid solver to reach the desired generality for graphics was presented recently in [Li et al. 2022]; it also offered much improved stability in handling *static boundaries* by proposing the use of a velocity-based distribution function instead of a momentum-based one, to avoid jumps between the two fluid phases of different densities. However, the authors lamented that their contributions to "ensure robustness for static obstacles do not suffice in the case of fast-evolving solid objects". This paper precisely addresses this limitation, in order to reach a far improved range of coupling in multiphase flow simulation.

3 OUR MULTIPHASE FLUID SOLVER

In this section, we recap the work of [Li et al. 2022] on which we base our solver, and detail the key changes we propose in order to significantly improve the numerical stability of fluid-solid coupling.

3.1 Fluid and phase-field equations

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

$$\begin{cases} \partial\rho/\partial t + \nabla\cdot(\rho\mathbf{u}) = 0, & \text{(1a)} \\ \partial(\rho\mathbf{u})/\partial t + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) = -\nabla p + \nabla\cdot\Pi + \mathbf{F}_s + \mathbf{F}_b, & \text{(1b)} \\ \nabla\cdot\mathbf{u} = 0, & \text{(1c)} \end{cases}$$

where ρ and $\mathbf{u}$ are respectively the spatially-varying fluid density and velocity of the two fluids, p is the hydrodynamic pressure enforcing incompressibility (Eq. (1c)), Π is the viscous stress tensor,

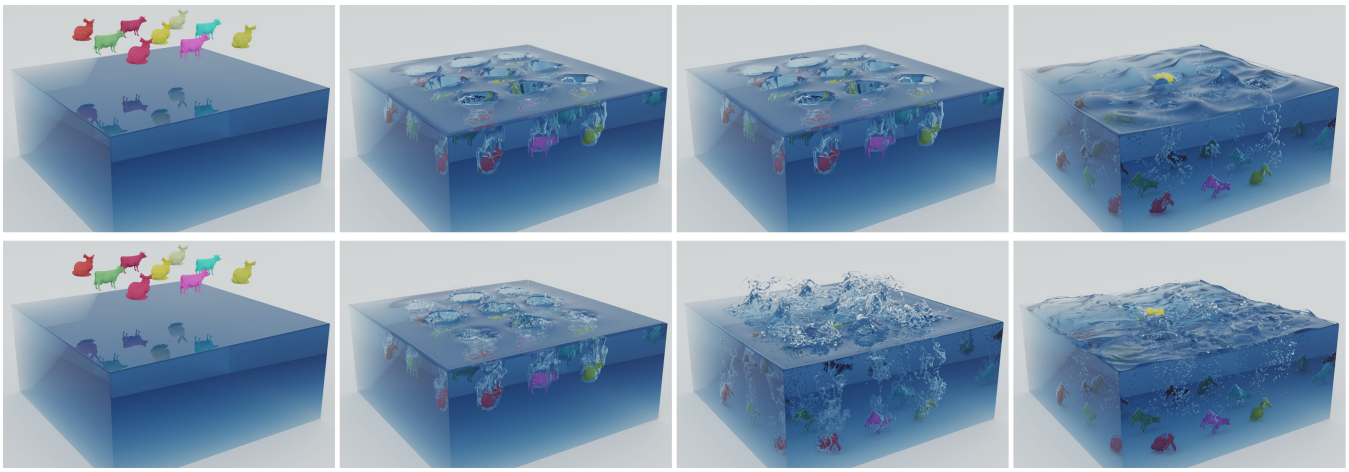

Fig. 5. **Raining Bunnies and Cows.** A bunch of cows and bunnies of various densities are dropped in a water tank, exhibiting complex bubbles and swirl patterns (top). Changing the global scale of the scene increases the Reynolds number from 13,000 to 130,000 (bottom), thus showing more turbulence.

while $\mathbf{F}_b$ and $\mathbf{F}_s$ are body and surface tension forces. To replace the typically sharp encoding of the interface and associated boundary conditions at this interface, a diffuse-interface model, called *phase field*, tracks the interface between the two phases through a conservative, profile-preserving Allen-Cahn equation to replace Eq. (1a) as explained in [Li et al. 2022]:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{u}) = \nabla \cdot \left[M\big(\nabla \phi - \frac{4}{\xi}\phi(1-\phi)\mathbf{n}\big)\right]. \tag{2}$$

Due to the right-hand side term of this equation, the phase field maintains a predefined steep profile where the interface between the two fluids lies, from which the interfacial forces can be well approximated. For a mixture of two immiscible, incompressible fluids with respective density ρ_H and ρ_L and respective viscosity ν_H and ν_L (where the subscripts H and L are used to differentiate the high-density fluid from the low-density fluid), the density field ρ is expressed from the phase field via:

$$\rho(\mathbf{x}, t) = \rho_L + (\rho_H - \rho_L)\,\phi(\mathbf{x}, t), \tag{3}$$

while the viscosity ν is defined from phase viscosities ν_H and ν_L as:

$$\frac{1}{\nu(\mathbf{x})} = \frac{1}{\nu_L} + \left(\frac{1}{\nu_H} - \frac{1}{\nu_L}\right)\phi(\mathbf{x}, t). \tag{4}$$

Lattice Boltzmann formulation. As described in [Li et al. 2022], the momentum and phase field equations for multiphase flows can be formulated through the time evolution of two distribution functions. The first one, a velocity-based distribution function $\mathbf{g}$, encodes the flow in time through:

$$g_i(\mathbf{x} + \mathbf{c}_i, t+1) = g_i(\mathbf{x}, t) + \Omega_i^g(\mathbf{x}, t) + G_i(\mathbf{x}, t), \tag{5}$$

where g_i represents $\mathbf{g}$ in the direction $\mathbf{c}_i$ from the $D3Q27$ lattice structure shown in Fig. 3(left), while Ω_i^g and G_i account for the collision and forcing terms respectively in this direction. Solving Eq. (5) is done in three steps: streaming, collision and boundary handling. The collision term requires the evaluation of the equilibrium distribution $\mathbf{g}^{\text{eq}}$, which, for this velocity-based formulation, is:

$$g_i^{\text{eq}} = \Gamma_i + (p^* - 1)w_i^c - \frac{G_i}{2}, \tag{6}$$

where w_i^c are the usual LBM lattice weights (see Fig. 3). The macroscopic hydrodynamic variables such as the normalized pressure p^* or the velocity $\mathbf{u}$ can be obtained respectively from the zeroth and first moments of the distribution function $\mathbf{g}$. The second distribution function, $\mathbf{h}$, discretizes the phase field equation as:

$$h_i(\mathbf{x} + \mathbf{d}_i, t+1) - h_i(\mathbf{x}, t) = \Omega_i^h(\mathbf{x}, t) + H_i(\mathbf{x}, t), \tag{7}$$

where h_i is the distribution function in the direction $\mathbf{d}_i$ from the $D3Q7$ lattice structure (Fig. 3, right). Solving Eq. (7) is done through the same three-prong (streaming, collision and boundary handling) process, and the macroscopic phase field can be recovered at any time step t as the zeroth-order moment of $\mathbf{h}$, i.e.,

$$\phi(\mathbf{x}, t) = \sum_{i=0}^{i=6} h_i(\mathbf{x}, t). \tag{8}$$

While we adopt most of this framework laid out by [Li et al. 2022] for multiphase flows, we present a series of targeted improvements on boundary handling and a new, finer-resolution scheme for the phase field equation to significantly improve the treatment and robustness of fluid-solid coupling as we detail next.

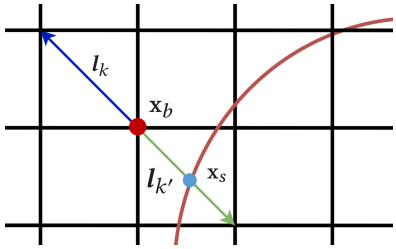

Fig. 6. **Hybrid moving bounce-back.** In the streaming step, we accommodate fluid-structure coupling through bounce-back: the distribution function at a node $\mathbf{x}_b$ whose link $l_{k'}$ hits a solid boundary at an intersection point $\mathbf{x}_s$ is made to bounce-back towards the opposite direction k, along with a momentum exchange due to the solid's speed.

3.2 Fluid-solid coupling

The pros and cons of the various approaches to handle boundaries are well documented by now [Lyu et al. 2021], with the simplicity of the moving bounce-back (MBB) scheme often winning over the

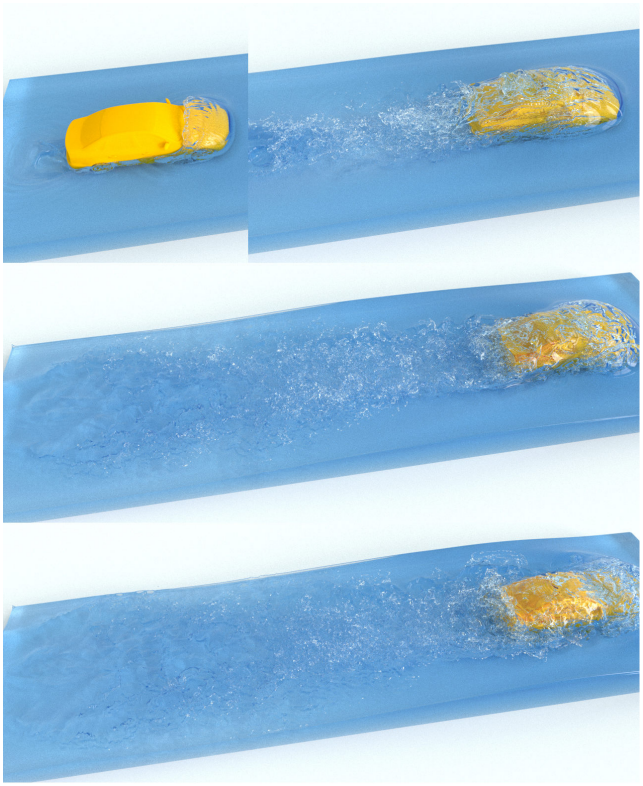

Fig. 7. **Car in a flood.** We simulate a car driving around 40mph through high water before coming to a stop; the usual "air bubble" forms in front of the car, and complex water patterns appear in the wake flow region.

inability of the immersed boundary method (IBM) to handle thin objects in graphics applications. We also adopt an MBB-like approach in our work to allow for arbitrary objects; but before introducing what we call a hybrid moving bounce-back (HMBB) scheme, let's review the original moving bounce-back scheme for single-phase flows first. Since streaming simply updates the distribution function $\mathbf{g}$ as if the flow was locally unobstructed, one needs to alter this basic streaming for links intersecting a boundary. For a node at $\mathbf{x}_b$ with a link $l_{k'}$ crossing a static obstacle as shown in Fig. 6, we can simply redirect the streaming along the opposite link l_k, which reproduces a bounce of the fluid particles off the obstacle. More generally, if the obstacle is moving, the moving bounce-back scheme proposed by [Ladd 1994] is written for a *momentum-based* distribution as

$$f_k(\boldsymbol{x}_b, t+1) = f_{k'}(\boldsymbol{x}_b, t) + 6\, w_{k'}\, \rho \boldsymbol{u}_s \cdot \boldsymbol{c}_k \,, \tag{9}$$

where $\boldsymbol{u}_s$ is the obstacle velocity at the link intersection $\boldsymbol{x}_s$: it corresponds to a bounce of the distribution on the obstacle to which is added a momentum transfer due to the boundary motion. In our context of *velocity-based* LBM multiphase flow simulation, this approach is simply rewritten as:

$$g_k(\boldsymbol{x}_b, t+1) = g_{k'}(\boldsymbol{x}_b, t) + 6\, w^c_{k'}\, \boldsymbol{u}_s \cdot \boldsymbol{c}_k \,. \tag{10}$$

This moving bounce-back approach no longer needs to consider the density discontinuity near the interface due to the use of velocity-based distributions — but it inherits the typical ring artifacts mentioned in [Lyu et al. 2021; Li et al. 2022] for high Reynolds numbers. To ensure numerical stability of our boundary treatment, we thus adopt the hybrid bounce-back scheme designed by [Li et al. 2022] for *static obstacles*, to which we *add* the momentum exchange when the obstacle is moving. Moreover, their approach relied on an equilibrium distribution based on the normalized pressure of a fluid node next to the obstacle, as this was deemed to be the best local approximation of pressure locally available. However, bounce-back often creates spurious pressure oscillations near solids, which renders the equilibrium distribution inaccurate. We thus employ a smoothed pressure formulated in [Sitompul and Aoki 2019] with a second order filter instead, to further improve the stability of our approach. In fine, our *hybrid moving bounce-back* leverages the equilibrium distribution function $\hat{\mathbf{f}}(\boldsymbol{u}, \bar{p}^*)$ coming from an obstacle boundary towards a fluid node along a boundary-crossing link based on the known solid velocity $\boldsymbol{u}_s$ and the most reliable normalized pressure available $\bar{p}^*(\mathbf{x}_b)$, resulting in a boundary update:

$$g_k(\mathbf{x}_b, t+1) = (1-\alpha_b)\, g_{k'}(\mathbf{x}_b, t) + \alpha_b\, \hat{f}^{\,\mathrm{eq}}_{k'}\big(\boldsymbol{u}_s, \bar{p}^*(\mathbf{x}_b)\big) + 6\, w^c_{k'}\, \boldsymbol{u}_s \cdot \boldsymbol{c}_k \,, \tag{11}$$

where the filtered normalized pressure to enhance the stability is expressed as [Sitompul and Aoki 2019]

$$\bar{p}^*(\mathbf{x}_b) = \sum_{i=0}^{26} w^c_i\, p^*(\mathbf{x}_b + \boldsymbol{c}_i, t). \tag{12}$$

Note that we set the blending coefficient in Eq. (11) to $\alpha_b = 0.1$ in order to safely suppress pressure fluctuations near moving obstacles. This differs from [Li et al. 2022] which used a value of α_b based on pressure instead; our experience shows that such a strategy can in fact back-fire during impact due to pressure inaccuracy, so we opt for a safer pressure-independent value.

3.3 Resulting Forces on Solids

While our hybrid treatment of bounce-back accounts for the momentum exchange from solid to fluid, we also need to evaluate the exchanged momentum $\Delta\mathbf{p}$ from the fluid to the obstacle based on the distribution $\mathbf{g}$. While many expressions have been proposed to evaluate this term with various degrees of accuracy, the kinetic-inspired approach of [Peng et al. 2016] which simply tracks momentum changes of fluid lattice particles with the fluid–solid boundary during bounce-back is particularly simple to evaluate: when a link at $\mathbf{x}_b$ is intersected by a boundary, the momentum *transferred to the solid* in the direction $\boldsymbol{c}_{k'}$ is $\rho(\mathbf{x}_b, t)\big[g_{k'}(\mathbf{x}_b, t) + g_k(\mathbf{x}_b, t+1)\big]\, \boldsymbol{c}_{k'}$. However, the simplicity of this evaluation is also its weakness: because it uses the local density ρ near boundaries, it is prone to inaccuracies: since LBM deals with weakly-compressible fluids, pressure waves near boundaries often lead to transient, yet exaggerated density changes. Our numerical tests indicate up to 20% density fluctuation on large impact of obstacles in water — e.g., in the cup drop example of Fig. 2. Therefore, we modify the expressions above by replacing the actual density ρ with a *filtered* density $\bar{\rho}$ computed as

$$\bar{\rho}(\mathbf{x}_b, t) = \sum_{i=0}^{26} w^c_i\, \big(\beta_i\, \rho(\mathbf{x}_b + \boldsymbol{c}_i, t) + (1-\beta_i)\rho(\mathbf{x}_b, t)\big) \tag{13}$$

$$\text{with } \begin{cases} \beta_i = 1 & \text{if } \|\phi(\mathbf{x}_b + \boldsymbol{c}_i) - \phi(\mathbf{x}_b)\| \le 0.25 \\ \beta_i = 0 & \text{otherwise;} \end{cases}$$

i.e., we primarily take the average of the densities around $\mathbf{x}$ exactly as done for the pressure in Eq. (12), unless the density jump along a link is too significant (>25%, which could not be due to spurious

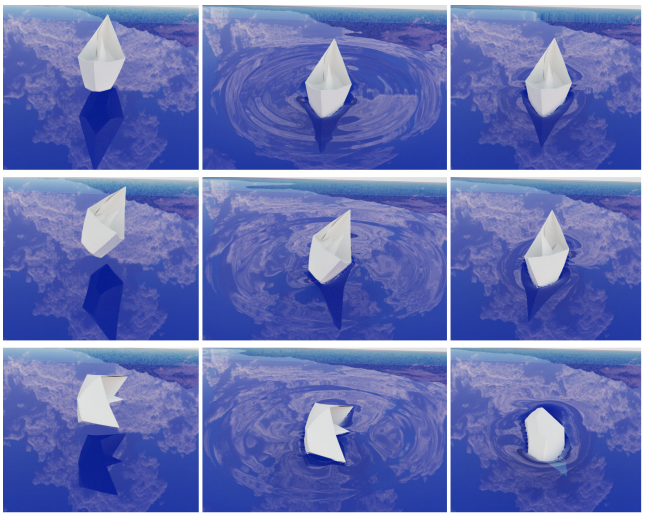

Fig. 8. **Floating paper boat.** In this example, surface tension keeps a very light paper boat afloat. Depending on how the paper boat is initially dropped, it lands squarely (top), or shifts back into a proper floating position (middle), or simply flips around (bottom). Capillary waves are visible as well.

oscillations alone) in which case we use the safer value at $\mathbf{x}_b$. Therefore, to obtain the force acting on the solid surface, one can sum up all these momentum changes to find the net loss of fluid momentum:

$$\boldsymbol{F}_B \equiv \sum_{\mathbf{x}_b} \bar{\rho}(\mathbf{x}_b, t) \sum_{\ell_k \in L_{\mathbf{x}_b}} \boldsymbol{c}_{k'} \left[g_{k'}(\mathbf{x}_b, t) + g_k(\mathbf{x}_b, t+1) \right], \tag{14}$$

where $L_{\mathbf{x}}$ refers to the set of links (if any) starting at node $\mathbf{x}$ which intersect a boundary. Similarly, if $\mathbf{x}_C$ denotes the barycenter of the obstacle, the total torque is written as:

$$\boldsymbol{\tau}_B \equiv \sum_{\mathbf{x}_b} \bar{\rho}(\mathbf{x}_b, t)(\mathbf{x}_b - \boldsymbol{x}_C) \times \sum_{\ell_k \in L_{\mathbf{x}_b}} \boldsymbol{c}_{k'} \left[g_{k'}(\mathbf{x}_b, t) + g_k(\mathbf{x}_b, t+1) \right]. \tag{15}$$

Note that the force and torque expressions are in LBM space, so we need to map them to rigid-body physical space before sending them to the solid integrator, as detailed in [Li et al. 2020; Lyu et al. 2021].

3.4 Finer resolution for phase field equation

Existing works in LBM-based multiphase flows [Li et al. 2021, 2022] use the same LBM grid resolution for both flow and phase fields, albeit with a higher lattice count for the velocity. Since coupling between the incompressible flow and phase field equations happens through the macroscopic velocity $\mathbf{u}$ and the macroscopic phase field ϕ and its gradient, our tests show that it is often a lack of accuracy in the interface normal that creates blowups. In this section, we thus detail how we employ *two different grid resolutions* (the phase field one being finer to improve interfacial accuracy) that we position in space in a staggered fashion for easy interfacing.

Grid ratio. Instead of implementing an adaptive grid refinement scheme to improve the resolution of the phase field, we simply decouple the flow and phase field resolutions: the distribution $\mathbf{g}$ is computed on a grid of size N^3, while the distribution $\mathbf{h}$ is computed on a $(sN)^3$ grid with a ratio $s \in \mathbb{N}$. While any scaling s could be accommodated, we found that $s=2$ is sufficient for graphics purposes and better in terms of cache efficiency: the 8-times finer spatial discretization for the phase field is enough to put velocity and phase field on equal footing in terms of accuracy. We will thus stick to this grid ratio in the remainder of this section.

Spatial evaluation on staggered grids. Since we now have two different grids on which to store and evolve the flow and phase fields, we need to determine how they must be positioned with respect to each other, and how to interface them so that unknown values of a field on a given grid can be quickly interpolated from the other grid. We opted for a *staggered* positioning to simplify interfacing: as Fig. 9 illustrates for the 2D case, every coarse node is centered on a fine grid cell. This makes interfacing the two grids quite straightforward and symmetric:

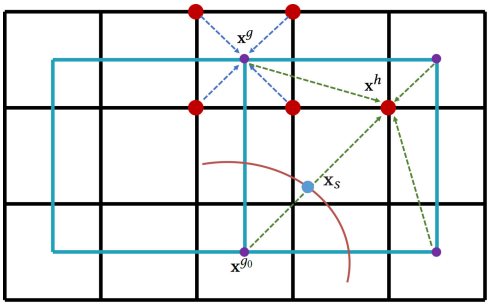

Fig. 9. **Staggered grids.** Our LBM grids for flow and phase fields are different; the phase field grid (with red nodes above) is twice as fine as the flow grid (with magenta nodes) for higher spatial accuracy of the fluid interface. Coupling between the two grids is particularly simple because of their staggered spatial locations: generally, if a phase value is needed at a flow node $\mathbf{x}^g$, an equal-weighted average of the corners of the corresponding fine grid is used (top), while if a flow value (like the macroscopic velocity) is needed at a phase field node $\mathbf{x}^h$, a simple multilinear interpolation between the corners of the corresponding coarse flow cell is performed (right).

- if a coarse node $\mathbf{x}^g$ storing the distribution $\mathbf{g}$ needs to evaluate any function of the phase field, it is first evaluated on the fine grid for all the corners of the fine grid cell corresponding to the coarse node $\mathbf{x}^g$ and then averaged with equal weighting;
- reversely, if a fine node $\mathbf{x}^h$ storing the distribution $\mathbf{h}$ needs to evaluate the velocity $\mathbf{u}$ locally, we use a trilinear interpolation from the corners of the coarse grid cell that $\mathbf{x}^h$ is in.

These two rules make the access of coupling values (macroscopic velocity and phase field) simple, even if the grids are not collocated.

Time integration. Having two different grid resolutions also requires a specific time integration process, as grid resolution and time-step sizes are coupled in LBM. Here again, the grid ratio being 2 makes for a simple approach: once distribution functions $\mathbf{g}_k$ and $\mathbf{h}_k$ are known at time t_k, the usual three-step (streaming, collision, boundary handling) LBM integration of the flow equation can be performed using the latest phase field evaluation derived from $\mathbf{h}_k$, leading to a new distribution function $\mathbf{g}_{k+1}$ at time t_{k+1}. We then proceed with two LBM updates of the phase field discretizations: we first compute $\mathbf{h}_{k+\frac{1}{2}}$ using the macroscopic velocity derived from $\mathbf{g}_k$, then compute $\mathbf{h}_{k+1}$ using the averaged macroscopic velocity derived from $\frac{1}{2}(\mathbf{g}_k + \mathbf{g}_{k+1})$. Fig. 10 summarizes this integration process.

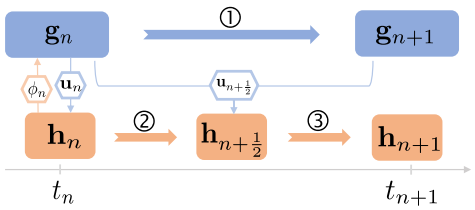

Fig. 10. **Time integration.** A time update from t_n to t_{n+1} is performed in three steps: (1) first, the flow equation is integrated using the phase field at time t_n (top); then the phase field is updated in two (2)+(3) substeps (bottom), first using the macroscopic velocity at time t_n, then using the average of the velocity at t_n and the one at t_{n+1}. Coupling between flow and phase field equations is indicated through hexagonal labels here.

3.5 Improving (un)normalized phase gradient

The evaluation of the *normalized phase gradient* $\nabla\phi/|\nabla\phi|$ (also called the interface normal $\mathbf{n}$) present in Eq. (2) is key to the correct time evolution of the phase field, and thus, it plays a major role in the accuracy of all the interfacial forces and coupling terms. Consequently, Li and colleagues [2021; 2022] used a rotationally-symmetric evaluation $\nabla^{[2nd]}\phi$ offering an efficient second-order accurate approximation of the gradient of the phase field before normalizing it. However, even when computed on a finer grid, their approximation suffers from grid-induced oscillations: Fig. 11 (left) shows the type of ringing artifacts that even a nearly still fluid exhibits, made visually more obvious using a fast-changing colormap. While these patterned oscillations did not affect stability in their static obstacle context, their very existence is extremely likely to lead to instability when simulating dynamic solid coupling, see Fig. 12.

In [Geier et al. 2015], a quest for the evaluation of an interface normal $\mathbf{n} := \nabla\phi/|\nabla\phi|$ which would not suffer from grid artifact led the authors to compute the normal not via differentiation, but via a direct central first-moment transform:

$$\mathbf{n}^* \propto \sum_i h_i \, (\mathbf{u} - \mathbf{d}_i) \tag{16}$$

where $\propto$ is used to indicate that a normalization is needed to return the unit normal $\mathbf{n}^*$. While this evaluation assumes a quasi-static phase field, it offers a very different and very localized way to approximate the normalized gradient.

Upon numerical testing, we found that these two different approaches to the normalized phase field gradient are in fact complementary. As Fig. 11 illustrates, $\mathbf{n}^*$ is clearly far less accurate in the air portion of the fluid, but also far smoother (and ringing-free) in the bulk of the heavy fluid. Thus, we propose a new approximation scheme combining their strengths through:

$$\frac{\nabla\phi}{|\nabla\phi|} := (1-\phi)\,\frac{\nabla^{[2nd]}\phi}{|\nabla^{[2nd]}\phi|} + \phi\,\mathbf{n}^*, \tag{17}$$

where the weighting based on the local phase field value ensures that the best estimate is used.

The evaluation at nodes of the grid storing the distribution $\mathbf{g}$ of the *unnormalized gradient* $\nabla\phi$ of the phase field is also crucial to prevent instabilities due to coupling. Near solid boundaries (i.e., on cut-cell nodes), we follow precisely the spatial evaluation strategy described in Sec. 3.4; i.e., we evaluate $\nabla^{[2nd]}\phi$ at the corners of the fine phase grid cell before averaging their values on the coarse flow grid node:

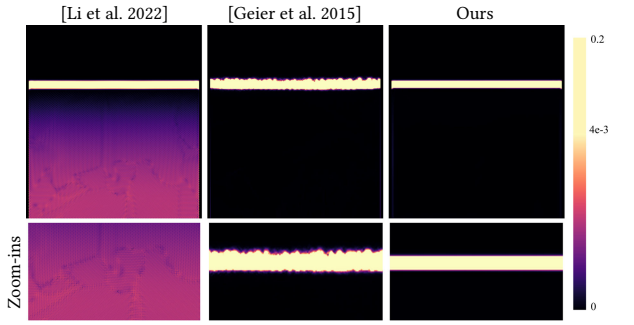

Fig. 11. **Enhancing accuracy of interface normal.** The images above visualize the magnitude of the gradient of the phase field for a nearly hydrostatic pool. While the finite-difference-based method of [Li et al. 2022] (left) gets a seemingly nice interface, the phase field gradient in fact contains ubiquitous grid-induced oscillations in the bulk of the heavy fluid as highlighted in a close-up view below. Reversely, the quasi-static approximation from [Geier et al. 2015] shows artifact-free phase gradients in the heavy fluid, but suffers from massive distortion near the light fluid (middle). By combining these two approximations (right), our weighted scheme for normal computation (Eq. (17)) has no visible artifact anywhere.

it provides the most isotropic second-order accurate approximation in these regions of fluid-solid interactions, which we denote by $\nabla^{[2nd]}\phi^{[h]}$ to express that the gradient is evaluated on the $\mathbf{h}$ grid before being transferred to the $\mathbf{g}$ grid. However, the same strategy in the bulk of the heavy fluid is ill-advised: as discussed earlier, small oscillations are present which can easily bring instability. We thus alter the gradient evaluation: we introduce an evaluation, denoted $\nabla^{[2nd]}\phi^{[g]}$ which, instead, first compute the phase field ϕ directly on nearby neighbors of the coarse flow grid, and only then use the second-order accurate evaluation of the gradient *directly on the coarse flow grid*. Note that this interpolation-then-differentiation approach corresponds to a larger spatial stencil than the previous differentiation-then-interpolation, and is thus less likely to suffer from the small oscillations mentioned above. Finally, we then use an evaluation which blends these two evaluations based on the value of the local phase field to determine whether we are in the bulk of the heavy fluid or not. That is, the gradient of the phase field is achieved through:

$$\nabla\phi := \begin{cases} \nabla^{[2nd]}\phi^{[h]}, & \text{in cut-cell nodes} \\ (1-\phi)\,\nabla^{[2nd]}\phi^{[h]} + \phi\,\nabla^{[2nd]}\phi^{[g]}, & \text{otherwise.} \end{cases} \tag{18}$$

Except for these new expressions, we strictly follow [Li et al. 2022] in their evaluation of interfacial forces, collisions, or wetting conditions — modulo the trilinear interpolations needed at times when interfacing the grids of $\mathbf{g}$ and $\mathbf{h}$ as described in Sec. 3.4.

3.6 Cut-cell updates

Since we are focusing on two-way coupling in this paper, intersection between links and boundaries have to be computed on the fly as obstacles continuously moves in time. Moreover, since we adopted a bounce-back based approach to coupling as described in Sec. 3.2, we have to track fresh *vs.* dead cells: traditionally, new distribution functions need to be assigned for *fresh* grid nodes (i.e., fluid nodes that were in the moving solid at the previous time step), while clean up of distribution functions need to be performed for dead cells (i.e.,

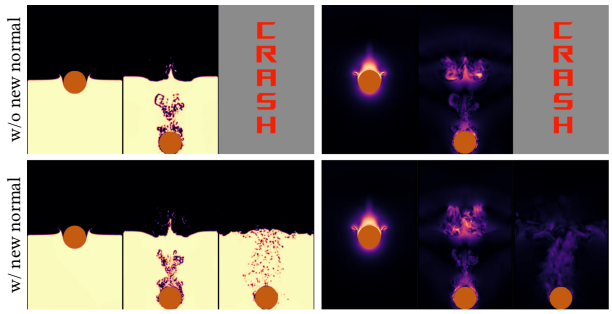

Fig. 12. **Ablation test for interface normal computation.** On this example of a ball dropping in a water tank (left: visualizing the phase field; right: visualizing the velocity field), if the normalized phase gradient is not handled through Eq. (17), the simulation crashes when the ball hits the bottom (top); with our new interface normal evaluation (and no other change in the simulation), the dynamics is properly captured (bottom).

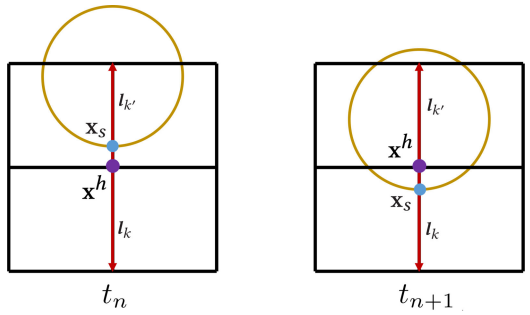

Fig. 13. **Dead cut-cells.** When phase grid node $\mathbf{x}^h$ gets covered by a solid (like the yellow disk here) between two time steps, it becomes a dead node, and its distributions h_k are then set to zero.

fluid nodes that become inside a moving solid); moreover, the choice of a "refilling" interpolation scheme used to derive fresh values from nearby nodes (similar to the Gauss-Jacobi process for invalid nodes in the context of Navier-Stokes fluid solvers [Guendelman et al. 2005]) can greatly influence the simulation accuracy around solid boundaries. As we show next, our particular setup makes the update of cut-cells quite simple.

Cut-cell treatment for distribution **g**. Given our use of the hybrid moving bounce-back described in Sec. 3.2, no specific treatment is required when updating the fluid flow: whether a grid node is covered by a solid object or not, the same exact process applies. A typical "fictitious flow" will happen inside objects, but with no negative impact on the physical behavior of the real flow.

Cut-cell treatment for distribution **h**. The treatment of the phase field is slightly more involved: since the phase equation and solid dynamics are only weakly coupled, even with a finer time step as we explained in Sec. 3.4, leakage could happen as seen in Fig. 13. First, we need to rapidly detect grid nodes that are becoming "dead", i.e., nodes over which a solid object is suddenly stepping. This is achieved by keeping track, for each phase-field node, of the links that are cut by a solid object. Given our use of D3Q7, we only need 6 boolean values per node to store the presence of cut-links in an array `cut-link[k]`. Then our detection of dead nodes becomes simple: if a current link ℓ_k from a node $\mathbf{x}$ is intersecting an object, we check if in the previous time step, the link ℓ_k at $\mathbf{x}$ was not cut but its opposite link $\ell_{k'}$ was, i.e., `(cut-link[k]==0)&&(cut-link[k']==1)`. Once this "dead" test is done (Fig. 13), we update the cut-link array for the next time step, and we *clean up the values of all dead nodes by setting all their* h_k *to zero*. This corresponds to the equilibrium distribution of $\phi=0$, meaning that we artificially set to light-air any node which becomes covered by a solid object to prevent artifacts. While other authors also suggest a specific treatment for fresh cells, our two half-steps for the integration of **h** (see Fig. 10) renders this step unnecessary: there is no need to specifically identify fresh cells since they will, like other regular cells, be automatically filled in via streaming from their neighbors . A final note: unlike for the distribution **g**, the bounce-back for **h** does *not* include a velocity, as the phase field is a scalar; cut-links are thus only performing a plain bounce-back: $h_k(\boldsymbol{x}_b, t+1) = h_{k'}(\boldsymbol{x}_b, t)$.

Solid velocity for phase advection. Finally, when a phase field node $\mathbf{x}^h$ computes the macroscopic flow velocity using trilinear interpolation from the corners of the bounding coarse flow cell, we replace the macroscopic velocity of every cut-cell corner by the solid's velocity evaluated at the link intersection point $\mathbf{x}_s$ (see Fig. 9), to account for the presence of an obstacle during the collision step.

3.7 Implementation details

We give pseudocode of our simulation solver in Alg. 1 to summarize the order in which the various steps are performed, before going through the few remaining steps we did not discuss until now.

Memory usage. Because we use two different grid resolutions, comparing memory usage with [Li et al. 2022] is not entirely straightforward. However, for multiphase flows, one can argue that it is the resolution of the interface which matters visually. For a phase resolution of $N\times N\times N$, [Li et al. 2022] ends up using a total of $48N^3$ floats, while our method only uses $22.37N^3$ floats in practice: since we reduce the resolution of the D3Q27 lattice, the extra memory we save is vastly superior to the added cost of our `cut-link[]` array and the additional storage of the macroscopic velocity from the previous frame to provide a half-time averaged velocity for the phase field. Thus we save around 54% memory compared to [Li et al.

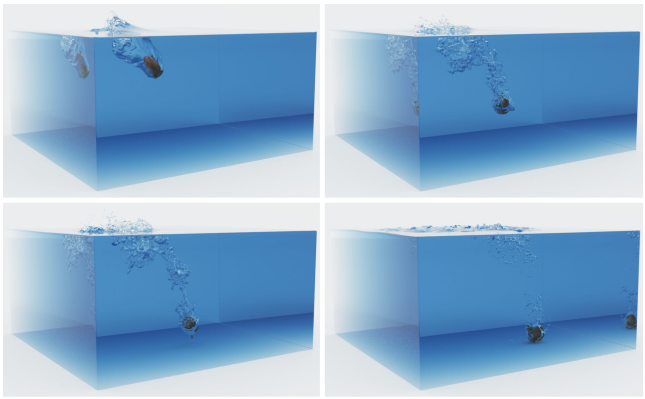

Fig. 14. **Onion Throwing.** A 3D model of an onion is thrown into a water tank; as the initial splash affects its trajectory, bubbles are dragged along its path, and it comes to rest at the bottom of the tank.

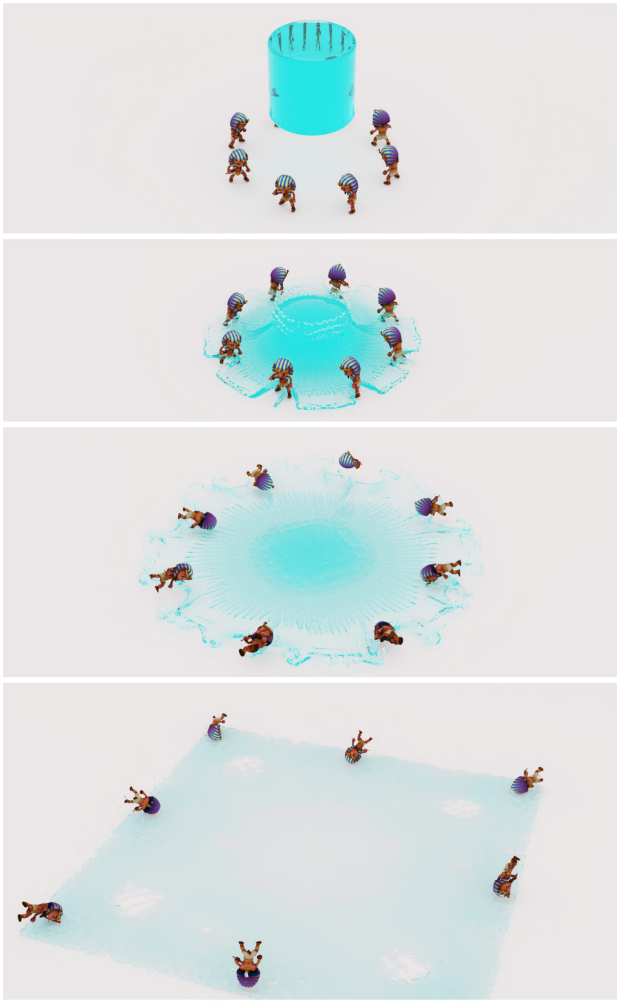

Fig. 15. **Toy Warriors.** As water is dropped, toy Egyptian warriors are swept away. This two-way coupling example clearly shows how the obstacles affect the flow of water, and how the water displaces the obstacles.

2022]. We list memory consumption for all our examples in Tab. 1 for completeness.

GPU-based implementation. Similar to [Li et al. 2022], our multiphase fluid-solid coupling solver was designed to maximize the locality of computations, thus benefitting from being implemented on GPU. The use of structure-of-arrays (SoA [Chen et al. 2021]) memory layout improves GPU memory access concurrency. For dynamic solids represented by a 3D mesh model, we need to perform link-mesh intersection at every iteration; to improve performance, we first rely on the bounding boxes of 3D models to determine cut-cells, i.e., cells that straddle an object; second, we perform link-mesh intersections for all cut-cell nodes during streaming and wetting boundary handling, and update the grid cut-link array accordingly. Note finally that we also gain efficiency compared to [Li et al. 2022] by only using their second-order rotationally-symmetric $\nabla\phi$ evaluation, not their third-order one: the use of a finer phase grid makes the second-order approximation sufficient, and it reduces the size of the evaluation stencil significantly — refer to Tab. 1 for timings.

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

```
t ← 0;
while t < T do
    GetRigidbodyTransformation at t_{1/2} and t_1;        ▷ Sec. 3.6
    FluidGridCutcellUpdate();                             ▷ Sec. 3.7
    PerformFlowCollision();                               ▷ Sec. 3.1
    FluidBoundaryCondition();                             ▷ Sec. 3.2
    FlowStreamingInRegular&CutCells();                    ▷ Sec. 3.1
    GetTwoWayForceOnSolid();                              ▷ Eq. 3.2
    UpdateFlowVelocity() at t_0 and t_1;                  ▷ Eq. 3.4
    GetCut-Link() at t_0;                                 ▷ Sec. 3.6
    while k < N do
        PhaseGridCutcellUpdate();                         ▷ Sec. 3.7
        CutcellTreatment();                               ▷ Sec. 3.6
        GetGradPhi();                                     ▷ Sec. 3.5
        PerformPhaseCollision();                          ▷ Sec. 3.1
        PhaseStreamingInRegular&CutCells();               ▷ Sec. 3.1
        UpdatePhaseField();                               ▷ Sec. 3.1
        UpdateCut-Link();                                 ▷ Sec. 3.6
        k ← k + 1;
    end
    UpdateFlowDensity();                                  ▷ Sec. 3.5
    WettingBoundaryConditionForPhase&FlowGrid;            ▷ Sec. 3.5
    CalculateForces();                                    ▷ Sec. 3.5
    UpdateMacroscopicVariablesWithForces();               ▷ Sec. 3.5
    RigidSolverUpdateWith (1/2)Δt and Δt;                 ▷ Sec. 3.6
    t ← t + 1;
end
```

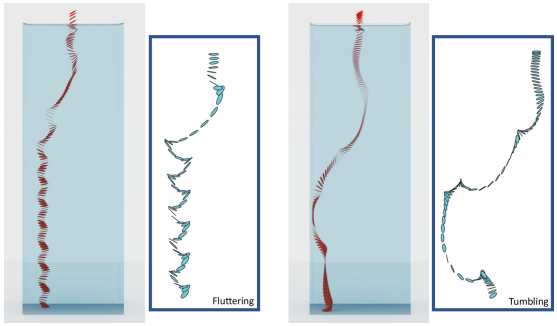

Fig. 16. **Coin toss in water.** Based on the initial conditions described in [Heisinger et al. 2014], our solver exhibits the expected fluttering (left) or tumbling (right) behaviors of a coin falling into a water tank; we show the coin at regular time intervals to help compare with the illustrations courtesy of [Heisinger et al. 2014] given as insets.

Soft-start initialization. In our context of two-way coupling, we often encounter situations involving a fast-moving solid. Starting an LBM simulation in this case often creates transient compression waves which can rapidly generate numerical instabilities. We thus use the soft-start initialization strategy suggested in [Li et al. 2021], adapted to our multiphase and dynamic coupling context: for the first 20 frames, we use the full weighted rotationally-symmetric evaluation of $\nabla\phi$ and systematically threshold any large "impulse" force as done in [Li et al. 2022]. This soft-start initialization quickly dampens all transient compression waves. Afterwards, we return

Table 1. **Statistics.** All examples run on a NVidia A6000 GPU, timings indicated for sequences at 60 frames-per-second.

Figure	Resolution (Flow/Phase)	Min./frame	v_H	v_L	ρ_H/ρ_L	Solid Mass (kg)	Domain dimensions (m^3)
1	330×480×270/660×960×540	1.54	0.00007	0.0028	800	0.1	0.33×0.48×0.27
2	224×350×224/448×700×448	1.10	0.00014	0.028	800	0.16/0.32/1.2	0.448 ×0.7×0.448
4 top	1120×140×336/2240×280×672	2.22	0.007	0.014	800	60,000	37.33×4.67×11.2
4 bottom	1120×140×336/2240×280×672	2.22	0.00014	0.00028	800	60,000	37.33×4.67×11.2
5 top	342×247×342/684×494×684	1.95	0.0038	0.038	800	0.3(min) to 2.9(max)	1.71×1.235×1.71
5 bottom	342×247×342/684×494×684	1.95	0.00038	0.0038	800	0.3(min) to 2.9(max)	1.71×1.235×1.71
7	1120×140×336/2240×280×672	2.22	0.00014	0.00028	800	1,000	37.33×4.67×11.2
8	432×120×432/864×240×864	1.82	0.0024	0.24	800	0.0075	0.54×0.15×0.54
14	360×260×360/720×520×720	1.60	0.000065	0.00013	800	0.3	0.792×0.572×0.792
15	420×180×420/840×360×840	1.50	0.000375	0.00075	800	0.07	1.596×0.684×1.596
16 left	180×600×150/360×1200×300	0.95	0.003	0.1	800	0.003	0.24×0.8×0.2
16 right	180×600×150/360×1200×300	0.95	0.0005	0.05	800	0.011	0.24×0.8×0.2
17	360×240×360/720×480×720	3.95	0.0002	0.003	800	2,000	18×12×18
18 [Li et al. 2022] (top)	800×360×400/800×360×400	3.60	0.0005	0.02	800	1.65	1.8×0.81×0.9
18 ours (bottom)	400×180×200/800×360×400	0.81	0.0001	0.0004	800	1.65	1.8×0.81×0.9
19	720×144×216/1440×288×432	1.12	0.00012	0.00014	800	0.71	7.20×1.44×2.16
20	456×190×456/912×380×912	1.84	0.000285	0.0057	800	2	15.2×6.33×15.2
21	320×256×320/640×512×640	1.24	0.00024	0.008	800	1	0.36×0.29×0.36
22	560×160×560/1120×320×1120	2.42	0.0003	0.008	1200	2	18.6×5.33×18.6
23	560×112×420/1120×224×840	1.35	0.00007	0.00014	800	0.142	4.80×0.96×3.60

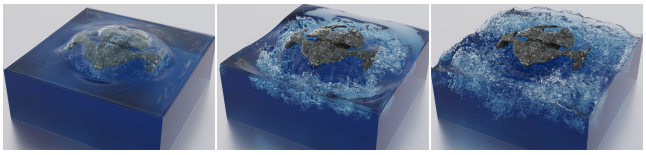

Fig. 17. **Spaceship surfacing from underwater**. A spaceship quickly emerging from deep underwater generates complex air-water interactions, including bubbles, splashes, wetting, and backwash.

to the cheaper and more local second-order rotationally-symmetric evaluation of $\nabla\phi$ and never perform any form of thresholding. This is an important advantage over [Li et al. 2022]: our accuracy improvements of fluid-solid coupling *remove* the need for artificial thresholding as systematically performed in [Li et al. 2022], or for a more complex adaptive temporal approach [Li et al. 2020].

4 RESULTS AND LIMITATIONS

We now go through our tests and results (including one-way and two-way examples, with complex, thin and/or thick moving solids), before discussing the remaining limitations and possible future works. Note that our tests used a CPU-based realtime rigid body solver [Coumans and Bai 2021], while the two-phase flow simulations were run on a variety of graphics cards, including an NVIDIA A100 with 40Gb and an NVIDIA RTX A6000 with 48Gb. We also used these GPUs to render our results using a modified GPU renderer based on [Jan van Bergen 2021].

4.1 One-way coupling

One-way coupling is simpler to achieve than two-way coupling as only the effects on a fluid of the scripted motion of a solid object are simulated. Fig. 4, for instance, shows an airplane moving through a pool of water for different Reynolds numbers. The wake flow generated by the moving airplane generates bubbles and splashing commensurate with the scale implied by the Reynolds number. (In fact, in the sequence shown on top of this figure, water even splashes all the way to the ceiling of the water container, as can be seen in the video.) Similarly, Fig. 7 shows a car entering at 40 mph into a pool of water, with a clear air bubble forming in front of the car. Fig. 17 shows a spaceship emerging from underwater and generating splashes and bubbles in the process. In Fig. 20, a propeller placed on an air-water interface starts spinning and creates the typical "frothing" action with bubbles, splashes and waves. Finally, Fig. 22 uses the same propeller, now placed higher up above the water. Its spinning creates an air flow which induces eddies and splashes.

4.2 Two-way coupling

Multiphase flow simulations involving two-way coupling are more challenging, and often lead to intricate motions. We now cover the various examples shown in this paper.

Coupling with solid objects. Fig. 14 shows an onion being thrown into a water container. When the onion breaks the water surface, a thin splash and bubbles are generated, and the motion of the onion is altered accordingly; bubbles are dragged along with the onion into the tank, before making their way back up to the surface. Fig. 21 demonstrates a similar scenario, but for a nozzle being dropped in a water tank this time. In Fig. 19, a coin-shaped stone is being skimmed on a still water surface, generating splashes and water waves. This two-way coupling example exhibits multiple bounces as expected from stone skipping.

Coupling with thin objects. Handling thin objects in the context of multiphase flows is challenging. We demonstrate a number of examples with thin shells to demonstrate how our simulator can seamlessly handle this case as well. For instance, Fig. 2 shows thin cups of various densities being dropped in a water tank: a light cup ends up floating, while heavier cups make their way down more or less fast, as the water rushes inside and pushes air away and makes the cup fall to the bottom of the tank.

Coupling with lighter objects. We also test the dropping of a very light paper boat onto water: here, surface tension keeps the paper boat afloat, either straight up, or upside down depending on how the boat was initially dropped. Surface-tension driven (i.e., capillary) water waves are also generated by the boat. Fig. 23 shows a light

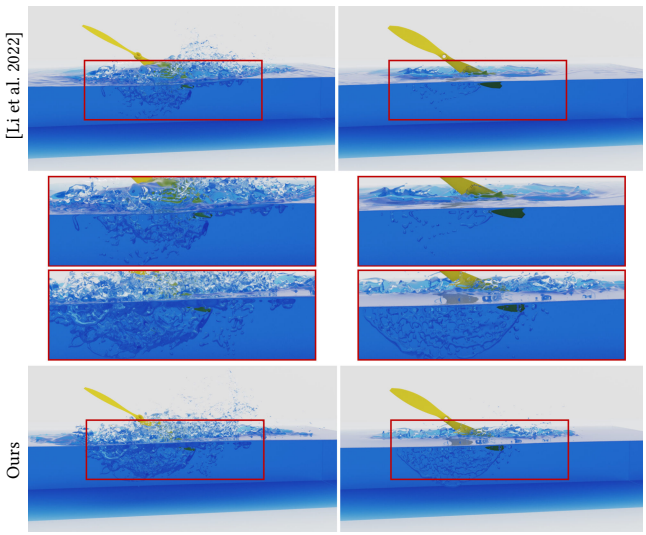

Fig. 18. **Comparison with [Li et al. 2022].** We compare a one-way coupling example with two different propellers (left: off-centered axis of rotation to create more splashes; right: correct rotation axis) for the immersed boundary method of Li et al. [2022] (top) and our solver (bottom), which exhibits finer turbulent details (see close-ups) as it can handle lower air viscosity and more accurate pressure boundary conditions to create bubbles.

disk (five times lighter than the stone skipping example from Fig. 19) which first bounces on a calm pool surface, before sliding for a while due to surface tension.

Multiple-object two-way coupling. While previous results only showed a single dynamic object interacting with the fluid, we can trivially handle arbitrary numbers of objects. Fig. 15 shows a water column being dropped onto the floor and blowing toy warriors away in the process; warriors flip over, and the water reacts to the warriors' movement as expected. Finally, Fig. 5 shows cows and bunnies of various densities being dropped into water.

4.3 Comparisons

Finally, we show comparisons between our solver and the previous work we built upon, as well as comparisons between real experiments and our attempts at reproducing them.

Comparison to [Li et al. 2022]. Fig. 18 shows two different propellers rotating while being half-immersed in water. While one of the propellers is rotating exactly around its proper axis of rotation, the other one is made to wobble so as to generate many bubbles and splashing. To be able to compare with [Li et al. 2022] (which used the immersed boundary method on the few examples of moving objects that they provide) fairly, we match their phase field resolution. We observe that our solver (bottom) generates more bubbles and splashing, while their poor handling of the pressure boundary condition results in weaker coupling than expected (top). Note also that water splashes are very smoothed out because of the high air viscosity used in [Li et al. 2022] to ensure stability. Importantly, our solver can handle an eight-fold increase in propeller speed, while theirs fails as soon the rotation speed increases even slightly. Finally, our solver is also around 4.5 times faster on this example.

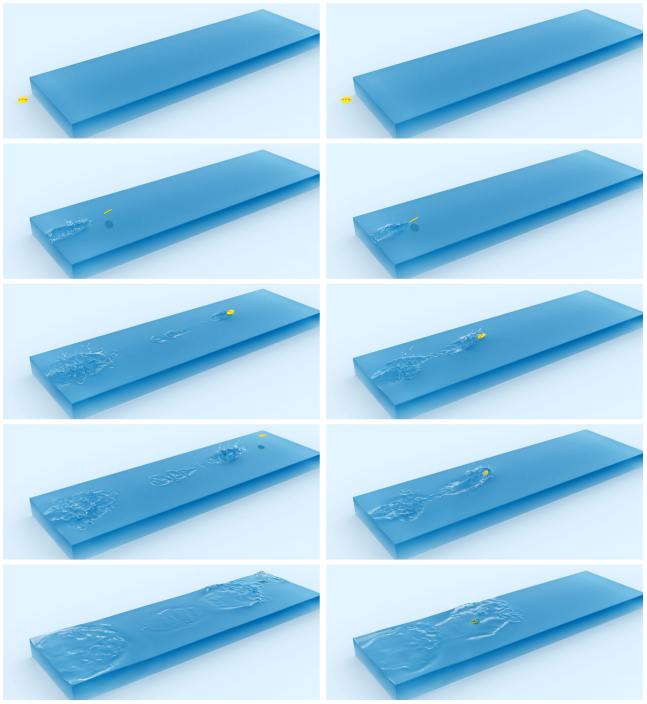

Fig. 19. **Stone Skipping.** A rotating disk labeled 'ACM' is thrown on a flat water surface with two slightly different initial velocities, exhibiting the usual stone-skipping behavior (left: multiple bounces, with the wake flow creating ripples; right: one bounce before sinking).

Comparisons to real-life experiments. There are many exquisitely-fine phenomena induced by multiphase flows and solid coupling, and the ubiquity of high-speed cameras offers us a large playground to evaluate our numerical solver. First, we compare our solver using an example of a video sequence found online [DepositPhotos 2021] showing a key dropped into water: the key generates a veil of bubbles at an early stage; as the key drops down, the veil shrinks and turns into a complex shape, and our simulation captures the same qualitative behavior as seen in Fig 1. Second, we also simulate a coin falling in water, a phenomenon that has well-documented types of trajectory depending on initial conditions, the dimensionless moment of inertia, and the Reynolds number [Heisinger et al. 2014]. Fig. 16 shows a "fluttering" fall and a "tumbling" fall, matching the expected real behaviors as depicted in [Heisinger et al. 2014].

5 CONCLUSION

In this paper, we present a kinetic two-phase flow solver with two-way coupling which can simulate in a massively-parallel fashion the dynamic interaction between solid structures (be they thick or thin, light or heavy) and the two phases of a real-life fluid (e.g., with a density ratio of 800 between air and water) at low or high Reynolds numbers. We reach this milestone by altering key components of the recent work of [Li et al. 2022] in order to increase the resolution of the interface and the accuracy of crucial coupling forces based on the phase field gradient. As a result, we no longer need to use artificially high viscosity in air or force thresholding to maintain stability, even for large interaction speeds. This opens up a vast area of possible simulations of daily phenomena, of which we only scratched the

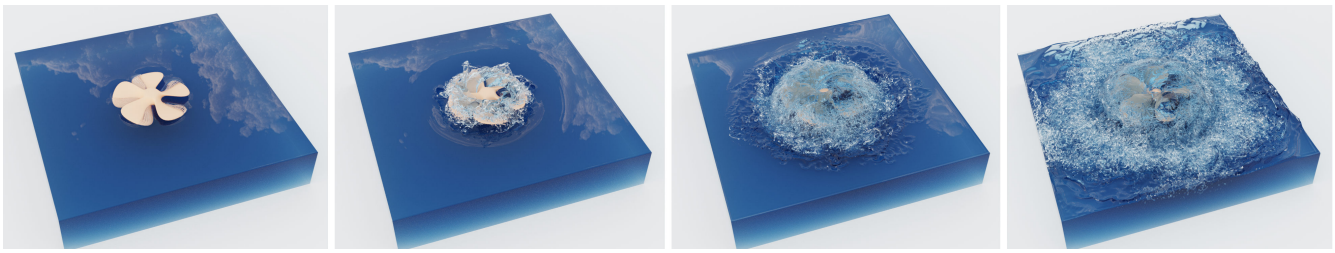

Fig. 20. **Water frothing**. A propeller starts spinning near the water surface, quickly generating splashing, waves and bubbles.

surface here by showing the correct numerical capturing of stone skipping or coin falling to name a few.

Future works abound, however, if we wish to keep improving the efficiency and accuracy of fluid-structure coupling. For instance, our treatment of dead cells is particularly efficient, but does not guarantee exact mass preservation; the addition of Lagrangian particles could correct this limitation. Moreover, we have not tested our coupling with *deformable* solids, which may require additional efforts to prevent blowups. Finally, we found out that many real-time rigid body solvers have unacceptably poor accuracy in their collision treatment; their tendency to exhibit fast oscillations during impact for instance can be a source of instability when coupled with a two-phase flow solver at high Reynolds numbers. Further improving our coupling in order to accommodate these typical flaws of rigid body solvers without having recourse to a monolithic solver is one of the many interesting avenues of research.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers (and in particular, the shepherd) for helping us improve exposition. MD acknowledges the generous support of a Choose France Inria chair. 3D models of obstacles are courtesy of the Stanford Computer Graphics Laboratory (Fig. 5), from CGtrader's users chiragveerwani (Fig. 4), 3dassetlibrary (Fig. 14), and markinhofaci (Fig. 15), from Sketchlab's user mcgamescompany (Fig. 8), from TurboSquid's user elvencity (Fig. 17), from GrabCAD's user Dhanasekar Vinayagamoorthy (Fig. 18), and from Technische Universität München (Fig. 7).

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.

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

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), 97.

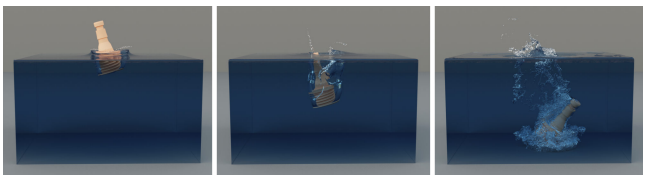

Fig. 21. **Nozzle cleaning**. A nozzle is dropped in water, generating bubbles and splashes. Note the water coming out of the outlet (middle frame).

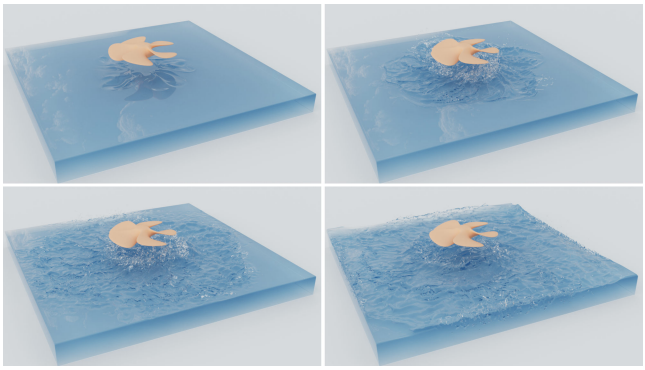

Fig. 22. **Air-driven eddies**. Same propeller as Fig. 20, but now placed above water; its spinning creates eddies and splashes on an initially still surface.

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

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. In *ACM Trans. Graph.*, Vol. 26. 100.

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

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

Mark Carlson, Peter J. Mucha, and Greg Turk. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM Trans. Graph.*, Vol. 23. 377–384.

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.

Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. 2021. GPU optimization for high-quality kinetic fluid simulation. *IEEE Trans. Vis. & Comp. Graph.* 28, 9 (2021), 3235–3251.

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), 1–13.

Nuttapong Chentanez, Tolga Goktekin, Bryan Feldman, and James O'Brien. 2006. Simultaneous coupling of fluids and deformable bodies. In *Symp. Comp. Anim.* 83–89.

Junghyun Cho and Hyeong-Seok Ko. 2013. Geometry-aware volume-of-fluid method. In *Computer Graphics Forum*, Vol. 32. 379–388.

Pascal Clausen, Martin Wicke, Jonathan R. Shewchuk, and James F. O'Brien. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Trans. Graph.* 32, 2, Article 17 (2013).

Erwin Coumans and Yunfei Bai. 2016–2021. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Meizhong Dai and David P Schmidt. 2005. Adaptive tetrahedral meshing in free-surface flow. *J. Comput. Phys.* 208, 1 (2005), 228–252.

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 Angelantonio Tafuni. 2022. A phase-field lattice Boltzmann method for the solution of water-entry and water-exit problems. *Computer-Aided Civil and Infrastructure Engineering* 37, 7 (2022), 832–847.

DepositPhotos. 2021. Key falling into water against white background — slow motion 4K. https://depositphotos.com/156702318/ From user Slowmotiongli.

Mathieu Desbrun and Marie-Paule Cani-Gascuel. 1998. Active implicit surface for animation. In *Graphics Interface*. 143–150.

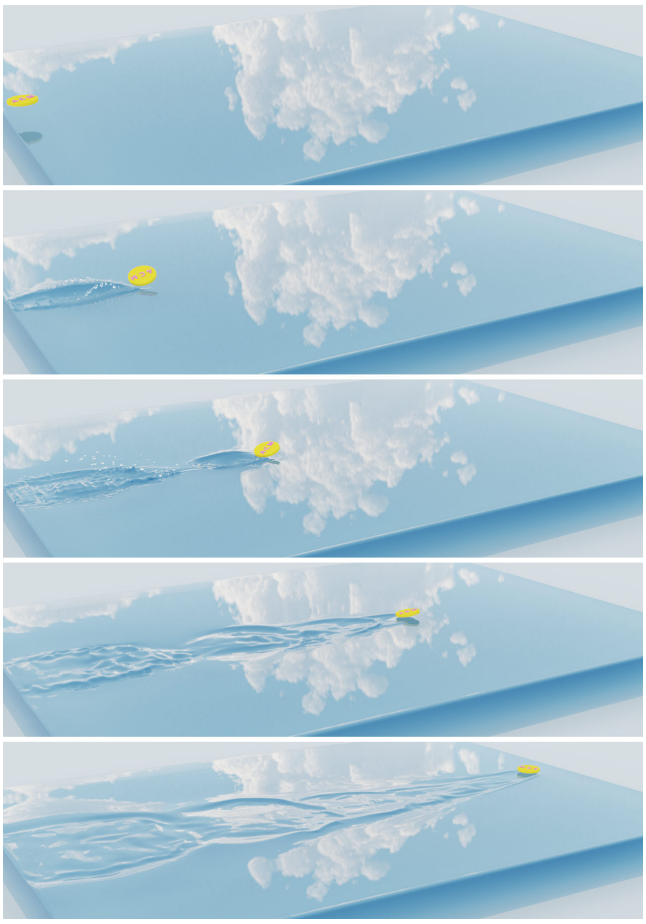

Fig. 23. **Disk Sliding.** A light rotating disk is thrown over a calm pool, sliding and bouncing over the surface; compare to the stone skipping example of Fig. 19, using a heavier disk.

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

Essex Edwards and Robert Bridson. 2014. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Trans. Graph.* 33, 4, Article 136 (2014).

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.* 26, 1, Article 4 (2007).

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 (2020), 51–1.

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 (2019).

Bryan Feldman, James O'Brien, Bryan Klingner, and Tolga Goktekin. 2005b. Fluids in deforming meshes. In *Symp. Comp. Anim.* 255–259.

Bryan E Feldman, James F. O'Brien, and Bryan M Klingner. 2005a. Animating gases with hybrid meshes. In *ACM Trans. Graph.*, Vol. 24. 904–909.

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

Martin Geier, Abbas Fakhari, and Taehun Lee. 2015. Conservative phase-field lattice Boltzmann model for interface tracking equation. *Phys. Rev. E* 91, 6 (2015), 063309.

Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. 2003. Simulating fluid-solid interaction.. In *Graphics Interface.* 31–38.

Frédéric Gibou and Chohong Min. 2012. Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions. *J. Comput. Phys.* 231, 8 (2012), 3246–3263.

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

Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.* 24, 3 (2005), 973–981.

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

Luke Heisinger, Paul Newton, and Eva Kanso. 2014. Coins falling in water. *Journal of fluid mechanics* 742 (2014), 243–253.

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, Article 150 (2018).

Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L. Michels, and Chenfanfu Jiang. 2021. Ships, splashes, and waves on a vast ocean. *ACM Trans. Graph.* 40, 6, Article 203 (2021).

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.

Jan van Bergen. 2021. GPU-Raytracer. (2021). https://github.com/jan-van-bergen/

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

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

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

Doyub Kim, Oh-Young Song, and Hyeong-Seok Ko. 2010. A practical simulation of dispersed bubble flow. *ACM Trans. Graph.* 29, 4 (2010), 70:1–5.

Bryan Klingner, Bryan Feldman, Nuttapong Chentanez, and James O'Brien. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graph.* 25, 3 (2006), 820–825.

Dan Koschier and Jan Bender. 2017. Density maps for improved SPH boundary handling. In *Symp. Comp. Anim.* Article 1.

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

Timothy R. Langlois, Changxi Zheng, and Doug L. James. 2016. Toward animating water with complex acoustic bubbles. *ACM Trans. Graph.* 35, 4 (2016), 95:1–13.

Wei Li, Kai Bai, and Xiaopei Liu. 2019. Continuous-scale kinetic fluid simulation. *IEEE Trans. Vis. Comp. Graph.* 25, 9 (2019), 2694–2709.

Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and scalable turbulent flow simulation with two-Way coupling. *ACM Trans. Graph.* 39, 4 (2020).

Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. 2021. Kinetic-based multiphase flow simulation. *IEEE Trans. Vis. & Comp. Graph.* 27, 7 (2021), 3318–3334.

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).

Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-reduced variational fluid simulation. *ACM Trans. Graph.* 34, 6, Article 244 (2015).

Jinyuan Liu, Mengdi Wang, Fan Feng, Annie Tang, Qiqin Le, and Bo Zhu. 2022. Hydrophobic and hydrophilic solid-fluid interaction. *ACM Trans. Graph.* 41, 6 (2022).

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

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

Viorel Mihalef, Betul Unlusu, Dimitris Metaxas, Mark Sussman, and M Yousuff Hussaini. 2006. Physics based boiling simulation. In *Symp. Comp. Anim.* 317–324.

Marek Krzysztof Misztal, Robert Bridson, Kenny Erleben, Jakob Andreas Bærentzen, and François Anton. 2010. Optimization-based fluid simulation on unstructured meshes. In *VRIPHYS.* 11–20.

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.
Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Symp. Comp. Anim.* 154–159.
Yen Ting Ng, Chohong Min, and Frédéric Gibou. 2009. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comput. Phys.* 228, 23 (2009), 8807–8829.
Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An implicit viscosity formulation for SPH fluids. *ACM Trans. Graph.* 34, 4 (2015), 1–10.
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. *Computers & Mathematics with Applications* 72, 2 (2016), 349–374.
Ziyin Qu, Minchen Li, Fernando De Goes, and Chenfanfu Jiang. 2022. The power particle-in-cell method. *ACM Trans. Graph.* 41, 4 (2022), 1–13.
Bo Ren, Chenfeng Li, Xiao Yan, Ming C Lin, Javier Bonet, and Shi-Min Hu. 2014. Multiple-fluid SPH simulation using a mixture model. *ACM Trans. Graph.* 33, 5 (2014), 171:1–11.
Avi Robinson-Mosher, R. Elliot English, and Ronald Fedkiw. 2009. Accurate tangential velocities for solid fluid coupling. In *Symp. Comp. Anim.* 227–236.
Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. In *ACM Trans. Graph.*, Vol. 27. 46.
Doug Roble, Nafees Bin Zafar, and Henrik Falt. 2005. Cartesian grid fluid simulation with irregular boundary voxels. In *ACM SIGGRAPH 2005 Sketches*. 138.
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, Article 120 (2021).
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).
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 (07 2022).
Yos Panagaman Sitompul and Takayuki Aoki. 2019. A filtered cumulant lattice Boltzmann method for violent two-phase flows. *J. Comput. Phys.* 390 (2019), 93–120.
Barbara Solenthaler and Renato Pajarola. 2008. Density contrast SPH interfaces. In *Symp. Comp. Anim.* 211–218.
Barbara Solenthaler and Renato Pajarola. 2009. Predictive-corrective incompressible SPH. *ACM Trans. Graph.* 28, 3 (2009), 40:1–6.
Oh-Young Song, Hyuncheol Shin, and Hyeong-Seok Ko. 2005. Stable but nondissipative water. *ACM Trans. Graph.* 24, 1 (2005), 81–97.
Haozhe Su, Tao Xue, Chengguizi Han, Chenfanfu Jiang, and Mridul Aanjaneya. 2021. A unified second-order accurate in time MPM formulation for simulating viscoelastic liquids with phase change. *ACM Trans. Graph.* 40, 4 (2021), 1–18.
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.
Tsunemi Takahashi, Heihachi Ueki, Atsushi Kunimatsu, and Hiroko Fujii. 2002. The simulation of fluid-rigid body interaction. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*. 266–266.
Michael Tao, Christopher Batty, Mirela Ben-Chen, Eugene Fiume, and David I. W. Levin. 2022. VEMPIC: Particle-in-polyhedron fluid simulation for intricate solid boundaries. *ACM Trans. Graph.* 41, 4 (2022).
Yun Teng, David I. W. Levin, and Theodore Kim. 2016. Eulerian solid-fluid coupling. *ACM Trans. Graph.* 35, 6, Article 200 (2016).
Nils Thuerey, Filip Sadlo, Simon Schirm, Matthias Müller-Fischer, and Markus Gross. 2007. Real-time simulations of bubbles and foam within a shallow water framework. In *Symp. Comp. Anim.* 191–198.
Daniel Weber, Johannes Mueller-Roemer, André Stork, and Dieter Fellner. 2015. A cut-cell geometric multigrid Poisson solver for fluid simulation. In *Comp. Graph. Forum*, Vol. 34. 481–491.
Yucheng Xiao, Guiyong Zhang, Da Hui, Haoran Yan, Song Feng, and Shuangqiang Wang. 2022. Numerical simulation for water entry and exit of rigid bodies. *Journal of Fluids and Structures* 109 (2022), 103486.
Jingrui Xing, Liangwang Ruan, Bin Wang, Bo Zhu, and Baoquan Chen. 2022. Position-based surface tension flow. *ACM Trans. Graph.* 41, 6 (2022), 1–12.
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), 1–13.
Xiao Yan, Yun-Tao Jiang, Chenfeng Li, Ralph R. Martin, and Shi-Min Hu. 2016. Multiphase SPH simulation for interactive fluids and solids. *ACM Trans. Graph.* 35, 4, Article 79 (2016).
Shuqi Yang, Shiying Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. 2021. Clebsch gauge fluid. *ACM Trans. Graph.* 40, 4 (2021), 1–11.
Tao Yang, Jian Chang, Ming C. Lin, Ralph R. Martin, Jian J. Zhang, and Shi-Min Hu. 2017. A unified particle system framework for multi-phase, multi-material visual simulations. *ACM Trans. Graph.* 36, 6 (2017), 224:1–13.
Gary Yngve, James O'Brien, and Jessica Hodgins. 2000. Animating explosions. In *Annual Conference on Computer Graphics and Interactive Techniques*. 29–36.
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.