Adaptive Phase-Field-FLIP for Very Large Scale Two-Phase Fluid Simulation

BERNHARD BRAUN, Technical University of Munich, Germany
JAN BENDER, RWTH Aachen University, Germany
NILS THUEREY, Technical University of Munich, Germany

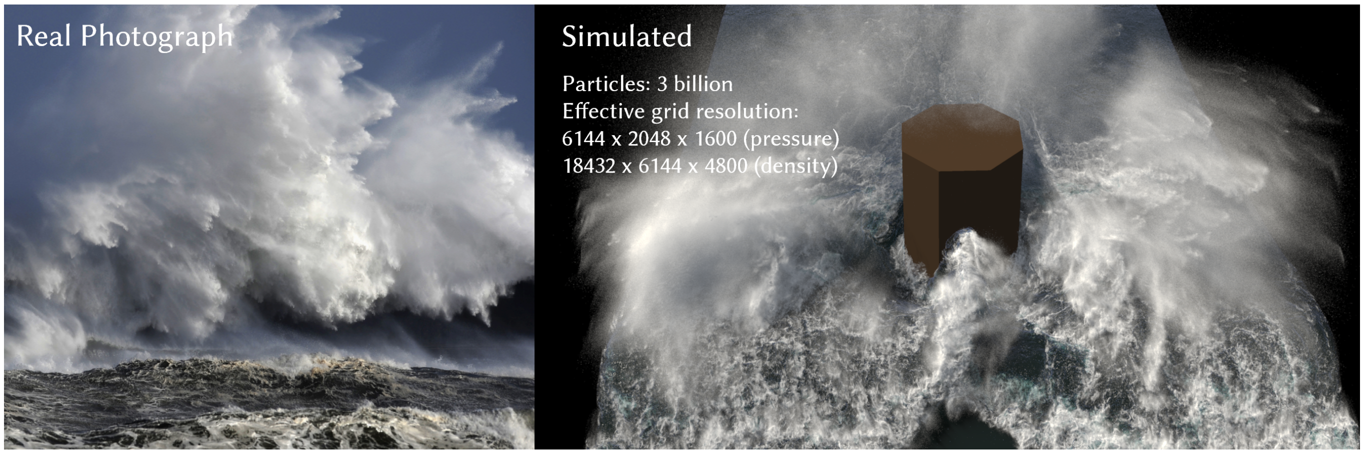

Real Photograph
Simulated
Particles: 3 billion
Effective grid resolution:
6144 x 2048 x 1600 (pressure)
18432 x 6144 x 4800 (density)

Fig. 1. Our simulation method reproduces the intricate and complex two-phase air-water interactions in very large-scale scenarios. It achieves this in a physics-based manner, resolving both phases at cinematic resolution on a single workstation: the example above (right) has ca. 3 billion particles, simulated with an average of only 2 minutes per time step. No procedural effects or post-processing spray heuristics were employed. Photograph © PA / Eloy Alonso.

Capturing the visually compelling features of large-scale water phenomena, such as the spray clouds of crashing waves, stormy seas, or waterfalls, involves simulating not only the water but also the motion of the air interacting with it. However, current solutions in the visual effects industry still largely rely on single-phase solvers and non-physical "white-water" heuristics. To address these limitations, we present Phase-Field-FLIP (PF-FLIP), a hybrid Eulerian/Lagrangian method for the fully physics-based simulation of very large-scale, highly turbulent multiphase flows at high Reynolds numbers and high fluid density contrasts. PF-FLIP transports mass and momentum in a consistent, low-dissipative manner and, unlike most existing multiphase approaches, does not require a surface reconstruction step.

Furthermore, we employ spatial *adaptivity* across all critical components of the simulation algorithm, including the pressure Poisson solver. We augment PF-FLIP with a dual multiresolution scheme that couples an efficient treeless adaptive grid with adaptive particles, along with a fast adaptive Poisson solver tailored for high-density-contrast multiphase flows. Our method enables the simulation of two-phase flow scenarios with a level of physical realism and detail previously unattainable in graphics, supporting billions of particles and adaptive 3D resolutions with thousands of grid cells per dimension on a single workstation.

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

Additional Key Words and Phrases: turbulent flows, large density ratio, two-phase flows, FLIP, spatial adaptivity, large-scale fluid simulation

ACM Reference Format:
Bernhard Braun, Jan Bender, and Nils Thuerey. 2025. Adaptive Phase-Field-FLIP for Very Large Scale Two-Phase Fluid Simulation. *ACM Trans. Graph.* 44, 4 (August 2025), 23 pages. https://doi.org/10.1145/3730854

Authors' Contact Information: Bernhard Braun, bernhard.braun@pbs.cit.tum.de, Technical University of Munich, Germany; Jan Bender, bender@cs.rwth-aachen.de, RWTH Aachen University, Germany; Nils Thuerey, nils.thuerey@tum.de, Technical University of Munich, Germany.

1 Introduction

Visual simulation of *large-scale* fluid phenomena —such as waterfalls, stormy seas, giant waves, or asteroid impacts— is arguably one of the most challenging and fascinating topics in computer graphics [Wang et al. 2024]. However, existing single-phase liquid simulators face significant difficulties in producing animations that fully convey the awe-inspiring scale and intensity inherent in these natural phenomena. Simply increasing the physical scale and resolution of a standard single-phase simulation is insufficient to address this challenge. For example, a single-phase simulation of an asteroid hitting the ocean gives the appearance of a rock being thrown into a pond in slow motion, conveying the impression of a much smaller scale, as illustrated in Figure 2. This occurs because natural fluid flows at large scales are inherently *two-phase* phenomena. They are governed by complex and highly nonlinear interactions between water and air that result in striking visual differences between small- and large-scale air-water flow phenomena. Typical examples, such as those in Figure 1, make clear that the (turbulent) motion of the air, made visible through spray and mist droplets, contributes at least as much—if not more—to the overall visual impression of large-scale water flows as the motion of the water itself.

Cinematic simulations of large-scale two-phase flows have so far relied on the use of numerous single-phase heuristics [Gissler et al. 2017; Ihmsen et al. 2012], complex (post)processing pipelines [Stomakhin et al. 2023; Stringhetti 2024], and labor-intensive manual or artistic interventions, often relying on distributed computing clusters to achieve cinematic resolutions. In this work, we present a fully physics-based method that produces such scenes from simple simulation setups on a single work station. To achieve this, our work addresses three central challenges: (i) two-phase simulation, (ii) spatial adaptivity, and (iii) efficient iterative pressure solvers.

Necessity of Two-Phase Simulation. Water simulations in Graphics often neglect the air phase, assuming constant zero air pressure. In these *free surface* methods, the governing Navier-Stokes equations are thus solved only for the water phase. This simplification, however, becomes insufficient for *large-scale* phenomena where the violent motion of large masses of water induces strong movements of the surrounding air and aerodynamic forces, which scale with the square of the relative air-water velocity, become dominant. The resulting high Weber numbers inhibit surface tension from stabilizing the liquid, leading to fragmentation into spray droplets. These droplets do not simply follow ballistic trajectories; instead, they form intricate, dynamic patterns driven by the complex interplay of gravity, inertial forces and—crucially—the turbulent air velocity field [Sirignano 2010]. Accurately capturing these phenomena requires both modeling the coupled liquid-air velocity and a numerical method powerful enough to resolve scales ranging from tiny spray droplets to large gravity-driven waves. Despite the immense visual importance of large-scale two-phase flows, much of the impressive recent progress in the field of fluid simulation [Aanjaneya et al. 2017; Ando et al. 2013; Ferstl et al. 2014; Fu et al. 2017; Liu et al. 2016; Nabizadeh et al. 2022; Qu et al. 2019; Shao et al. 2022] has so far been limited to single-phase cases.

When two-phase flows have been addressed, it has typically been in the context of smaller-scale, surface-tension driven flows [Boyd and Bridson 2012; Hong and Kim 2005] or medium-scale, real-time scenarios [Li et al. 2020, 2024]. In movie production, however, the lack of sufficiently scalable and fully physics-based two-phase solvers is one of the reasons why the special-effects industry still largely relies on heuristics, post-processing pipelines and manual intervention to produce convincing large-scale water animations [Stringhetti 2024].

Phase-Field FLIP. Choosing the appropriate computational representation for a simulated effect is a classic challenge. For liquids, grid-based (Eulerian) approaches have been successfully combined with Lagrangian methods, e.g., in the classic FLIP (Fluid Implicit Particles) approach [Bridson 2015], or for simulating multi-material flows [Hu et al. 2019b].

We leverage the versatility of hybrid particle-grid representations by introducing *Phase-Field FLIP* (PF-FLIP), a variant of FLIP for the efficient simulation of high density-contrast two-phase flows. PF-FLIP provides a simple and efficient interface-capturing scheme that eliminates the need for surface maintenance steps, as required in Volume of Fluid (VOF) or Level-Set methods [Sussman et al. 1994], or the evolution of PDEs, as in classical Phase-Field methods [Jacqmin 1999]. Our approach requires surface reconstruction only for rendering of the video frame, i.e. every 5 to 10 time steps. Its Lagrangian advection inherently conserves mass, avoiding the need for numerical safeguards [Jain 2022] required in traditional methods, and preventing inconsistencies often encountered in popular multiphase schemes like VOF, which typically rely on separate advection schemes for velocity and interface geometry.

High density ratios, such as 1000 : 1 for water and air, present a significant challenge for many traditional two-phase fluid solvers. One key idea of PF-FLIP is to turn this into an advantage by leveraging the stark density difference to reliably *separate phases*, even in the presence of FLIP-typical particle-induced noise. The 1000-fold higher density of water compared to air ensures a high "signal-to-noise" ratio, where noisy density fluctuations in the water phase remain clearly distinguishable from the noisy background of the air phase.

Spatial Adaptivity. Most existing multiphase fluid simulation methods rely on uniform grids or constant-size particles [Boyd and Bridson 2012; Li et al. 2022; Mihalef et al. 2009; Song et al. 2005; Yan and Ren 2023] sufficient for grid resolutions of several hundred cells per dimension. Achieving cinematic realism for large-scale simulations, however, requires both larger simulation domain sizes and substantially larger resolutions. As the computational cost scales with spatial dimensions and time, this yields a *fourth-order* scaling in terms of resolution. *Spatial adaptivity*, which focuses resources on high-priority regions, is thus essential to avoid an explosion in resource requirements.

Adaptivity encompasses *spatial sparsity* and *multiresolution* techniques. Spatial sparsity computes and stores data only in "active" regions [Ferstl et al. 2016; Hu et al. 2019a; Liu et al. 2016; Museth 2013; Shao et al. 2022; Wang et al. 2020]. Multiresolution, as employed in our algorithm, takes adaptivity and flexibility a step further by representing different simulation regions at varying resolutions, based on their complexity (as measured by a user-defined refinement criterion). Modern CPUs and GPUs rely on large caches, high core counts, and wide SIMD units for performance gains, making regularity and coherence in memory access critical. Adaptive schemes should therefore retain the efficiency of uniform grids locally, even within a globally irregular adaptive structure. Tree-based methods achieve this by associating leaf nodes with uniform sub-grids (*blocks*) but introduce administrative overhead and parallelization challenges.

As an efficient tree-less alternative, we introduce *Multiresolution Sparse Block Grids* (MSBG), a hardware-friendly scheme using simple linear indexing of large blocks. In Section 8.1, we show that MSBG outperforms the industry standard VDB by up to an order of magnitude and demonstrates excellent scalability, solving a test PDE with 100 billion unknowns at an effective resolution of 32768^3 in under two minutes on a single workstation.

For Eulerian-Lagrangian methods, it is moreover crucial that adaptivity targets both representations. We use a dual scheme: the grid adapts to particle sizes, while particle size and sampling density are dynamically adjusted based on refinement criteria from

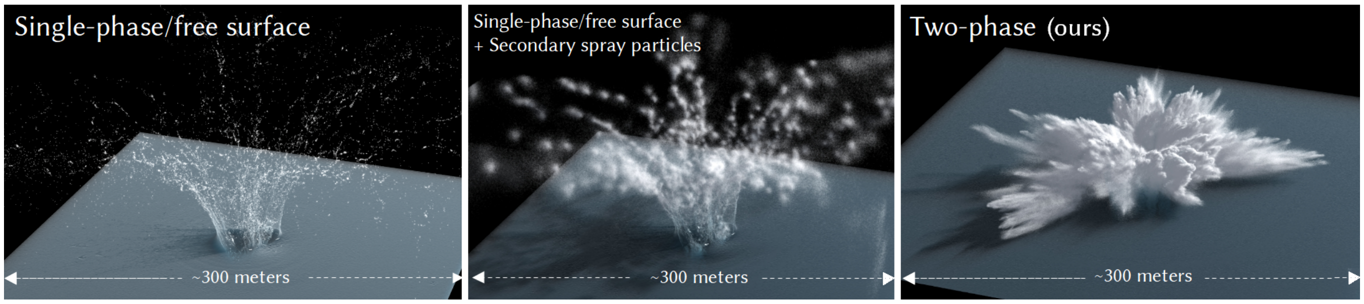

Single-phase/free surface
Single-phase/free surface
+ Secondary spray particles
Two-phase (ours)
~300 meters
~300 meters
~300 meters

Fig. 2. Motivating example: A 50-meter asteroid hits the ocean at high speed. Left: Standard single-phase solver [Bridson 2015]. Middle: With secondary spray particles [Ihmsen et al. 2012]. Right: Our two-phase simulation. Single-phase simulations give the impression of a small rock thrown into a pond in slow motion. Without air resistance, the water "splash rim" rises unrealistically high and spray particles diffuse unnaturally. In reality, spray follows intricate trajectories driven by the turbulent air velocity as captured by our simulation. All simulations used the same resolution of 1024^3 (pressure) and 3072^3 (spray density).

the grid. Previous methods for Lagrangian adaptivity [Adams et al. 2007; Ando et al. 2012, 2013] require particle neighborhoods. This, however, removes a key advantage of the FLIP method: its lightweight, noninteracting particles. To maintain this advantage, we propose a fast stochastic coarsening scheme that eliminates neighborhood searches altogether, ensuring that the Lagrangian part of our pipeline remains fully local and highly efficient.

Adaptive Linear System Solvers. A final potential bottleneck remains in the form of the very large linear Poisson problem that needs to be solved in pressure projection-based Navier-Stokes simulations. In the context of two-phase flows, the pressure system also becomes highly ill-conditioned for large density ratios. To address this, we propose a fully adaptive Poisson solver based on (unsmoothed) algebraic aggregation multigrid [Shao et al. 2022]. Leveraging MSBG, we extend the method to support *multigrid multiresolution* adaptivity. A two-stage-red-black hybrid Gauss-Seidel smoother ensures massive parallelizability while halving the memory footprint compared to conventional parallel smoothers.

Crucially, our MSBG-block-based multigrid V-cycle enables the efficient integration of *spatially adaptive relaxation* [Kowarschik et al. 2006; Rüde 1993], which we adapt for the specific use case of high density contrast pressure correction. Furthermore, we extend adaptive relaxation-based multigrid to function as a preconditioner for the Conjugate Gradient (CG) method, even accounting for the inherent asymmetry introduced by adaptive relaxation in the multigrid preconditioner.

These innovations collectively result in a pressure Poisson solver with runtime, even at extreme resolutions and density ratios, comparable to other simulation components like particle-grid transfers and advection —in effect, eliminating the pressure solve bottleneck.

1.1 Summary of Contributions

To summarize, our work presents four main contributions that, taken together, significantly improve the state-of-the-art in physics-based simulation of two-phase flows at very large scales:

- A hybrid Eulerian/Lagrangian method, *Phase-Field-FLIP* (PF-FLIP), for simulating highly turbulent multiphase flows at large Reynolds numbers and high fluid density contrast without requiring surface reconstruction.
- *Multiresolution Sparse Block Grids* (MSBG): A simple, efficient, and modern-hardware-friendly framework for spatially adaptive simulations.
- A dual particle-grid adaptivity scheme with a fast stochastic method for particle coarsening without neighborhood search.
- A novel, adaptive Poisson solver, specifically tailored for two-phase, high density-contrast pressure projection.

In combination, our contributions provide the building blocks for very large-scale simulations of two-phase flows. Our overall algorithm, employs adaptivity across all critical components of the pipeline—i.e., computational grid, particles, and, crucially, also in the pressure Poisson solver—ensuring that no bottlenecks with suboptimal scaling remain. As a result, the proposed solver surpasses existing state-of-the-art methods in terms of scale, efficiency, and the ability to reproduce complex two-phase flow details. We demonstrate its capabilities through a series of comparisons and large-scale multibillion-particle simulations, performed on a single workstation, without using the GPU. A public implementation of the MSBG framework is available at https://github.com/tum-pbs/MSBG.

2 Related Work

Fluid simulation has been a prominent research area in computer graphics for more than two decades [Bridson 2015; Losasso et al. 2004; Stam 2023], and we refer to recent surveys for an in-depth overview [Wang et al. 2024].

Due to the high density ratio of water to air, *free surface methods* solve the Navier-Stokes equations only for the water phase, applying curvature-based forces [Brackbill et al. 1992], or use an embedded interface via the Ghost Fluid Method (GFM) [Kang et al. 2000; Liu et al. 2000]. In Graphics, single-phase liquid simulation has reached a mature state, producing impressive results [Aanjaneya et al. 2017; Ferstl et al. 2014; Liu et al. 2016; Shao et al. 2022; Wang et al. 2024]. We employ the FLIP method [Zhu and Bridson 2005], which has seen numerous extensions [Fu et al. 2017; Jiang et al. 2015; Nabizadeh et al. 2024; Qu et al. 2022], however, typically focusing on improvements for single-phase simulations.

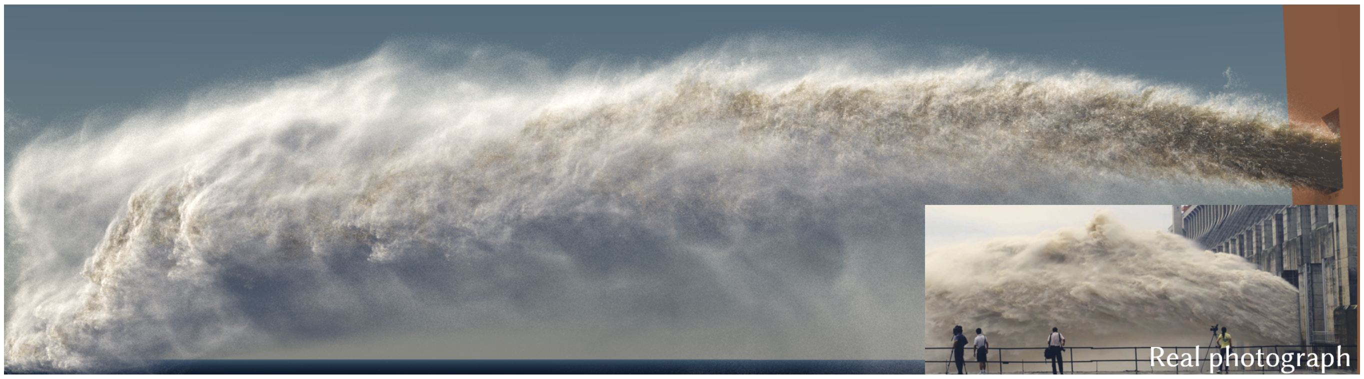


Fig. 3. Phase-Field-FLIP simulation of a large dam discharge showing violently turbulent water-air interaction induced by a powerful water jet at high speed ($\approx 40 m/s$). A real photograph of a dam discharge event at similar scale (Three Gorges Dam, China, © Picture Alliance) is included in the lower right corner. No effort was made to tune the simulations for the exact conditions of the reference, and no turbulence enhancement or post-processing effects were applied. The effective resolution is 2048×1024^2 for the pressure and 8192×4096^2 for the liquid surface and spray density, with $\approx$ 40s simulation time per time step.

To model the effects of the missing second phase, many previous works employ secondary spray particles [Ihmsen et al. 2012], or add an ad-hoc spray velocity field at reduced resolution [Kim et al. 2006]. Another heuristic, popular in the special-effects industry, is to use two single-phase solvers, and treat the liquid as a solid boundary for the air solve [Lesser et al. 2022; Stomakhin et al. 2023]. These heuristics either require extensive manual tuning, or fall short of fully capturing the large-scale fluid behavior due to the lack of a physically accurate two-phase velocity field (cf. Figure 2). Nielsen and Østerby [2013] proposed treating spray as an additional fluid phase. However, this approach does not address the coupling of the air with the bulk liquid phase.

Two-Phase Fluid Simulation. Given the importance of coupling both phases, full two-phase solvers have been extensively studied in the CFD community, with applications ranging from gas bubbles [Sirignano 2010], to sloshing in containers [Remmerswaal and Veldman 2022]. An overview of CFD approaches is given, for example, in Yeoh and Tu [2019]. One of the first two-phase solvers in Graphics was introduced by Hong and Kim [2005], employing a Particle Level Set (PLS) scheme [Enright et al. 2002], augmented by octree-based adaptivity to simulate bubbles. Song et al. [2005] proposed using the Constrained Interpolation Profile (CIP) method for low-dissipation advection, while Mihalef et al. [2008] used the Coupled Level Set Volume-of-Fluid (CLSVOF) method [Xiao 2012] for water-air interactions with animated models, later refining it with the Marker Level Set (MLS) method [Mihalef et al. 2009] to track subgrid bubbles and droplets. More closely aligned with our approach, Boyd and Bridson [2012] used FLIP for multiphase fluid simulation. In contrast to our method, this method requires a relatively costly interface reconstruction, and the inability to resolve shear layers between phases leads to a heuristic divergence model with "particle bounce-back" to impose a free-slip boundary between the phases. Without addressing scalability and density contrast-awareness of the pressure solver, they focused on small-scale scenarios dominated by surface tension with around 80^3 grid cells. The Material Point Methods (MPM) can be seen as an extension of FLIP [Zhang et al. 2017] and have been very successfully improved through spatial adaptivity and GPU-based implementations [Gao et al. 2018; Wang et al. 2020]. MPM can flexibly handle a wide range of highly complex materials. Our work instead trades in generality for increasing the scale of water-air simulations. Additionally, Lattice-Boltzmann methods have recently been popularized as an alternative [Li et al. 2020], and were likewise employed for two-phase [Li et al. 2022] and multiphase simulations [Li et al. 2024]. The underlying algorithm is very amenable to GPUs, and led to impressive results approaching real-time capability. Complementarily, our focus on a CPU-based algorithm removes the bottleneck of scarce GPU memory, and simplifies employing spatially adaptive algorithms, which is key for offline-simulation at cinematic resolutions in movie-production. Two- and multiphase flow simulation has also been explored with Smoothed Particle Hydrodynamics (SPH): Density Contrast Interfaces [Solenthaler and Pajarola 2008] handle density ratios of up to 1:100, while Ren et al. [2021, 2014] employ mixture models from CFD. Despite producing complex mixtures, the corresponding interface model necessitates small time steps, which in combination with non-adaptive particles impedes simulating large scales at very high resolutions.

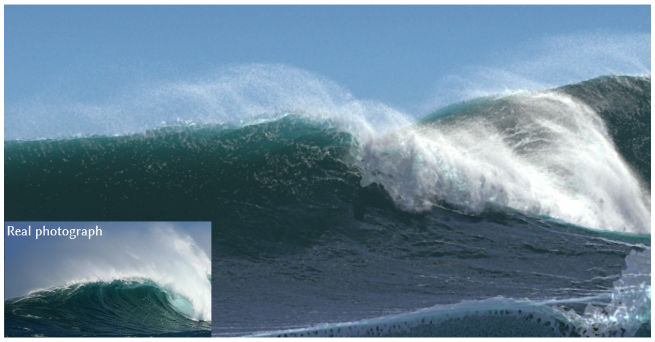


Fig. 4. Simulation of breaking ocean waves, exhibiting the characteristic streamers of mist shaped by turbulent wakes in the air behind wave crests. The fully physics-based high-resolution animation of such effects has been unattainable with existing simulation methods in graphics.

Most prior state-of-the-art work on multiphase fluid simulation in Graphics relies on regular grids or constant-size particles. This approach suffices for many small-scale scenarios, typically at resolutions of a few hundred grid cells. However, achieving cinematic realism in large-scale scenarios demands effective resolutions in the thousands, which is extremely challenging due to the computational cost of incompressible fluid simulation increasing with the *fourth* power of grid resolution (Section 4). Consequently, *spatial adaptivity*—allocating higher resolution to visually important regions—becomes indispensable.

Sparsity vs. Multiresolution Adaptivity. It is important to distinguish between two types of adaptivity in simulation methods. Many approaches [Ferstl et al. 2016; Hu et al. 2019a; Liu et al. 2016; Museth 2013; Shao et al. 2022; Wang et al. 2020; Wu et al. 2018; Liu et al. 2018] provide spatial *sparsity*, where data is stored and computed only in active regions, avoiding empty or inactive areas. In contrast, true *multiresolution* adaptivity, as employed in this work, allows different regions of the simulation domain to be represented at varying resolutions based on their complexity or "activity" as measured by a user-defined refinement criterion. This is crucial in fluid simulations, where the Navier-Stokes equations must be solved not only near the surface but also within the fluid "bulk," albeit at reduced resolution.

Adaptive grids. For single-phase fluid simulation, various adaptive techniques have been proposed. Key approaches include using octrees for hierarchical refinement of Cartesian grids [Ando and Batty 2020; Ferstl et al. 2014; Losasso et al. 2004; Popinet 2003], tetrahedral meshes [Ando et al. 2013; Chentanez et al. 2007], power diagrams [Aanjaneya et al. 2017] and memory-constrained GPU-based schemes [Raateland et al. 2022]. While tetrahedral meshes provide greater adaptivity and naturally avoid T-junctions at resolution transitions, we prioritize hexahedral Cartesian grids for their simple structure—enabling regular numerical stencils for highly parallelizable compute kernels and facilitating the construction of multilevel structures.

An increasingly important feature of adaptive schemes is the ability to preserve the hardware-friendliness of uniform grids locally, even within a globally irregular topology. In tree-based schemes, this can be achieved by assigning a uniform sub-grid (*grid block*) to each leaf node instead of a single grid cell. Building on this idea, Setaluri et al. [2014] introduced SPGrid, a block-based adaptivity technique later utilized by Aanjaneya et al. [2017] for adaptive power diagrams. While SPGrid's use of hardware page-address translation is elegant, it is not operating-system-agnostic (currently limited to Linux [Setaluri et al. 2014]) and the growing adoption of *huge memory pages* to reduce TLB overload [Navarro et al. 2002; Panwar et al. 2019] conflicts with SPGrid's reliance on fixed-size traditional 4k memory pages. Also, in virtualization and cloud environments, direct control over memory pages is often limited. SPGrid natively supports *sparse* data storage but requires multiple SPGrids and an additional octree structure to achieve multiresolution.

Finally, we highlight VDB [Museth 2013], the state-of-the-art method in the visual effects (VFX) industry to manage volumetric data, which has also been applied to liquid simulation [Shao et al. 2022]. VDB excels at handling spatial sparsity but lacks support for multiresolution adaptivity, limiting its effectiveness in scenarios requiring hierarchical refinement. Although random access is O(1), it involves traversing a shallow, fixed-depth tree, adding overhead compared to flat indexing. Additionally, like SPGrid, VDB enforces relatively small block sizes. In Section 4.2, we introduce a sparse multiresolution scheme (MSBG) that leverages grid blocking for maximal simplicity and efficiency on modern CPUs.

Adaptive particles. In our hybrid Eulerian-Lagrangian method, enabling adaptivity solely for the grid is insufficient; adaptivity for particles is also required. This is typically achieved through particle splitting and merging [Adams et al. 2007; Ando et al. 2012, 2013]. For single-phase SPH, Winchenbach et al. [2017] proposed continuously adapting particle sizes rather than relying on fixed size levels. While splitting is purely local, merging requires searching a particle's neighborhood. This is not an issue in SPH, where neighborhood information has to be gathered anyway, but for FLIP it undermines FLIP's advantage of noninteracting particles. Nevertheless, prior work on adaptively sampled FLIP particles has employed neighborhood search-based merging [Ando et al. 2012, 2013], thus not fully exploiting the efficiency of the FLIP principle.

In FLIP-based liquid simulation, Ferstl et al. [2016] introduced Narrow Band FLIP (NB-FLIP), which confines particles to a narrow band around the liquid surface, reducing computational costs away from the interface. NB-FLIP has become the industry standard for liquid simulation in VFX production and is implemented in the leading commercial software *Houdini*. However, its use of a single layer of uniformly sized particles limits adaptivity. Computationally, NB-FLIP is comparable to our scheme with only one level of particle refinement. In the context of single-phase liquid simulation, Ando et al. [2013] proposed an innovative approach to FLIP-based adaptivity using tetrahedral meshing. Besides two-phase capabilities, a difference of our work is the lightweight MSBG-based construction of the adaptive grid topology from the particle cloud while their work uses relatively expensive meshing, involving building both an octree and a BBC (Body-Centered Cubic) mesh which, as stated by the authors themselves, can become a bottleneck.

Pressure Poisson Solver. Pressure projection is often the primary computational bottleneck in incompressible fluid simulation, particularly for multiphase flows as it requires solving an elliptic PDE, whose discretization produces a large, frequently ill-conditioned linear system. This challenge has been extensively addressed in both Graphics and Computational Physics. For a comprehensive overview of multigrid pressure solvers in CG, see Shao et al. [2022]. Two-phase simulations with high fluid density ratios exacerbate this challenge due to severely ill-conditioned pressure matrices. Computational Fluid Dynamics (CFD) pursued various approaches, including geometric multigrid preconditioned Krylov methods [MacLachlan et al. 2008], algebraic multigrid [Raw 1996], operator-dependent intergrid transfers [Dendy 1982], and deflation-based methods [Tang and Vuik 2007]. MacLachlan et al. [2008] provides a detailed comparison of these techniques for two-phase flows with high density ratio.

Our work emphasizes integrating multigrid methods with spatial adaptivity. Adaptive Mesh Refinement (AMR) introduces challenges for multigrid, such as efficient coarse-grid representation and smooth error transfer across resolution boundaries. CFD research addresses this with methods tailored for patch-based [Berger and

Colella 1989; Martin et al. 2008] and octree-based [Popinet 2003; Teunissen and Ebert 2018] AMR frameworks. In Graphics, Setaluri et al. [2014] and Aanjaneya et al. [2017] developed geometric multigrid schemes using an octree-derived SPGrid hierarchy, but their focus on single-phase flows left high-density ratio challenges unaddressed. Unlike prior AMR methods, which require additional elliptic solves at resolution boundaries [Martin et al. 2008] or interlevel transfer passes for ghost cell data [Setaluri et al. 2014], our method seamlessly applies smoothing and grid transfer operators across resolution levels. Existing multigrid solvers can be very efficient, but they often still dominate simulation costs. For instance, Setaluri et al. [2014] report pressure correction consuming 90% of total simulation time in high-resolution scenarios. In Section 6, we introduce a fully adaptive Poisson solver that matches the efficiency of the other simulator components, effectively eliminating the pressure solver bottleneck.

3 Phase-Field FLIP

Our goal is to numerically solve the Navier–Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + \frac{\mathbf{f}}{\rho} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

for the 3-dimensional velocity field $\mathbf{u}$ of two *incompressible*, *immiscible* fluids characterized by their densities ρ_l, ρ_g and kinematic viscosities ν_l,ν_g, where the subscripts l and g denote liquid and gas phase with $\rho_l \gg \rho_g$. For simplicity, we use the terms 'gas' and 'air' interchangeably. p is fluid pressure, t is time and $\mathbf{f}$ denotes external forces. For two-phase flow, a *phase function* $\phi : \Omega = \Omega_l \cup \Omega_g \rightarrow [0, 1]$ distinguishes gas and liquid phases within the domain Ω. The density ρ is calculated with $\rho(\mathbf{x}) = lerp(\rho_l, \rho_g, \phi(\mathbf{x}))$, where $lerp$ denotes linear interpolation. The viscosity $\nu(\mathbf{x})$ is computed analogously.

Numerical modeling of two-phase flows presents numerous challenges due to the discontinuities at an interface of rapidly changing topology. Numerous interface-tracking methods have been proposed, e.g., with explicit representations via a mesh [Wojtan et al. 2009], and interface-capturing schemes, where the interface is implicitly defined [Mirjalili et al. 2017]. The latter are popular for their ability to effectively handle complex and dynamic topological changes, and can be divided into sharp interface methods such as volume of fluid (VOF) [Hirt and Nichols 1981; Scardovelli and Zaleski 1999], level-set (LS) schemes [Osher and Sethian 1988; Sussman et al. 1994], and diffusive interface or phase-field (PF) methods [Anderson et al. 1998; Jacqmin 1999]. Here, we use the term "phase field" in its most general sense. to distinguish it from sharp-interface approaches while some authors use the term more specifically to denote the Cahn-Hilliard free energy-based phase field approach [Jacqmin 1999].

VOF methods conserve mass but require relatively complex geometric interface reconstructions. LS methods, on the other hand, typically do not conserve mass and require iterative surface re-initialization. Phase-field (diffusive interface) methods conceptually assign a finite "thickness" to the interface, which helps avoid numerical problems associated with discontinuous interface representations. Classical, Eulerian phase-field methods are based on modeling fluid free energy density [Jacqmin 1999] and typically require solving a fourth-order PDE with regularization terms to evolve the phase field ϕ. This is often further complicated by specific precautions to ensure mass conservation [Jain 2022]. In this work, we present an alternative hybrid Eulerian-Lagrangian approach, *Phase-Field-FLIP*, which does not require any interface maintenance or reconstruction step.

3.1 Overview of FLIP and Notation

The FLIP method [Brackbill and Ruppel 1986; Zhu and Bridson 2005] is a hybrid technique that combines the advantages of particle-based (dissipation-free and mass-conserving advection) and grid-based methods (efficient discretization of spatial derivatives for pressure projection). FLIP particles are comparably inexpensive because they do not interact, i.e. no neighborhood search is necessary. The efficiency of the extremely lightweight but "blind" FLIP particles comes at the cost of additional noise. However, this is not a disadvantage for the highly turbulent, splashy scenarios we target, as these already exhibit a significant level of "natural" noise. For brevity, we will not repeat the structure of the basic FLIP- algorithm and refer to the work of Zhu and Bridson [2005] instead.

Conventions and Notation. We use subscripts p,q for particles and subscripts i,j for grid cells. The grid follows a standard staggered MAC layout [Harlow et al. 1965], where pressure and velocity are sampled at cell centers and cell face centers, respectively. The subscript a refers to a coordinate axis direction ($\mathbf{e}_a$, where $a \in \{1, 2, 3\}$ in 3D), and faces are referred to by cell index and axis. For example u_{ai} denotes the cell face velocity sampled at $\mathbf{x}_i + \frac{1}{2}\mathbf{e}_a$, i.e. at the center of the cell i plus face offset for axis a. For brevity, we often refer interchangeably to a continuous field, for example $\mathbf{u}$ and its discrete representation on the grid, for example $\mathbf{u}_{ai}$.

3.2 Particles-to-grid Transfer (P2G) in PF-FLIP

For the PF-FLIP algorithm, we assign a type to each particle to associate it either with the air or the liquid phase. At the beginning of each time step we then transfer mass

$$m_{ai} = \sum_p w_{ai,p} m_p, \tag{3}$$

and momentum

$$\mathbf{P}_{ai} = \sum_p w_{ai,p} m_p \mathbf{u}_p, \tag{4}$$

from the particles to the grid cell faces, where the particle mass

$$m_p = \begin{cases} \rho_l V_p & \text{if } p \text{ is a liquid particle,} \\ \rho_g V_p & \text{if } p \text{ is an air particle,} \end{cases} \tag{5}$$

depends on the particle type (liquid or air) which is fixed throughout the simulation, thus ensuring mass conservation. $w_{ai,p}$ are the cell face centered splatting kernel weights as defined below. In rare cases, particles may be deleted, e.g. when "stuck" in solid obstacles, but this does not result in any significant mass loss in practice. A re-seeding of particles is generally not necessary. We assume unit volume $V_p = 1$ for the particles, except when dealing with spatially adaptive particle sizes (Section 5).

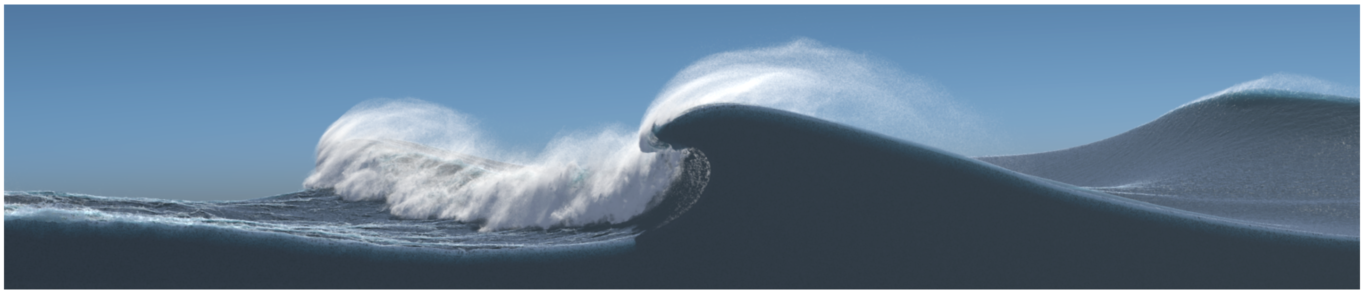


Fig. 5. PF-FLIP can resolve realistic large scale water-air interactions. The example above shows breaking ocean waves simulated with our method.

We will later reuse the accumulated splatting kernel weights for the phase field, so in order to obtain a sufficiently smooth field, we do not rely on the trilinear weighting commonly employed in FLIP. Instead, similar to SPH [Monaghan 1992], we use a higher-order polynomial kernel

$$w_{ai,p} = (\max(1 - ((||\mathbf{x}_i + \mathbf{e}_a/2 - \mathbf{x}_p||/r_p)^2, 0))^3, \tag{6}$$

with spherical support that is cubic in the squared distance, thus avoiding square root calculations. $\mathbf{x}_i$ denotes the center of cell i and r_p is the particle radius. Dividing face momentum by face mass gives the (preliminary) cell face velocities $\tilde{\mathbf{u}}^*_{ai} = \mathbf{P}_{ai}/m_{ai}$.

3.3 Obtaining the Phase Field

We obtain the phase field essentially for free as a by-product of the particle-to-grid transfer of the velocity field by simply reusing the splatting kernel weight accumulators as raw mass densities, $\tilde{\rho}$, which are already conveniently available at the cell faces $\tilde{\rho}_{ai} = m_{ai}$ —exactly where they are needed for the variable-coefficient Poisson equation in the subsequent pressure projection step.

The raw densities exhibit irregularities in higher spatial frequencies due to the FLIP-typical noise in the particle distribution (Figure 6), but we can leverage the large density ratio ($\rho_g = \frac{\rho_l}{1000}$) to easily separate phases even under noisy conditions. To this end, we normalize, compress and clamp the raw values:

$$\phi(\tilde{\rho}) = \begin{cases} 0 & \text{if } \tilde{\rho} < \tilde{\rho}_{min}, \\ \min(\sqrt{(\frac{\max(\tilde{\rho}-\tilde{\rho}_{min},0)}{\alpha_\phi \tilde{\rho}_0 \rho_l})}, 1) & \text{else.} \end{cases} \tag{7}$$

$\tilde{\rho_0}$ denotes the density obtained by applying the weighting kernel to perfectly uniformly distributed particles. The parameter α_ϕ (default value: 1) controls the trade-off between noisiness (higher values) and "stiffness" (lower values) of the phase field, similar in effect to "artificial surface tension". While not critical for solver stability, this parameters offer a potential knob for artistic control of the interface. $\tilde{\rho}_{min}$ is $\eta_\phi \rho_g \tilde{\rho}_0$, with a scaling factor $\eta_\phi = \log(\rho_l/\rho_g)$ — large enough to avoid misidentifying spurious air density peaks as liquid. We tag cells with phase field values below 0.5 as "air," while others are "liquid." We set $\phi = 1$ for liquid-liquid cell faces and $\phi = 0$ for air-air faces, eliminating all remaining phase field fluctuations away from the interface. The *interface thickness* ϵ_ϕ of our phase field is given by the particle base radius r_0 which we set equal to the grid spacing $r_0 = \Delta x$. Note that this allows for sharper interfaces than those imposed by classical phase field methods, which typically use a thickness of more than one grid cell (often employing 3 cells and more).

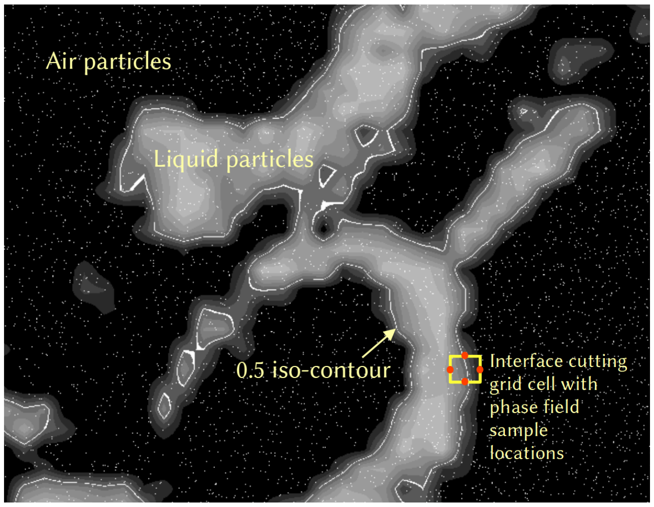

Air particles
Liquid particles
0.5 iso-contour
Interface cutting grid cell with phase field sample locations

Fig. 6. 2D slice of a phase field ϕ as computed by Eq. (7). The PF-FLIP simulation algorithm does not require surface reconstruction; the iso-contour shown here with linear interpolation is only depicted for illustration. The air phase appears uniformly black because density fluctuations have 1000× smaller amplitude than in the liquid phase.

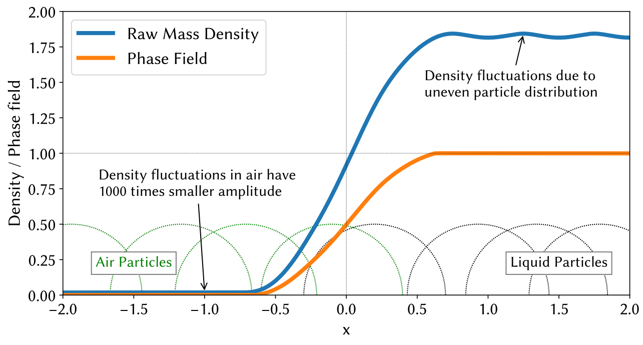


Fig. 7. 1D example of raw mass density $\tilde{\rho}$ from particles and the corresponding phase field $\phi(\tilde{\rho})$ providing a reliable separation.

The resulting phase transition is stable, and robust to particle-induced noise, as illustrated in Figure 7. Unlike classical phase-field methods, the inherently Lagrangian nature of our approach avoids numerical interface diffusion, and does not require sharpening or re-initialization steps [Arrufat et al. 2021; Jain 2022; Sussman et al. 1994]. It also naturally ensures consistent transport of mass and momentum, which is not necessarily the case for classical phase-field or level-set based simulation methods because those generally use different advection schemes for the velocity field and the interface geometry —often requiring complicated numerical countermeasures [Arrufat et al. 2021; Raessi and Pitsch 2012]. Figure 6 shows an example of a phase field as computed by Eq. (7). The high-frequency particle density fluctuations from FLIP manifest as irregularities in the phase field, but due to the high density contrast, the field achieves a robust separation of the phases.

Finally, we note that our method differs from existing "variable-density" FLIP-solvers [Bridson 2015], which are essentially single-phase simulators with continuously varying density within the phase. In contrast, PF-FLIP explicitly distinguishes the phase field from the (raw) density field, ensuring stable separation of two distinct phases even under FLIP-induced noise. The phase field $\phi \in [0, 1]$ is then used to compute the actual physical density (and viscosity) trough interpolation across an interface of finite thickness, following the principle of classical phase-field methods.

3.4 Escaped Particles and Surface Tension

Previous FLIP-based two-phase methods [Boyd and Bridson 2012] rely on "particle bounce-back" heuristics to prevent uncontrolled mixing of air and water particles. This is avoided in PF-FLIP because our velocity field adheres to the physically more accurate no-slip condition between phases. We can resolve this condition due to the high resolution capabilities of our method —thus eliminating velocity discontinuities at the interface. If particles are still found on the "wrong" side of the interface, this can be physically interpreted as subgrid-scale droplets or bubbles. Similarly to earlier work [Enright et al. 2002; Mihalef et al. 2007], we treat such *escaped particles*, p^*, at position $\mathbf{x}^*$, as candidates for conversion into droplets ($\phi(\mathbf{x}^*) < \frac{1}{2} - \delta_\phi$) or bubbles ($\phi(\mathbf{x}^*) > \frac{1}{2} + \delta_\phi$), respectively. δ_ϕ controls the rate of droplet or bubble production, with a default value of 0.2. Escaped droplets are subsequently simulated as purely Lagrangian point masses as outlined in Section 7. Bubble particles are simply deleted in our current implementation because we found that their visual contribution is negligible in our large-scale scenarios. When droplet particles merge with the liquid body again, we re-convert them to FLIP-simulation particles.

Neglecting Surface Tension. Surface tension is computationally expensive to handle in two-phase flow solvers [Popinet 2009], but it primarily affects small-scale phenomena, such as capillary waves in the millimeter range, which are not visually significant for our target use case of large-scale flows where we simply neglect it for efficiency.

3.5 Two-Phase Pressure Projection

The velocity field u^* from the P2G step after adding gravity force is generally not divergence-free. To satisfy Eq. (2) we perform a standard pressure projection step by subtracting the inverse density weighted pressure gradient, obtaining $\mathbf{u} = \mathbf{u}^* - \Delta t \frac{1}{\rho} \nabla p$.

Taking the divergence on both sides and using $\nabla \cdot \mathbf{u} = 0$ gives the variable coefficient pressure Poisson equation [Gibou et al. 2002]

$$\Delta t \nabla \cdot (\beta(\mathbf{x}) \nabla p(\mathbf{x})) = \nabla \cdot \mathbf{u}^*(\mathbf{x}) + c_{div}(\mathbf{x}), \tag{8}$$

with *face coefficients*

$$\beta(\mathbf{x}) = 1/\rho(\mathbf{x}). \tag{9}$$

Here, the dependence on $\mathbf{x}$ highlights the spatially varying coefficient structure which crucially distinguishes multiphase pressure projection from its simpler single-phase counterpart with constant density. $c_{div}(\mathbf{x})$ is a divergence correction term to ensure volume conservation by locally maintaining the target particle density $\tilde{\rho_0}$ [Losasso et al. 2008]. Discretizing Eq. (8) on the MAC grid yields a linear system of equations, in 1D for simplicity, giving

$$\beta_{i+\frac{1}{2}} \left(\frac{p_{i+1} - p_i}{\Delta x} \right) - \beta_{i-\frac{1}{2}} \left(\frac{p_i - p_{i-1}}{\Delta x} \right) = \frac{1}{\Delta t} \left(u^*_{i+\frac{1}{2}} - u^*_{i-\frac{1}{2}} \right). \tag{10}$$

Note that the coefficients β of the Laplacian matrix $\mathcal{L}$ of the resulting system $\mathcal{L}p = \frac{1}{\Delta t} \mathcal{D}(u^*)$, with $\mathcal{D}$ being a discrete divergence operator, are sampled at the cell faces. Importantly, the PF-FLIP algorithm gives these coefficients by construction, while other schemes rely on interpolation from cell-center sampled values [Xiao 2012] or require a signed distance field for ghost-fluid extrapolation [Boyd and Bridson 2012].

$\mathcal{L}$ is symmetric and positive definite, making iterative solvers such as the preconditioned conjugate gradient (CG) algorithm a natural choice. However, $\mathcal{L}$ becomes ill-conditioned in the presence of high fluid density ratios, severely slowing convergence even with multigrid solvers [MacLachlan et al. 2008]. We will address this challenge with a custom linear solver in Section 6.

3.6 Grid-to-particles Transfer

After obtaining the divergence-free grid velocity $\mathbf{u}_{ai}$, we subtract the preliminary velocity field $\tilde{\mathbf{u}}^*_{ai}$ (a copy of which was saved before adding external forces), yielding the overall velocity change during the time step

$$\Delta \mathbf{u} = \mathbf{u} - \tilde{\mathbf{u}}^*, \tag{11}$$

which can now be interpolated from the grid at the particle position and added to the velocity stored on the particles. FLIP methods usually "mix in" a small amount $(1 - \alpha_{FLIP})$, of grid-interpolated velocity u_{ai} [Bridson 2015]. Using an interpolation operator I, it introduces artificial viscosity for damping noise as follows:

$$\mathbf{u}_p{}^{\text{new}} = \alpha_{FLIP} \left(\mathbf{u}_p{}^{\text{old}} + I(\Delta \mathbf{u}, \mathbf{x}_p) \right) + (1 - \alpha_{FLIP}) I(\mathbf{u}, \mathbf{x}_p). \tag{12}$$

3.7 Viscosity and Advection

Variable, phase-specific viscosity is important in multiphase flows, as it governs velocity gradients in turbulent water-air boundary layers, such as wind over breaking waves. Bridson [2015] observed that the artificial viscosity induced by the α_{FLIP} parameter directly determines physical viscosity:

$$\nu = (1 - \alpha_{FLIP}) \frac{\Delta x^2}{6 \Delta t}. \tag{13}$$

We take advantage of Eq. (13) to solve the full multiphase Navier-Stokes equations, including spatially varying viscosity. This is possible at the same computational cost as the inviscid Euler equations, by adjusting the FLIP weight based on particle type (liquid or air) during the G2P step. While efficient, this approach is constrained to a specific range of viscosities between a minimum base viscosity needed to suppress FLIP noise and a maximum dictated by the inherent limitation of explicit schemes in handling large viscosities. However, for our case of large-scale scenarios involving water and air at high resolutions and large CFL numbers, Eq. (13) provides a good approximation.

Finally, the particles are advected through the new, divergence-free velocities $\mathbf{u}_{ai}$ using third-order Runge-Kutta integration with locally adaptive time stepping [Bridson 2015]. The liquid velocity is extrapolated into the air phase by a few cells to ensure that liquid particles are advected with the corresponding velocities.

4 Adaptivity

The computational cost of fluid simulation in 3D scales with $O(n^4)$ for grid resolution n, and multiphase flows further aggravate the scaling by the higher complexity of the pressure solve. Hence, resolving the turbulent details around the interface in natural flows necessitates computational *adaptivity*, i.e., sparse and multiresolution-capable numerical representations. In this context, block-based methods, which represent the grid as a union of uniform sub-grids, are essential for maintaining the regular data access patterns required to fully utilize the computing capabilities of modern hardware. Existing block-based methods such as VDB [Museth 2013], SPGrid [Setaluri et al. 2014] or block-octrees [Teunissen and Ebert 2018] typically keep the block size small ($n = 4^3$ or $n = 8^3$ grid cells) to minimize the waste of unused cells within blocks. Indeed, if we assume that fluid interfaces are smooth 2D manifolds embedded in 3D space, then the "unused cell overhead" for blocks of size n is $1 - O(n^2)/n^3$, which quickly approaches 100% when block size increases, as shown in Figure 8 (a).

4.1 Block Size Does Matter

We make several key observations that, in contrast to existing work, motivate using significantly larger blocks ($n = 16$, $n = 32$).

Fractal interfaces. Visually interesting fluid interfaces in nature are fractals of dimension $d \approx 2.5$ [Sreenivasan 1991], not smooth 2D manifolds. For these 2.5D interfaces, the empty-cell overhead $(1 - O(n^{2.5})/n^3\ )$ is significantly lower than for 2D interfaces, as shown by the difference in curves (a) and (b) in Figure 8.

Block boundary overhead. In 3D, the fraction of boundary cells per total cells $(n^3 - (n-2)^3)/n^3 = 1 - (1 - \frac{2}{n})^3$ decreases rapidly with increasing block size (curve c in Figure 8). This is important as stencil-based operations that dominate simulation time (i.e. multigrid smoothing) require costly operations at block boundaries.

Hardware efficiency. Modern hardware, with larger CPU caches, wider SIMD lanes, and larger virtual memory pages, benefits from larger blocks. Unlike a decade ago, blocks of 16^3 or 32^3 now fit easily in L2 or L3 CPU cache, enabling more efficient hardware utilization than smaller blocks.

Reduced block count. Naturally, larger blocks drastically reduce the *number* of blocks for a given simulation domain size.

The last aspect is particularly important: a small number of blocks makes it feasible to use dense linear arrays as a global block index. The curve (d) in Figure 8 shows the memory overhead $1/(0.01n^3)$ of a dense index for a sparsely populated domain with 1% occupancy and a single 4-byte data "channel". Notably, for blocks of size 16^3, the dense index requires only 2% of the total memory usage (and this relative overhead decreases further with each additional data channel). Naturally, compared to tree-based and other more sophisticated alternatives, dense linear indices are attractive due to their unmatched simplicity and hardware friendliness.

Besides mere sparsity, another key aspect of adaptivity is handling spatially *varying resolution.* A common approach in block-based spatial structures is to recursively decompose grid blocks into smaller subblocks, maintaining a constant number of cells per block. This is typically implemented using tree-based decompositions in high-performance computing [Dubey et al. 2014; Olson and MacNeice 2005; Teunissen and Ebert 2018; Zhang et al. 2021]. Despite their wide-spread use, tree-based methods introduce algorithmic complexity and overhead. Tree depth affects random access times, and efficient parallelization of operations is challenging [Sundar et al. 2008]. Stencil-based operations are further complicated by the non-uniform block neighborhood structure.

4.2 MSBG

In light of the preceding considerations, we propose a novel data structure based on a dense linear array with large blocks. The dense linear array stores pointers to blocks, whose geometric size is kept constant while varying only the *resolution* of each block. Note that this is in contrast to conventional tree-based schemes that adapt the blocks' size while keeping the resolution constant. In this context, the distinction between (geometrical) block size and block resolution is important. The former is chosen once for each simulation, while the latter is spatially adapted during the course of a simulation. For brevity, we will often call finely resolved blocks *large*, in contrast to *small* blocks with a coarse resolution.

We denote the resulting data structures as *Multiresolution Sparse Block Grids* (MSBG) in the following. They enable multi-resolution adaptivity in an extremely simple way, and allow for fast, massively parallel global operations. The resolution of the blocks is varied with a *block refinement map* M, the calculation of which is driven by an application-specified refinement criterion (Section 5.4).

While varying the resolution of fixed-size blocks limits the number of refinement levels to the binary logarithm of the base resolution, this is not a constraint in practice: in three dimensions, computational cost decreases by a factor of eight with each coarser resolution level. In this study, all results are based on a maximum of three or four refinement levels. Unless noted otherwise, the base block resolution is set to 16. The block boundary overhead (curve (c) in Figure 8) increases at coarser MSBG resolution levels. However, this limitation is offset by the fact that most grid cells are still contained in MSBG blocks at the finest resolution levels, due to the factor-of-eight reduction in the number of active cells with each coarser level.

The discretization of differential operators for gradients, divergence, and the treatment of solid obstacles follows previous work

Table 1. Comparison between MSBG and Block-Octree

	MSBG	Block-Octree
Construction, e.g. from particles	$O(n)$, Easy parallelization	$O(n \log n)$, non-trivial parallelization
Random access	$O(1)$	$O(log(n))$
Stencil access	Uniform access to 6 neighbor blocks	Non-uniform access to variable number of up to 24 neighbor blocks
Grading	Fast	Parallelization challenging
Adaptivity	Only restricted by block raster	Less flexible because "terminal siblings" have same resolution
Supports multi-color techniques	Yes	No
Code complexity	Low	High
Refinement depth	$\log_2$(base-resolution)	Unlimited

[Losasso et al. 2004; Weber et al. 2015], with a straightforward adaptation to MSBG. An example of pressure gradients is shown in Figure 9(c), where the constant $\Delta_g = \frac{3}{4}$ [Losasso et al. 2004].

4.3 Capabilities of MSBG

Next, we highlight a range of advantages that result from the design of the MSBG data structure. Table 1 provides an overview of MSBG compared to block-based octrees representing the state of the art in adaptive grid methods [Teunissen and Ebert 2018]. Its uniformly constant blocks size makes MSBG computationally more efficient and improves adaptivity by capturing refinement areas more tightly.

Multi-resolution Multi-color schemes. One particularly distinctive feature of MSBG is its ability to support *Multiresolution multi-color schemes*, such as block-based red-black multigrid smoothers and fast lock-free 8-color particle splatting. Such algorithms are not supported by, e.g., octrees due to their nonuniform block sizes. Specifically, we employ an adaption of a recently proposed eight-color splatting scheme [Fang et al. 2018] to achieve lock-free splatting during the P2G step of PF-FLIP from Section 3.2, enabling maximum parallelism of *multiresolution* particle-grid transfer. We also compute the splatting weights for all faces of a MAC grid cell in parallel using SIMD/AVX-vector instructions. In practice, this yields P2G transfer rates of several hundred million particles per second.

Spatiotemporally coherent memory access. The contiguous blocks of MSBG ensure within-block cache efficiency, but we further enhance spatio-temporal memory coherence by conceptually ordering active blocks along a space-filling curve via Morton coding. Unlike SPGrid [Setaluri et al. 2014], which integrates Morton coding into cell address calculations (requiring potentially expensive bit interleaving), we retain simple linear indexing and sort active block indices by their Morton codes before traversal. This requires computing the Morton code only once per block, taking advantage of the additional level of indirection offered by our dense array of block pointers. Block-based sorting, like other operations on global grid topology, is extremely fast in MSBG due to the use of relatively few but large blocks.

Dynamic task based scheduling. At the same time, a naive parallelization of operations on a MSBG grid can lead to load balancing issues due to the non-uniform number of grid cells per block. To address this, we adopt *dynamic task based* scheduling [Robison et al. 2008], as provided by the TBB multithreading library [Pheatt 2008].

5 Adaptive Particles and Dual Adaptivity

As PF-FLIP employs particles in addition to the underlying grid, it is crucial to support particle-based adaptivity alongside the grid-based adaptivity provided by MSBG. In the context of SPH methods, adaptivity is typically achieved through particle splitting (coarse-to-fine transition) and merging (fine-to-coarse transition) [Adams et al. 2007]. We adopt a similar approach for particle splitting; however, instead of merging, we introduce a fast probabilistic coarsening technique. We assign a *particle level* l_p to each particle p that determines its radius as $r_p = r_0 2^{l_p}$. Note that we consider the particle level l_p orthogonal to the *grid level* $l_i = M[b_i]$, of a given grid cell within a block b_i. Here, M denotes the block refinement map of the MSBG grid.

5.1 Probabilistic Coarsening

Existing adaptive particle methods handle the fine-to-coarse transition by merging several smaller particles into larger ones [Adams et al. 2007; Ando et al. 2013], a process that requires a particle neighborhood search. While this is not an issue for SPH methods, where neighborhood lists have to be collected anyway, we aim to retain the essence of FLIP as a neighborhood-search-free method.

To this end, we propose a simple and fast probabilistic coarsening scheme. It is based on the observation that coarsening a uniform particle distribution by a factor of two in each spatial dimension results in a uniform distribution with a particle density eight times lower, which can be obtained by randomly removing 7/8 of the particles. More formally, once a particle is marked for coarsening, we remove it with probability $prob_{\text{del}} = 1 - 1/(2^{l_{\text{new}} - l_{\text{old}}})^3$. Oscillations near the coarse-to-fine transition zone are prevented by a simple hysteresis scheme using a deferred-coarsening counter.

5.2 Refinement Criteria

As the default particle refinement criterion, we employ the commonly used distance to the liquid–air interface. Since PF-FLIP does not maintain a distance field for the surface, we generate a reduced-resolution auxiliary SDF $\tilde{d}(\mathbf{x})$ from the downsampled and thresholded phase field with a simple, first-order accurate, marching algorithm. This is sufficient because, unlike previous work, we use the distance field solely for particle refinement.

Optionally, the magnitude of the curl of the velocity can also be employed [Popinet 2003] to capture flow details away from the surface, particularly in the air phase, where turbulence is typically more intense than in the bulk liquid (Figure 12, left column). However, for the examples presented in this study, we found the distance-based refinement to be sufficient for all cases.

5.3 Equalizing Spatially Varying Numerical Viscosity

Numerical methods for solving the Navier-Stokes equations introduce numerical diffusion, or artificial viscosity, depending on the

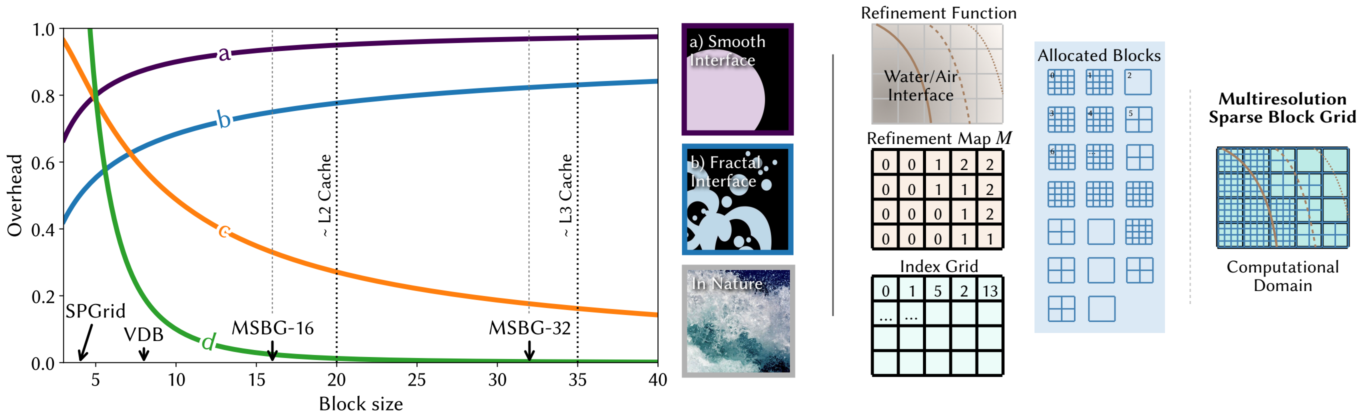


Fig. 8. (Left) Theoretical overhead of adaptive data structures as a function of block size: (a),(b) Wasted cells per block near the interface for smooth and fractal interfaces respectively. (c) Block boundary overhead. (d) Storage overhead of a dense block index assuming a float channel with 1% occupancy. (Right) Overview of the MSBG data structure with refinement levels stored in the per-block refinement map, the linear block index and individual blocks constituting the final MSBG data structure.

grid spacing Δx. In multiresolution methods with varying Δx, this leads to unphysical spatial variations in viscosity—an issue often overlooked in the adaptive mesh refinement literature. A notable exception is Ando et al. [2013], and we likewise leverage FLIP's controllable numerical viscosity (Eq. (13)) to adjust α_{FLIP}. In our two-phase framework, we account for both phase-specific physical viscosity and numerical viscosity variations from adaptive grid spacing. In contrast to Ando et al., we base corrections on local grid spacing instead of particle size. This decouples particle radius from cell size, which is crucial for large time steps with high CFL numbers, where particles may cross grid resolution boundaries.

5.4 Dual Particle-Grid Adaptivity

The resulting algorithm for coupled particle-grid adaptivity proceeds with the following steps:

(1) Compute the *block refinement map* M: For each particle p, determine all blocks B_i that intersect the particle's bounding box (enlarged by a CFL-dependent epsilon) and set $M[i] \leftarrow \min(l_p, M[i])$ using a fast atomic minimum operation.
(2) Regularize the block refinement map to enforce a 2:1 grading constraint, ensuring that the resolution levels of neighboring blocks differ by no more than one.
(3) Initialize the MSBG grid with the new refinement map and allocate grid data channels.
(4) Transfer data from the particles to the MSBG grid as follows: For each particle p, determine its containing grid block b_i, grid level $l = M[b_i]$ and cell-within-block c_{ik}. Using the linear, dense global MSBG block index, this computation requires only shift and logical-and operations on the particle coordinates. Next, loop over all neighboring grid blocks b_j and cells c_{jl} within the particle's radius $r_p = r_0 2^{l_p}$. This corresponds to the usual 27-cell neighborhood when the particle level matches the grid level, i.e., $l_p = l$. We also allow $l_p = l + 1$ for a smooth transition of coarse particles into regions where the grid is finer by one level. At each touched cell, splat the particle's weighted mass and velocity contributions to the staggered cell locations.
(5) On the MSBG grid, compute auxiliary fields, such as $\tilde{d}$, required for evaluating the particle refinement criterion.
(6) For each particle p, use the refinement criterion interpolated at its position $\mathbf{x}_p$ to determine the new particle level, e.g. $l_{\text{new}} = \lfloor|\beta_{bw}\tilde{d}(\mathbf{x}_p)|/m_0\rfloor$ where m_0 is the MSBG base block resolution and β_{bw} controls the width of refinement bands. If $l_{\text{new}} < l_p$, split the particle into 8 smaller particles p', randomly jittered within the octants of the grid cell containing p, and set $l_{p'} \leftarrow l_{\text{new}}$. If $l_{\text{new}} > l_p$, increment a *deferred-coarsening* counter D_p and mark p for coarsening if $D_p \geq 3$. Reset $D_p = 0$ if $l_{\text{new}} \leq l_p$. If p is marked for coarsening, remove the particle with probability $prob_{del} = 1 - 1/(2^{l_{\text{new}}-l_p})^3$.
(7) Advance the PF-FLIP simulation one time step and advect the particles to their new positions using MSBG for adaptive-grid velocity interpolation.

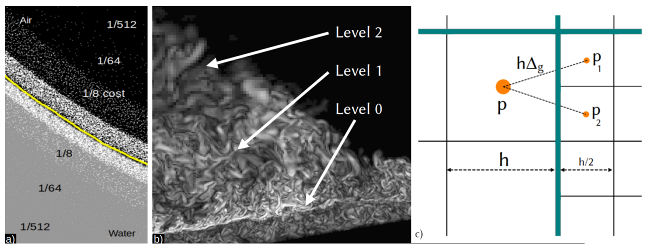


Fig. 9. a) Particle density and size are adapted depending on the distance to the fluid interface. b) Velocity field as sampled by particles with grid cell size adapted to particle size (2D slice of 3D simulation, vorticity magnitude). c) Pressure gradient discretization at coarse-fine transition: $\frac{p_1 - p}{h\Delta_g}$. Thin and bold lines depict cell and MSBG block boundaries, respectively.

6 Adaptive Poisson Solver

Pressure correction is typically the computational bottleneck in incompressible fluid simulation, particularly for high density-contrast

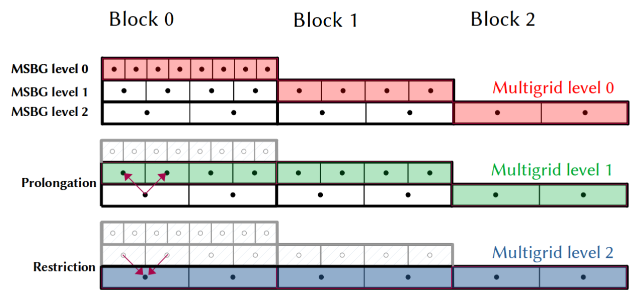


Fig. 10. Multiresolution multigrid: MSBG maintains *multigrid levels* independently from *MSBG resolution levels*

multiphase flows, as this step requires solving a large and highly ill-conditioned Poisson problem $\mathcal{L}x = b$ (cf. Section 3.5). Our multiresolution discretization scheme ensures favorable properties for the Laplacian system matrix $\mathcal{L}$, such as symmetry and a simple 7-point stencil. However, for multiphase flows with high density ratios, the ill-conditioned, nearly discontinuous coefficient structure of $\mathcal{L}$ leads to a deterioration in convergence speed, even when using powerful methods such as multigrid-preconditioned CG [MacLachlan et al. 2008]. To address the aforementioned challenges, we propose a novel, fully adaptive variant of the multigrid preconditioned conjugate gradient method, specifically designed to solve pressure correction equations arising from highly irregular two-phase flows with large fluid density contrasts.

6.1 Algebraic Aggregation Multigrid

Shao et al. [2022] successfully employed algebraic aggregation multigrid, essentially a cell-centered Galerkin coarsening approach, to construct a multigrid hierarchy for fast but fixed-resolution single-phase fluid simulations. We extend this method to multiresolution adaptive grids by introducing a novel adaptive smoother and integrating it into a specialized conjugate gradient algorithm for efficient two-phase pressure projection. In the following, we will not reiterate the full details of the standard multigrid method, such as the structure of a basic V-cycle, as these are well documented in the literature (e.g., [Briggs et al. 2000; Mandel 1988; Trottenberg et al. 2000]). Instead, we focus on the aspects that differentiate our method from previous approaches. In Galerkin coarsening, grid operators are computed algebraically from the fine grid ones via

$$\mathcal{L}^{2h} = R_h^{2h} \mathcal{L}^h P_{2h}^h, \tag{14}$$

where h is the spacing of the finer grid. P and $R = P^T$ denote *Restriction* and *Prolongation* operators, respectively [Trottenberg et al. 2000]. See Figure 10, bottom left, for a 1D illustration.

Multiresolution adaptive Galerkin coarsening. Algorithm 1 details the efficient, matrix-free, and massively parallel construction of our multiresolution-adaptive Galerkin multigrid hierarchy, based on the MSBG block structure. For each cell j of block i at multigrid level $L > 0$, three face-coefficients $\mathcal{F}[i, L][j, a]$ of the symmetric 7-point stencil are computed ($a \in \{1, 2, 3\}$ denoting face normal directions $\mathbf{e}_a$). Face coefficients at the finest level are obtained directly from the phase-field via. Eq. (9). The diagonal elements of the (implicitly) assembled matrices are the sum of the off-diagonal (face) coefficients. The resolution transition scaling constants $\sigma_{CF} = 1/(4\Delta_g)$ and $\sigma_{FC} = 2\sigma_{CF}$ follow from the symmetry-preserving multiresolution discretization of the pressure gradient (Section 4.2, Figure 9(c)) and the Galerkin coarsening principle Eq. (14).

Algorithm 1: Parallel Multiresolution Galerkin Multigrid Coarsening

```
Input: Face coefficients $\mathcal{F}$ at finest level $L = 0$ Output: Face
  coefficients for all coarse levels
for $L \leftarrow 1$ to $M - 1$    ▹ Multigrid levels
do
    for $i \leftarrow 1$ to $N$    ▹MSBG Blocks in parallel
    do in parallel
        $l \leftarrow level(i, L)$    ▹Actual resolution level at this multigrid level
        if $l = level(i, L - 1)$ then
            $\mathcal{F}[i, L] \leftarrow \mathcal{F}[i, L - 1]$;
                ▹No finer child block, simply copy block by reference
            continue
        for $a \leftarrow 1$ to 3    ▹Face normal directions (x,y,z)
        do
            ▹Determine scaling factor for resolution transition
            switch $l - level(neighborBlock(i, a), L - 1)$ do
                case 0 do
                    $s \leftarrow 2^l$    ▹Fine-fine
                case 1 do
                    $s \leftarrow 2^l \sigma_{CF}$    ▹Coarse-fine
                case -1 do
                    $s \leftarrow 2^l \sigma_{FC}$    ▹Fine-coarse
            $n \leftarrow n_0 2^{-l}$    ▹Block resolution
            for $j \leftarrow 1$ to $n^3$    ▹Loop over all cells in block
            do
                $sum \leftarrow 0$ ;
                ▹Loop over 2x2 fine cells of coarse cell face in direction a
                for $\mathbf{k} \leftarrow (1, 1)$ to $(2, 2)$ do
                    $j' \leftarrow \mathrm{cellIndexInFineBlock}(j, \mathbf{k}, a)$
                    $sum \leftarrow sum + \mathcal{F}[i, L - 1][j', a]$
                $\mathcal{F}[i, L][j, a] \leftarrow s\frac{sum}{4}$
```

6.2 Leveraging MSBG

We leverage the MSBG block structure to efficiently represent two resolution hierarchies: the MSBG hierarchy, defined by the *resolution level* l, and the multigrid hierarchy, defined by the *multigrid level* L. For the coarsest levels, which are coarser than MSBG's block index, we use additional, uniform grids. Figure 10 illustrates our extension of basic multigrid construction from uniform to spatially adaptive multiresolution grids using a 1D example. While our multigrid hierarchy shares similarities with the multiresolution pyramids proposed by Setaluri et al. [2014], there are key distinctions. First, MSBG does not require maintaining an additional octree, as in SPGrid-based solvers. Second, the inherent multiresolution structure of MSBG facilitates multigrid relaxation across resolution

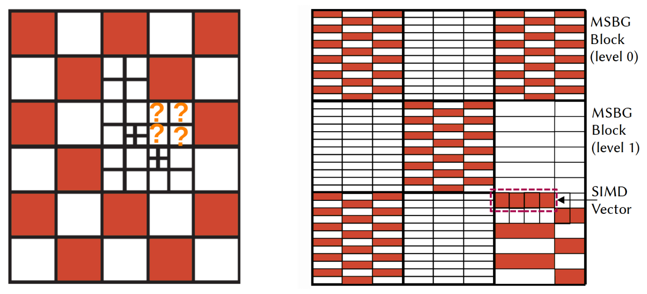


Fig. 11. (a) Multi-color techniques such as red-black (checkerboard) cannot be efficiently applied to octrees (as indicated by the "?") (b) Active elements during the first half of one parallel half-sweep ($I_{rb} = i_{rb} = 0$ in Algorithm 2) of our SIMD accelerated, two-stage MSBG block based hybrid red-black-Gauss-Seidel-Jacobi smoother. Shown are blocks of SIMD vectors at two different resolution levels 0 and 1.

boundaries, eliminating the need for additional inter-level transfer passes to accumulate and propagate ghost cells.

6.3 Block-based Relaxation

The *smoother* is a critical component of multigrid schemes. Commonly, a fixed number (1–3) of weighted Jacobi or Gauss-Seidel relaxation steps are applied before residual restriction (pre-smoothing) and after error prolongation (post-smoothing) [Briggs et al. 2000]. For massively parallel implementations, Jacobi-type smoothers are often preferred due to the parallel nature of updates. However, this comes at the cost of slower convergence compared to Gauss-Seidel, where updated values are immediately visible to neighboring cells. Moreover, Jacobi-type smoothers require additional memory for a temporary destination grid.

Exploiting the regular block structure of MSBG, we propose a variant of a *hybrid Gauss-Seidel* smoother [Baker et al. 2011] that employs red-black ordered, parallel half-passes of Jacobi relaxations at the block-level. It additionally uses Gauss-Seidel (in-place) iterations for the unknowns *within* blocks, likewise with a red-black ordering. These in-place updates are computed at the level of entire SIMD-words, that is, 8 floats for the Intel AVX CPU instruction set. This scheme, illustrated in Figure 11 (b), improves convergence speed due to the *partial Gauss-Seidel* nature of the updates. It also reduces the memory footprint by a factor of two compared to standard Jacobi, as red-black-ordered MSBG blocks can be updated in place without requiring an additional destination buffer. Similar to the P2G transfer, this block-"colored" algorithm inherently relies on the globally uniform block structure of MSBG and is not applicable to tree-based data structures (Figure 11(a)).

Due to the partial Jacobi nature of our hybrid smoother, it is necessary to apply a weighting factor ω [Adams et al. 2003] to dampen the relaxation updates in order to guarantee overall convergence. Here, we chose $\omega = \frac{6}{7}$, which is optimal in three spatial dimensions [Trottenberg et al. 2000].

6.4 Adaptive Relaxation

The smoother described in the previous section in combination with Galerkin coarsening significantly improves overall multigrid convergence. To address the remaining convergence issues of highly irregular flows, we leverage ideas from *fully adaptive multigrid* [Kowarschik et al. 2006; Rüde 1993] to perform adaptive relaxation. The central idea is to adaptively focus the smoothing effort locally on regions with high residual errors. To this end, we maintain a dynamic list of *active* MSBG blocks. Initially, this list includes all relaxation-relevant blocks b_i for a given multigrid level L—those with $l(i, L) \leq L$, along with a one-block-wide transition zone at the next coarser level. Blocks are iteratively selected from the active list —a fast lockless FIFO queue—for relaxation. If a block's maximum *effective residual error* exceeds a threshold θ after relaxation, it is reinserted into the list. Neighboring blocks are also added if border cell residuals exceed the threshold. Unlike Kowarschik et al. [2006]; Rüde [1993], we do not continue relaxing until the active set becomes empty. Instead, we terminate the loop after a fixed number (10–20) of adaptive relaxation steps, as this is sufficient for our purpose of employing the algorithm within a preconditioner rather than as a standalone multigrid solver.

We again leverage the MSBG block structure to keep track of and process active blocks efficiently and in parallel. The procedure is detailed in Algorithm 2, and Figure 12 illustrates how this process rapidly reduces the active smoothing area, significantly lowering the total computational effort. Overall efficiency is further enhanced because, after a few iterations, the set of remaining active blocks often fits entirely into the CPU's last-level cache, eliminating the RAM bandwidth bottleneck (typically associated with relaxation-type small-stencil sweeps) during later iterations.

To further enhance cache efficiency, we perform multiple relaxation sweeps on a block once its active data channels are loaded into cache memory. This approach amortizes the cost of transferring data from RAM to the CPU cache and assembling the block's local "halo buffer", which contains boundary data from neighboring blocks. Unlike previous methods [McAdams et al. 2010], we do not fix the number of block-local sweeps. Instead, we adaptively determine it by monitoring the block-local residual, available as the absolute value of the local update during relaxation. The loop terminates when the residual is reduced by 1/3, approximating the optimal reduction factor for a Gauss-Seidel smoother in 3D Poisson problems [Mohr and Wienands 2004]. Importantly, this exact value, like other parameters in Algorithm 2, is not critical for overall convergence, as our adaptive smoother functions solely within a preconditioner.

Defining an Effective Residual in the Context of High-Contrast Phase-Fields. Our algorithm also differs from Kowarschik et al. [2006] in how the effective residual error is defined. While their work uses the *scaled residual*—the residual divided by the matrix diagonal— we found this approach to be ill-suited for solving the linear systems arising in two-phase fluid flow. In such cases, the matrix diagonal is orders of magnitude larger in the air phase than in the water phase of the simulation domain. This disparity effectively compresses the dynamic range of the residual error in the air phase by a factor of 1000, making it unlikely for air blocks to ever exceed the error threshold for inclusion in the active relaxation list. We therefore

Algorithm 2: Parallel Two-Stage-Red-Black Adaptive Multigrid Relaxation

```
Input: Multigrid level $L$, effective residual threshold $\theta$
$n, n' \leftarrow 0$    ▷ Number of active blocks (current, new)
$S_0, S, S' \leftarrow \{0\}$    ▷ Block status flags arrays
Function RelaxBlock($i, R$):
    for $i_{rb} \leftarrow 0$ to 1    ▷ Inner red-black scheme
    do
        for each cell $j$ of block $i$ where color($j$) = $i_{rb}$ do
            $\delta \leftarrow \omega(b_j - \mathcal{L}_j\mathbf{u})/\mathcal{L}_{jj}$    ▷ Weighted Gauss-Seidel; $\omega = \frac{6}{7}$
            $R[j] \leftarrow \delta$
            $u_j \leftarrow u_j + \delta$
    return max($R$)
Procedure ActivateBlock($i, R$):
    if AtomicCompareExchange($S'[i]$, 1, 0) = 0 then
        $B'$[AtomicIncrement($n'$)] $\leftarrow i$
    ▷ Activate affected neighbor blocks
    for each face $f_{k \in [1,6]}$ of block $i$ do
        $i_2 \leftarrow$ NeighborBlock($i, k$)
        if $S_0[i_2]$ and $\max_{j \in f_k} R[j] > \theta$ then
            if AtomicCompareExchange($S'[i_2]$, 1, 0) = 0 then
                $B'$[AtomicIncrement($n'$)] $\leftarrow i2$ ;
for each block $i$ do in parallel
    if $level(i, L) \leq L$ or ((IsResolutionBoundaryBlock(i) and
      $level(i, L) = L + 1$)) then
        $B$[AtomicIncrement($n$)] $\leftarrow i$    ▷ Initial list of active blocks
        $S_0[i] \leftarrow S[i] \leftarrow 1$
for $i_{iter} \leftarrow 1$ to $N_{iter,max}$ do
    if $n$=0 then break;
    SortByMortonCode($B$)    ▷ Improve coherence of memory access
    for $I_{rb} \leftarrow 0$ to 1    ▷ Outer red-black scheme
    do
        for $j \leftarrow 1$ to $n$ do in parallel
            $i \leftarrow B[j]$
            $x_b, y_b, z_b \leftarrow$ BlockCoords($i$)
            if $(x_b + y_b + z_b)$ mod 2 = $I_{rb}$ then continue;
            $r0 \leftarrow$ RelaxBlock($i, R$);  $k \leftarrow 1$    ▷ R is a local buffer
            repeat
                $r, R \leftarrow$ RelaxBlock($i, R$) ;  $k \leftarrow k + 1$
            until $r/r0 < \frac{1}{3}$ or $k \geq k_{max}$;
            if $r > \theta$ then ActivateBlock($i, R$) ;
    for $j \leftarrow 1$ to $n$ do $S[B[j]] \leftarrow \mathbf{0}$
    swap($n, n'$);  swapRef($B, B'$);  swapRef($S, S'$)
```

propose using an alternative metric for the effective residual, namely the maximum change over all variables of a block after the local block relaxations, multiplied by the diagonal—effectively "contrast-equalizing" the phases.

Adaptive error threshold. Previous work on adaptive relaxation proposed using a fixed user-defined residual error threshold θ but provided no guidance on selecting an appropriate value. Relying on a prescribed, fixed θ is problematic in our case, as we use adaptive relaxation-based multigrid as a preconditioner for CG, where the dynamic range of the effective residual error may vary widely (often decreasing exponentially) with each CG iteration. We therefore take a different approach and propose determining θ *adaptively* by computing the log-histogram of the initial effective residual and applying fast histogram segmentation. In practice, we found that a simple choice of setting θ to the mean plus two standard deviations of the log-histogram works well. Using MSBG, histogram construction can be performed very efficiently at the block level, using only the per-block maxima of the residuals.

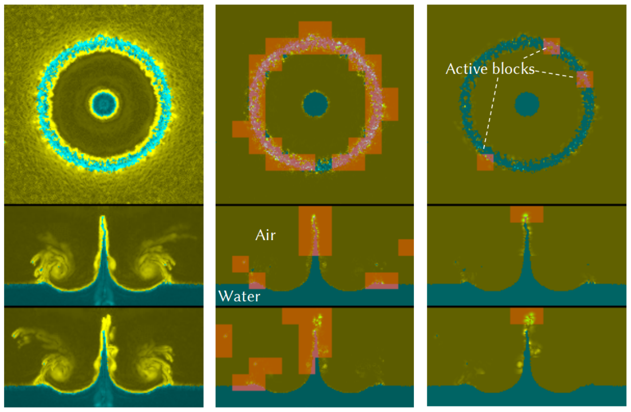


Fig. 12. Adaptive multigrid relaxation. 2D slices along XY, XZ and YZ of a 3D simulation are shown (blue: water, yellow: air, orange: active set). Left column: Vorticity magnitude. Middle: Residual error and active relaxation blocks after 2 smoothing steps (~10% of all blocks). Right: Active blocks after 6 steps (~1% of all blocks).

Flexible Preconditioned Conjugate Gradient Method. One reason why adaptive relaxation has not been considered in the context of multigrid-preconditioned CG probably lies in the fact that CG requires the preconditioner to preserve symmetry. This constraint necessitates performing exactly the same number of relaxation sweeps during pre- and post-smoothing [McAdams et al. 2010; Tatebe 1993], a requirement that is inherently violated by adaptive relaxation.

To resolve this issue, we employ the *flexible preconditioned conjugate gradient method* [Bouwmeester et al. 2015]. It has been shown to be robust even when the preconditioner is not symmetric positive definite (SPD), and is implemented efficiently by replacing the standard Fletcher–Reeves update step with the more robust Polak–Ribière variant. While it requires storing an additional vector, this overhead is offset in our overall algorithm thanks to the block-based red-black scheme (Section 6.3), which eliminates the need for a temporary destination buffer during parallel iterations.

In combination, our contributions to improved adaptivity on the Poisson solver side, namely,

- Multiresolution Galerkin-multigrid
- Adaptive relaxation with improved residual metric and adaptive error threshold
- Adaptive multigrid as a preconditioner for CG
- A memory-efficient parallel multiresolution smoother

resolved the pressure-solver bottleneck in all our examples, regardless of simulation resolution or flow complexity.

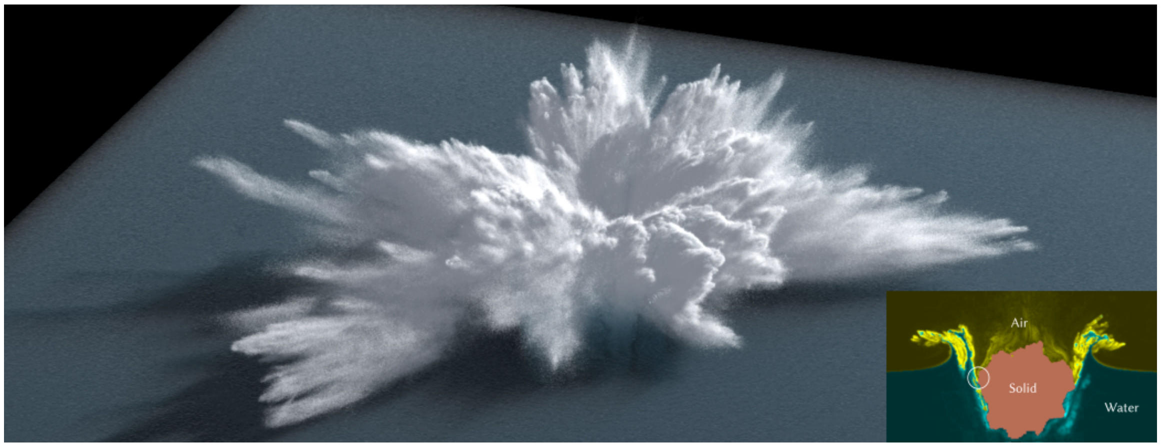


Fig. 13. Asteroid impact with 600 million particles simulated on a laptop. Effective resolution: 1024^3 for pressure, and 3072^3 for spray. Avg. simulation time per time step: $\approx$ 70s. Bottom right: 2D-slice of the phase field shaded by velocity curl magnitude.

7 Spray and Rendering

To simulate spray effects, we largely follow previous work: escaped FLIP particles (Section 3.4) are converted into purely Lagrangian droplet particles, which move under the influence of gravity and drag forces induced by the air velocity field [Gissler et al. 2017; Song et al. 2005]. We integrate the corresponding ODE using RK2 with locally adaptive time stepping. Additional spray particles are emitted near the interface in areas of steep velocity gradients [Ihmsen et al. 2012]. For increased physical realism, we employ per-particle droplet radii, r_p, which are sampled from a log-normal distribution [Babinsky and Sojka 2002] on droplet emission, subsequently evolved according to a physics-based evaporation ODE [Sirignano 2010], $dr_p/dt \sim -1/r$.

We found it crucial for realism to simulate and render spray particles at a significantly higher resolution than that provided by the base pressure solver. To achieve this, we employ a *spray density field* (at 16 bit float precision) with a resolution two to four times higher than the pressure and velocity fields. Spray particles are simulated at the increased resolution of this field and rasterized into it to yield a volumetric density for rendering. Here, we leverage MSBG to efficiently handle sparse grids with up to tens of thousands of voxels per dimension (Section 8.1).

We used a (MSBG based) volumetric ray tracer with approximate multiple scattering to render the spray density field and the water. The rendered surface is reconstructed from the liquid particles using standard particle skinning and surface smoothing [Bhatacharya et al. 2011]. Note, that spray particle rasterization and water surface reconstruction are only necessary at render time, i.e. every 5-10 time steps. High-curvature areas of the air-water interface were rendered with foam-like scattering coefficients. Apart from this, no post-processing effects or procedural textures are applied.

8 Results

We first evaluate the performance of the key components of our method, the MSBG framework for spatial adaptivity and the adaptive Poisson solver, in comparison to state-of-the-art methods. We then perform comparisons to reference simulations from Graphics and CFD, and highlight the capabilities of PF-FLIP on MSBGs with a range of large-scale simulation examples at very high grid resolutions.

Unless stated otherwise, all tests and simulations were conducted on a single AMD workstation (Ryzen Threadripper) with 32 cores and 256 GB of RAM. To demonstrate the method's performance on less powerful hardware, we also include an example (Figure 13) executed on a laptop (Intel Core i7, 14 cores, 32 GB of RAM). We emphasize that our solver does not rely on GPU acceleration, which is important for our target use case of very large-scale scenarios whose memory requirements can exceed the limited RAM capacities (compared to CPUs) of even high-end GPUs.

8.1 Adaptive Grids with MSBG

Performance across representative operations. In order to evaluate the performance of MSBG, we conducted tests across representative operations that typically determine the performance of many physics-based simulation schemes: particle-to-grid transfer (P2G), stencil-based iterative evolution of a partial differential equation (PDE) and interpolation of grid quantities at point positions (G2P).

In these tests, 1.6 billion particles were rasterized (P2G) as spheres with a radius of 3 grid cells on a sparse grid with an effective resolution of $10,000^2 \times 5,000$ cells, resulting in 10.6 billion active voxels, which were subsequently filtered using ten iterations of Laplacian diffusion (PDE). An illustration of this setup is shown in Figure 14. Next, one billion point queries were performed with linear interpolation of the filtered grid values (G2P). To ensure realistic test conditions, the particles were taken from a snapshot of a complex, high-resolution simulation (the asteroid impact scenario shown in Figure 13). Table 2 presents the results in comparison to the VFX industry standard VDB [Museth 2013], which serves as the state-of-the-art baseline (we used OpenVDB version 11). Compared to VDB, MSBG achieves performance improvements by a factor of 3.7 to 9.6. At the same time, the memory footprint increases only moderately by 11%, despite the MSBG blocks in this example being twice as large (eight times the 3D volume) as the VDB blocks (16^3 vs. 8^3).

This provides strong evidence supporting the theoretical considerations presented in Section 4.1, which justify our decision to employ significantly larger blocks than those used in prior methods.

Table 2. MSBG Performance across operations representative of physical simulations in graphics: particle- to-grid transfer (P2G), stencil-based iterative evolution of a PDE, and interpolation of grid quantities at point positions (G2P). Performance is measured in units of million particles per second (P2G, G2P) or million grid cells per second (PDE). The test scenario is shown in Figure 14.

Test	VDB	MSBG	Improvement
P2G (Mparts/s) ↑	16.7	62.6	× **3.7**
PDE (Mcells/s) ↑	948	9097	× **9.6**
G2P (Mparts/s) ↑	634	4051	× **6.4**
Memory usage (GB) ↓	37.3	41.4	× **0.9**

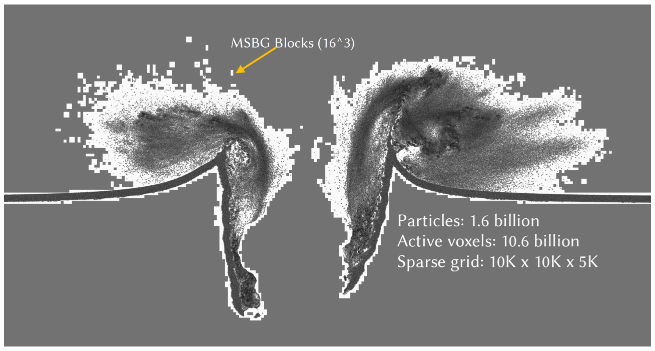


Fig. 14. The test scenario for the results reported in Table 2 involves rasterizing 1.6 billion particles as spheres with a radius of 3 grid cells on a sparse grid with an effective resolution of $10,000^2 \times 5,000$, resulting in 10.6 billion active voxels, subsequently filtered by Laplacian diffusion.

Memory bandwidth utilization. A key requirement for any spatial adaptivity method is that the efficiency gained through adaptivity is not negated by the administrative overhead of managing internal data structures. In this context, a spatial adaptivity scheme can be considered optimal if operations on an adaptive grid can be performed at nearly the same speed per *active* cell as on a uniform grid. The performance of uniform grids, in turn, is mostly determined by RAM bandwidth due to relatively low arithmetic intensity (measured in FLOPS per byte transferred) of typical computational kernels (PDE). Consequently, the effective RAM transfer rate (expressed as a percentage of the peak machine bandwidth) for a given kernel serves as an effective metric for evaluating the efficiency of adaptivity schemes.

Table 3 compares MSBG to two prominent state-of-the-art methods, VDB and SPGrid, using the aforementioned metric applied to a seven-point stencil Laplacian kernel, which is representative of many computations in graphics. A simple streaming copy serves as the baseline proxy for peak machine bandwidth. The SPGrid result was derived from Table 1 in [Aanjaneya et al. 2017]. As shown, SPGrid is approximately four times more efficient than VDB but cannot fully utilize the available bandwidth. In contrast, MSBG achieves performance close to the peak machine bandwidth.

Table 3. Memory bandwidth utilization of a seven-point stencil Laplacian kernel in percent peak machine bandwidth vs. streaming copy baseline. MSBG nearly saturates machine bandwidth.

	Streaming copy	Laplacian kernel		
Method		VDB	SPGrid	MSBG
Bandwidth utilization	100%	11%	39%	**90%**

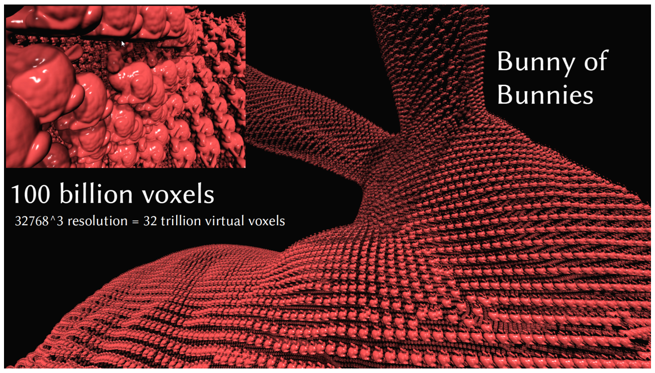


Fig. 15. A $32k^3$ MSBG grid with 100 billion active voxels as a narrow band surface reconstruction from $32k$ bunnies each consisting of the $32k$ point samples of the Stanford dataset. Effectively, a mean curvature flow PDE was evolved on the MSBG grid at a speed of 10 billion unknowns per second per iteration.

100-billion-voxels. As a final example to showcase the potential of MSBG we took the $32k$ point samples of the original Stanford bunny dataset and rasterized them as a cloud of $32k$ small bunnies, giving $32k \times 32k = 1B$ particles in total, onto a $32k^3$ MSBG grid (base block resolution 32) as a narrow-band signed distance field, yielding approximately 100 billion active out of 35 trillion virtual voxels. Then a smooth surface was reconstructed from the particles through ten iterations of a mean curvature flow PDE on the MSBG grid. To fit 100 billion voxels into main memory, we used a single MSBG data channel with half-precision and exploited MSBGs block-multicolor-based PDE-stencil-operators for solving the mean curvature flow–a second order PDE with wide stencil–in parallel via an eight-color scheme without the need for an additional temporary data channel. The surface, shown in Figure 15, was rendered at 4K resolution in ≈ 30 seconds using a MSBG-based ray tracer.

8.2 Basic Phase-Field Tests

In order to evaluate the basic accuracy and physical soundness of Phase-Field FLIP in comparison with results from established two-phase flow literature, we conducted two classical benchmark tests.

Table 4. Relative phase field error of the shear flow advection test.

Model	Relative error
[De Rosis and Enan 2021]	0.0490
[Li et al. 2022]	0.0500
Ours	0.0314

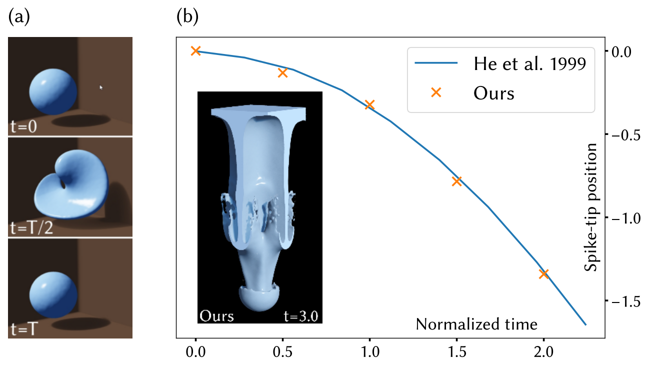


Fig. 16. (a) Shear flow advection test, (b) Rayleigh-Taylor Instability at high density ratio (A = 0.9).

Advection of phase field in prescribed velocity field. In this test, we simulated the passive advection of a spherical shape in a prescribed, time-dependent velocity field that induces a typical shear flow—commonly used in previous studies [De Rosis and Enan 2021; Fakhari et al. 2019; Li et al. 2022]. For a detailed description of the test setup, we refer to the work of Li et al. [2022]. Figure 16 (a) shows renderings of the test sphere at the start of the simulation ($t = 0$), at the point of maximum deformation ($t = T/2$), and at the end of the test following reverse advection ($t = T$). Table 4 presents the resulting relative error, computed as $(\sum_i (\phi(x_i, 0) - \phi(x_i, T))^2/\phi^2(x_i, 0))^{1/2}$, and compares it with previously published results. As expected, our hybrid Lagrangian-Eulerian advection method yields lower error than existing purely Eulerian phase field transport schemes.

3D Rayleigh-Taylor instability. In this test, we adopted the same setup as described by He et al. [1999], featuring a simulation of the Rayleigh-Taylor instability with a high density contrast. The Atwood number is $A = (\rho_l - \rho_g)/(\rho_l + \rho_g) = 0.9$, and surface tension is neglected. The grid resolution is $256 \times 64 \times 64$. We set the domain length to $L = 0.05$, viscosity to $\nu_l = \nu_g = 4 \times 10^{-5}$, gravitational acceleration to $g = 9.81$ (all in SI units) and simulation parameters $\alpha_\phi = 0.5$ and $CFL = 0.5$. Figure 16 (b) shows the development of the spike-tip position, which is in good agreement with the reference. In later stages of the simulation, because of the absence of surface tension, thin sheets of the interface begin to break up into ligaments and droplets. This is not observed in the corresponding simulation by He et al. [1999], highlighting the ability of our relatively sharp ($\epsilon_\phi = \Delta x$) phase field to resolve fine interface structures near the Nyquist limit of the grid.

8.3 Adaptive Poisson Solver

To evaluate the performance of our pressure solver against state-of-the-art methods, we simulated a highly challenging scenario—the asteroid impact—using both prior methods and our approach. This scenario features complex dynamic fluid-solid interactions, high density contrast two-phase fluid-fluid interface geometry, and intense turbulence, as illustrated in Figure 13.

For this test, the effective grid resolution was set to 1536^3, with approximately 1.6 billion particles. The comparison was evaluated based on the number of Preconditioned Conjugate Gradient (PCG) iterations and the total CPU time required per time step, averaged over 10 steps during the most challenging phase of the simulation in terms of relative speeds between the air, water and solid phases. The results, shown in Figure 17, highlight significant differences among the tested methods: Incomplete Cholesky [Bridson 2015], Geometric Multigrid (e.g., [Setaluri et al. 2014]), Galerkin Multigrid [Shao et al. 2022] with additional boundary relaxations [McAdams et al. 2010], and our fully adaptive multigrid method. The results are presented on a logarithmic scale on the y-axis to clearly illustrate the wide range of results across the methods.

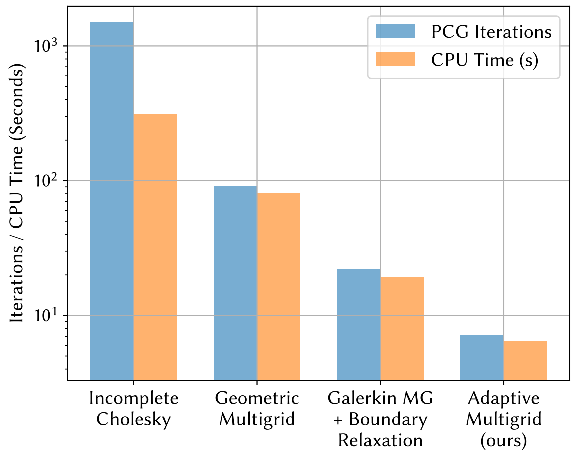


Fig. 17. Number of preconditioned CG iterations and CPU time (logarithmic y-axis) per time step for the pressure Poisson solve in the asteroid impact scenario. Our adaptive multigrid scheme outperforms the closest competitor from previous work roughly by a factor of three.

Incomplete Cholesky, as a baseline, exhibited the highest iteration count, exceeding 10^3. In contrast, Geometric Multigrid significantly reduced iteration count and CPU time, achieving approximately an order of magnitude improvement over Incomplete Cholesky. Galerkin Multigrid [Shao et al. 2022], when combined with additional boundary relaxations [McAdams et al. 2010], further reduced computational cost by about a factor of four. Finally, our fully adaptive multigrid solver proved to be the most efficient approach, improving iteration count and CPU time roughly by an additional factor of three.

Comparison With High-Performance CFD-Solvers. Next, we present a comparison with CFD solutions to demonstrate both the physical soundness of our overall method and the performance potential of our adaptive pressure solver. We re-simulated the "difficult scenario" from the work of MacLachlan et al. [2008] which evaluated fast pressure solvers for high-density-contrast incompressible two-phase flow by simulating rising and disintegrating bubbles. This benchmark was designed to be particularly challenging because of its deliberate exclusion of surface tension, resulting in highly complex and dynamic interface geometries. It is therefore also well suited for our target of large-scale simulations, where the influence of surface tension is likewise negligible. The residual error tolerance was intentionally set to a low value of $\epsilon_{tol} = 10^{-8}$ for this benchmark (in contrast to the 10^{-4} used for other examples).

Table 5. Comparison with CFD pressure solvers for a two-phase flow benchmark. CG Iterations per time step for BoxMG and DIC as reported in Figure 7 a) in MacLachlan et al. [2008] during the "difficult" phase of the simulation.

Method	**Iterations per step**	**Stencil size**	**Pre-processing cost per step**	**Adaptive mesh refinement**
Ours	≈ 30	4	Low	Native/MSBG
BoxMG-CG	≈ 40	4/14	High	Difficult
DIC-CG	> 300	4	Medium	Medium

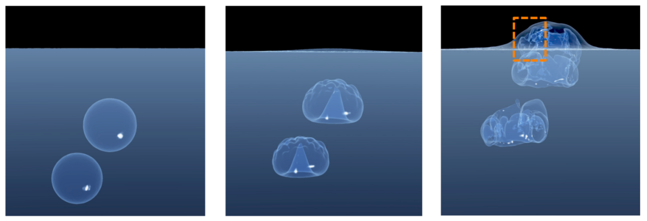
Table 5 compares our method with the two best performing methods in the study of MacLachlan et al.: Deflation IC (DIC) and Box-Multigrid (BoxMG). Our solver requires fewer iterations per time step than the best performing method (BoxMG) in the aforementioned study, while also incurring lower computational cost per iteration due to its smaller, 4-point stencil. In contrast, BoxMG requires 14-point stencils on all but the finest resolution levels. Another advantage of our approach is its faster per-step preprocessing. Specifically, our method requires only a simple sweep over the multigrid hierarchy to calculate three coefficients per cell (see Algorithm 1), whereas BoxMG involves more expensive sweeps with nested loops to compute 14 coefficients for a symmetric 27-point stencil. Most importantly, our method can be naturally extended to a multiresolution adaptive scheme, as demonstrated in the present work using MSBG. In contrast, extending Box-MG to an adaptive framework would be inherently more difficult and less efficient due to its 27-point stencil structure in 3D.

Table 6. Timing breakdown of an average simulation step of our method for the scenario shown in Figure 25, compared to the SPGrid-based solver from Setaluri et al. [2014] for their highest resolution example and a recent two-phase method in CFD [Zeng et al. 2022] for a dam-break simulation. Our method successfully avoids the pressure bottleneck of other methods which spend a disproportionate amount of time solving the Poisson equation. (A dash indicates not available / applicable.)

Method	P2G	Pressure	Advect	Other
Ours	27%	**30%**	13%	30%
Setaluri et al.	-	91%	5%	4%
Zeng et al.	-	83%	-	17%

To demonstrate the full potential of our method, we reran the same benchmark scenario at a resolution ten times higher than the highest reported in MacLachlan's original study, that is, 2000^3 compared to 200^3 at approximately the same number of iterations per time step. Figure 18 shows a zoom-in of the complex interface dynamics of a fragmenting air bubble.

Eliminating the Pressure-Solver Bottleneck. For prior state-of-the-art methods, pressure correction often accounts for the vast majority of total simulation time, particularly at high grid resolutions and large phase density contrasts. In contrast, in all our examples, the pressure solve accounted for no more than about one-third of the total runtime, regardless of grid resolution or flow complexity (cf. Table 6).



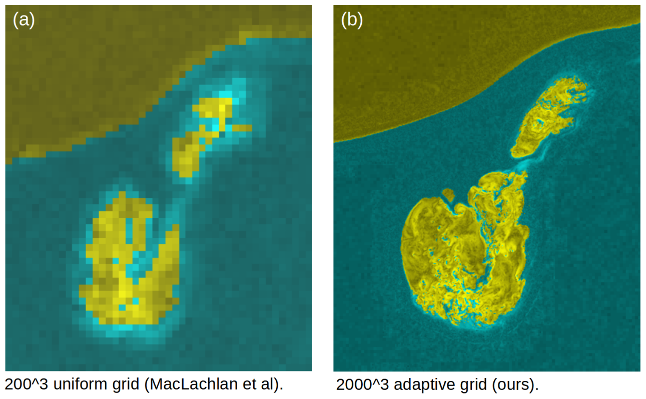

(a)
(b)
200^3 uniform grid (MacLachlan et al).
2000^3 adaptive grid (ours).

Fig. 18. Top row: Rising and disintegrating bubbles as of the CFD-benchmark from [MacLachlan et al. 2008] ("difficult scenario"). Bottom row: 2D slices (zoomed-in) at $t = 0.0266$ seconds, showing the phase field (blue=water, yellow=air) and the vorticity magnitude of the velocity field. (a) Grid resolution of 200^3 as used by MacLachlan et al. (our re-simulation) (b) Our adaptive simulation at ten times the original resolution.

8.4 Full Simulation Results

We simulated a variety of examples to demonstrate the effectiveness of our method in animating complex, large-scale two-phase flow scenarios at very high, cinematic resolutions. Where possible, we also included real photographs for qualitative comparison. We did not, however, make an effort to precisely match the conditions of the photographs. For an impression of the animated versions we also strongly encourage viewing the accompanying video.

Simulation times were typically on the order of 30 to 100 seconds per time step with 2–7 time steps per frame at CFL numbers of 3–4 (Table 7). Viscosity-parameters α_{FLIP} were set to 0.985 and 0.96 for liquid and air, respectively. Resolutions are expressed as *effective resolution*, referring to the equivalent resolution of a *uniform* grid that would be required to resolve the phase interface at the resolution of the finest MSBG level. All simulations are physics-based, without vorticity confinement or any other artificial turbulence enhancement.

Dam break with obstacle. First, we set up a classic dam break scenario similar to [Li et al. 2022, 2024], where we enlarged the relative width of the simulation domain to provide more room for splashing and water-air interaction. The effective resolution was also increased from 800×400^2 (Li et al. [2024]) to 1600^3. Small-scale (3 meters) and large-scale (400 meters) simulations were conducted with both a standard single-phase free-surface and our proposed two-phase solver. Figure 19 shows snapshots from the resulting

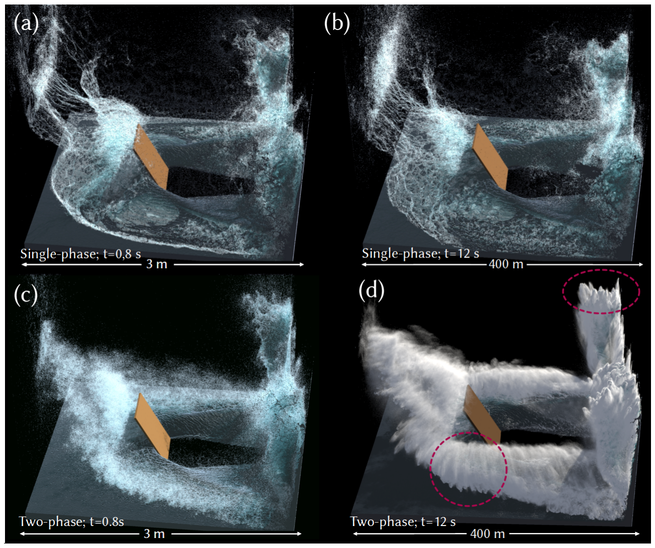


Fig. 19. Dam break scenario. Small-scale (left column) vs. Large-scale (right column) simulations. Single-phase simulations are almost identical regardless of scale, while our two-phase method can reproduce characteristic features of violent large-scale flows (d), such as "explosive" jets of spray.

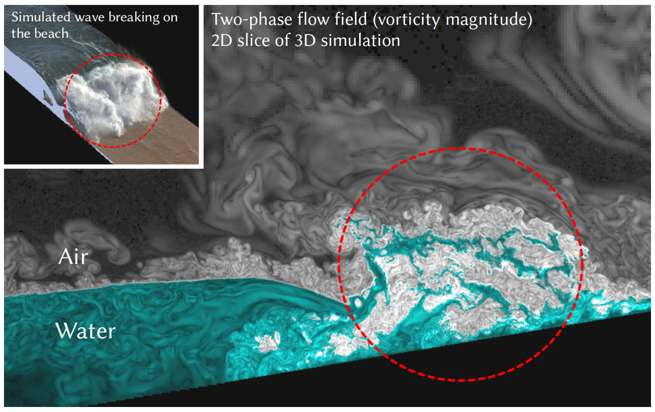

Simulated wave breaking on the beach
Two-phase flow field (vorticity magnitude)
2D slice of 3D simulation
Air
Water

Fig. 20. Fully simulating the surrounding air velocity field is crucial for visually convincing large-scale water simulations: An ocean wave breaking on the beach was simulated with our proposed method. Shown is the two-phase velocity field (vorticity). Upper left corner: The same simulation rendered in 3D. Effective resolution: $4096 \times 1024 \times 512$.

simulations. Single-phase simulations essentially look the same regardless of scale (except for time scaling, i.e. large scale looks like small scale in slow-motion) because of missing non-linear effects of the air phase interactions. In contrast, our two-phase simulation in (d) conveys the sense of a large-scale dam break. It faithfully reproduces characteristic large-scale flow features such as explosion-like jets and streamers of spray formed by vortex tubes of the turbulent air velocity (encircled in purple).

Numerical Wave Tank and Two-Phase Turbulence. In the next experiment, we generated large solitary ocean waves in a numerical wave tank driven by a standard piston-type wavemaker [Wang et al. 2019]. The resulting *two-phase velocity field*, shown in Figures 20 and 22 (top), highlights the capability of our method to resolve the liquid-air boundary layer and to reproduce realistic two-phase turbulence phenomena in the vicinity of a complex, highly dynamic liquid-air interface. This is crucial for realistic detail of complex spray movement (encircled areas in Figure 20), and, to the best of our knowledge, has not been demonstrated before in Graphics at high resolutions without procedural heuristics.

Wave and Obstacle at Extreme Resolution. To test the scalability of our method up to the limits of machine resources (256 GB of memory), we further increased the adaptive grid resolution to $6k \times 2k \times 1.6k$ for the pressure (and $18k \times 6k \times 5k$ for the spray density field) with more than 3 billion particles. A solid obstacle was also placed in the middle of the wave tank. A snapshots of the resulting simulation is shown in Figure 1. The average simulation time was approximately 2 minutes per time step with a total duration of 3.5 days.

Dam Discharge. For this experiment, we modeled the discharge flow of a large hydrodynamic dam, inspired by the Three Gorges Dam in Sandouping, China. Water flows at a velocity of $40\,\mathrm{m\,s^{-1}}$ through a floodgate with a diameter of 5 m, inclined at an angle of 18 degrees to the horizontal. The floodgate is located at a height of 9 m above a reservoir pool with an initial water depth of 3 m. Figure 3 highlights the intense two-phase water-air turbulence caused by the resulting water jet, ejected at very high speed through the floodgate. The total size of the simulation domain is $L = 200\,\mathrm{m}$, resolved at a resolution of 2048 cells for the pressure and 8192 cells for the rendered water surface and spray density fields, respectively.

Tsunami Wave Hitting a Complex Coast Line. The next example demonstrates the robustness of PF-FLIP in handling complex solid geometry alongside an intricate liquid-air interface. In this scenario, we simulated a tsunami wave impacting a procedurally generated fractal coast, which exhibits geometric detail across all spatial frequencies. The initial wave speed is $v_0 = 30\,\mathrm{m\,s^{-1}}$. Effective Resolution is 4096×1024^2. Figure 24 shows a snapshot of the simulation, with the included reference photograph highlighting the high level of physical realism achieved by our method in reproducing the characteristic spray clouds driven by the turbulent air phase of the two-phase velocity field.

Ocean Wave-Train Breaking on the Beach. Finally, we present a simulation of a large, kilometer-sized ocean scene with multiple wave trains shoaling and breaking on an artificial beach. Here, in order to create more complex and natural wave patterns, we inclined the wave generator to let waves interact with their own reflections from the domain boundary. Figure 21 illustrates the setup. The effective resolution was set to $3072 \times 1536 \times 1024$, with approximately 2 billion particles. Each time step required, on average, 60 seconds of CPU time. The resulting simulation snapshots, shown in Figures 4, 5, 25, and 23, demonstrate the ability of our solver to faithfully capture key visual features of large-scale water-air flows. These features include wind-blown mist streamers over wave crests, the characteristic "explosion-like" jets and spray plumes of highly turbulent air-water interactions associated with plunging wave lips, and the characteristic "wave tubes." Owing to the solver's physics-based nature this complex behavior is generated from simple simulation setups without requiring procedural detail enhancements, post-processing heuristics, or artistic intervention.

Table 7. Statistics for our large scale simulations. The number of active pressure cells is number of FLIP particles divided by 8.

Figure	Effective resolution Pressure	Spray-density	Max particles FLIP/Spray	Min. per Step/frame
1	6144×2048×1600	18432×6144×4800	3.1/1.0 B	1.9/9.5
3	2048×1024^2	8192×4096^2	1.1/0.5 B	0.7/4.0
4,5,25	3072×1536×1024	12288×6144×4096	2.0/0.1 B	1.0/5.7
22	4096×1024×512	12288×3072×1536	1.0/0.2 B	0.6/6.1
24	4096×1024^2	12288×3072^2	1.2/0.5 B	0.8/7.0

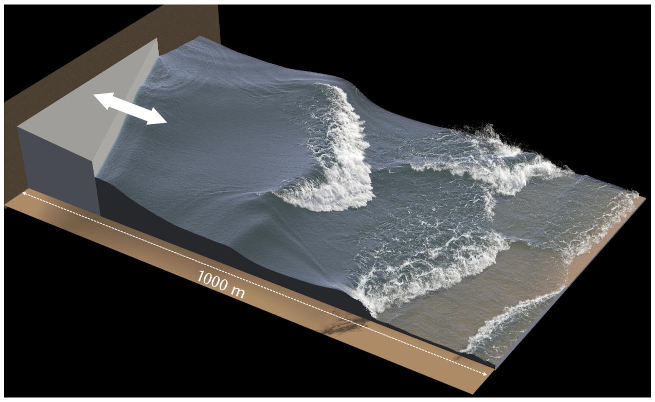


Fig. 21. Simulation setup with inclined wave generator.

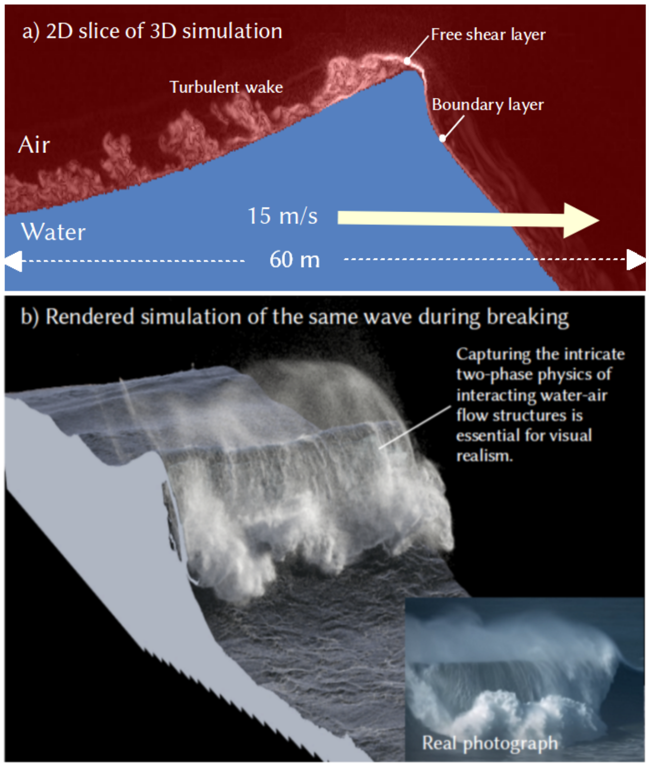

a) 2D slice of 3D simulation
Free shear layer
Turbulent wake
Boundary layer
Air
Water
15 m/s
60 m
b) Rendered simulation of the same wave during breaking
Capturing the intricate two-phase physics of interacting water-air flow structures is essential for visual realism.
Real photograph

Fig. 22. Large ocean wave. Adaptive PF-FLIP is capable of resolving the air-water boundary layer, flow separation, free shear layer and the turbulent wake behind the wave crest in 3D. (a) Shown is part of a 2D slice (vorticity magnitude) from a 3D simulation with an effective resolution of 4096 × 1024 × 512 cells and a total simulation domain size of 500 meters. (b) 3D rendering of the same wave during breaking.

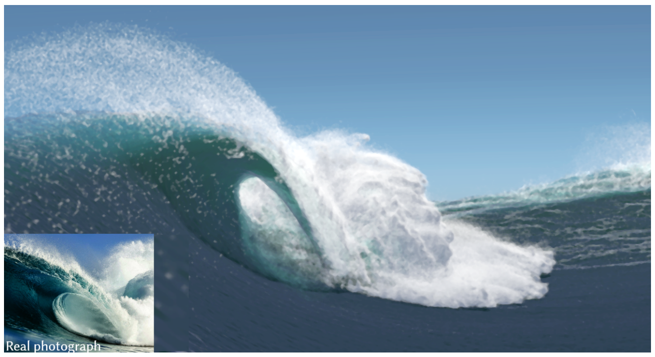


Fig. 23. Breaking wave. Turbulent wave tube. Detail zoom-in of the large-scale simulation shown in Figure 25

9 Limitations

While our method demonstrates significant improvements and capabilities, it is not without limitations. One inherent constraint, shared with most Cartesian grid-based adaptivity schemes, is that resolution boundaries currently cannot cross the liquid-air interface. We note that, in principle, it is possible to lift this restriction [Ando and Batty 2020] and we plan such an extension in future work. Furthermore, the maximum depth of refinement is restricted to the binary logarithm of the MSBG base block resolution, that is, $\log_2(16)$ or $\log_2(32)$. We have not found this limiting in practice, as in 3D each level of refinement reduces computational costs significantly, by a factor of eight.

Finally, as with other FLIP-based methods, our approach can exhibit relatively high levels of noise in field quantities arising from high-frequency noise in the particle distribution. While this is not critical for our large-scale, highly turbulent target scenarios, which inherently exhibit a high level of "natural" noise, it could impede the effectiveness of our method for very small-scale, surface-tension dominated scenarios, especially at low density ratios.

10 Conclusions and Outlook

We have introduced Adaptive Phase-Field-FLIP, a method that combines a novel FLIP variant for two-phase flow, an efficient dual multiresolution scheme for grids and particles, and a highly efficient adaptive Poisson solver. This approach addresses the key challenges of large-scale, high-resolution, and highly turbulent two-phase fluid simulation with large density ratios. By leveraging the proposed MSBG (Multiresolution Sparse Block Grids) framework and integrating adaptivity across all critical simulation components, including the Poisson solver, our method achieves unprecedented efficiency

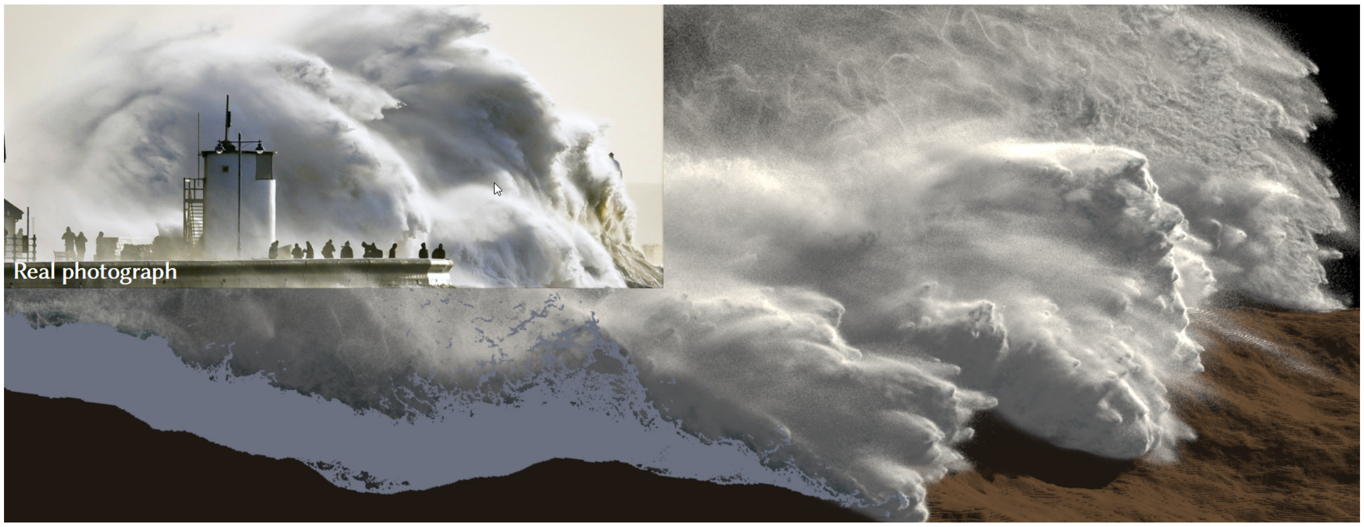


Fig. 24. Large waves hitting the coast at high speed. Comparison between our simulation and a real photograph for qualitative reference. Photograph © AP Photo / Ben Birchall.

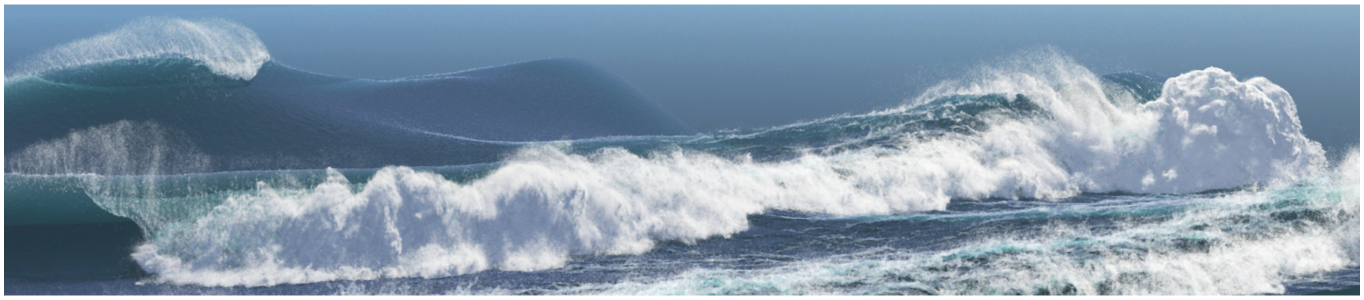


Fig. 25. Breaking ocean waves simulated with our method (2 billion particles). Our method faithfully captures key visual features of large-scale air-water flows, such as wind-blown mist streamers over wave crests and the characteristic 'explosion-like' jets and spray plumes of highly turbulent air-water interactions.

and scalability. This enables the fully physics-based simulation of intricate two-phase phenomena like turbulent air wakes of breaking waves, at cinematic resolutions on a single work station.

Looking ahead, a number of interesting directions exist for future exploration. As discussed in Section 9, we aim to extend spatial adaptivity to cases where resolution boundaries intersect the liquid surface. Another interesting line of research enabled by our work would be the integration of simulated turbulent wind fields which could open up the possibility to simulate realistic storm scenarios with ocean waves driven by physical wind dynamics. In future work we would also like to improve our spray model, possibly by incorporating droplet atomization models from the field of CFD. Finally, we see potential for employing adaptive PF-FLIP in the context of CFD applications, such as large-scale numerical wave tanks or the study of wave-structure interactions in coastal and offshore engineering.

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 Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. 2007. Adaptively sampled particle fluids. *ACM Trans. Graph.* 26, 3 (July 2007), 48–es.

Mark Adams, Marian Brezina, Jonathan Hu, and Ray Tuminaro. 2003. Parallel multigrid smoothing: polynomial versus Gauss–Seidel. *J. Comput. Phys.* 188, 2 (2003), 593–610.

Daniel M Anderson, Geoffrey B McFadden, and Adam A Wheeler. 1998. Diffuse-interface methods in fluid mechanics. *Annual review of fluid mechanics* 30, 1 (1998), 139–165.

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

Ryoichi Ando, Nils Thurey, and Reiji Tsuruno. 2012. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE transactions on visualization and computer graphics* 18, 8 (2012), 1202–1214.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

T Arrufat, M Crialesi-Esposito, D Fuster, Y Ling, L Malan, S Pal, R Scardovelli, G Tryggvason, and S Zaleski. 2021. A mass-momentum consistent, volume-of-fluid method for incompressible flow on staggered grids. *Computers & Fluids* 215 (2021), 104785.

E Babinsky and PE Sojka. 2002. Modeling drop size distributions. *Progress in energy and combustion science* 28, 4 (2002), 303–329.

Allison H Baker, Robert D Falgout, Tzanio V Kolev, and Ulrike Meier Yang. 2011. Multigrid smoothers for ultraparallel computing. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2864–2887.

Marsha J Berger and Phillip Colella. 1989. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics* 82, 1 (1989), 64–84.

Haimasree Bhatacharya, Yue Gao, and Adam Bargteil. 2011. A level-set method for skinning animated particle data. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vancouver, British Columbia, Canada) *(SCA '11)*. Association for Computing Machinery, New York, NY, USA, 17–24.

Henricus Bouwmeester, Andrew Dougherty, and Andrew V Knyazev. 2015. Nonsymmetric preconditioning for conjugate gradient and steepest descent methods. *Procedia Computer Science* 51 (2015), 276–285.

Landon Boyd and Robert Bridson. 2012. MultiFLIP for energetic two-phase fluid simulation. *ACM Transactions on Graphics (TOG)* 31, 2 (2012), 1–12.

Jeremiah U Brackbill, Douglas B Kothe, and Charles Zemach. 1992. A continuum method for modeling surface tension. *Journal of computational physics* 100, 2 (1992), 335–354.

Jeremiah U Brackbill and Hans M Ruppel. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational physics* 65, 2 (1986), 314–343.

Robert Bridson. 2015. *Fluid simulation for computer graphics.* AK Peters/CRC Press, Boca Raton, FL.

William L Briggs, Van Emden Henson, and Steve F McCormick. 2000. *A multigrid tutorial.* SIAM, Philadelphia, PA.

Nuttapong Chentanez, Bryan E Feldman, François Labelle, James F O'Brien, and Jonathan R Shewchuk. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Eurographics Association, Goslar, DEU, 219–228.

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

JE Dendy. 1982. Black box multigrid. *J. Comput. Phys.* 48, 3 (1982), 366–386.

Anshu Dubey, Ann Almgren, John Bell, Martin Berzins, Steve Brandt, Greg Bryan, Phillip Colella, Daniel Graves, Michael Lijewski, Frank Löffler, et al. 2014. A survey of high level frameworks in block-structured adaptive mesh refinement packages. *J. Parallel and Distrib. Comput.* 74, 12 (2014), 3217–3227.

Douglas Enright, Stephen Marschner, and Ronald Fedkiw. 2002. Animation and rendering of complex water surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques.* Association for Computing Machinery, New York, NY, USA, 736–744.

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

Yu Fang, Yuanming Hu, Shi-Min Hu, and Chenfanfu Jiang. 2018. A Temporally Adaptive Material Point Method with Regional Time Stepping. *Computer Graphics Forum* 37, 8 (2018), 195–204.

Florian Ferstl, Ryoichi Ando, Chris Wojtan, RÃ¼diger Westermann, and Nils Thuerey. 2016. Narrow Band FLIP for Liquid Simulations. *Computer Graphics Forum* 35, 2 (2016), 225–232.

Florian Ferstl, Rüdiger Westermann, and Christian Dick. 2014. Large-scale liquid simulation on adaptive hexahedral grids. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1405–1417.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–12.

Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018. GPU optimization of material point methods. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–12.

Frederic Gibou, Ronald P Fedkiw, Li-Tien Cheng, and Myungjoo Kang. 2002. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *J. Comput. Phys.* 176, 1 (2002), 205–227.

Christoph Gissler, Stefan Band, Andreas Peer, Markus Ihmsen, and Matthias Teschner. 2017. Generalized drag force for particle-based simulations. *Computers & Graphics* 69 (2017), 1–11.

Francis H Harlow, J Eddie Welch, et al. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids* 8, 12 (1965), 2182.

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

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

Jeong-Mo Hong and Chang-Hun Kim. 2005. Discontinuous fluids. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 915–920.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019a. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

Yuanming Hu, Xinxin Zhang, Ming Gao, and Chenfanfu Jiang. 2019b. On hybrid lagrangian-eulerian simulation methods: practical notes and high-performance aspects. In *ACM SIGGRAPH 2019 Courses (SIGGRAPH '19).* Association for Computing Machinery, New York, NY, USA, Article 16, 246 pages.

Markus Ihmsen, Nadir Akinci, Gizem Akinci, and Matthias Teschner. 2012. Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer* 28 (2012), 669–677.

David Jacqmin. 1999. Calculation of two-phase Navier–Stokes flows using phase-field modeling. *Journal of computational physics* 155, 1 (1999), 96–127.

Suhas S Jain. 2022. Accurate conservative phase-field method for simulation of two-phase flows. *J. Comput. Phys.* 469 (2022), 111529.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

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

Janghee Kim, Deukhyun Cha, Byungjoon Chang, Bonki Koo, and Insung Ihm. 2006. Practical animation of turbulent splashing water. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06).* Eurographics Association, Goslar, DEU, 335–344.

Markus Kowarschik, Iris Christadler, and Ulrich Rüde. 2006. Towards Cache-Optimized Multigrid Using Patch-Adaptive Relaxation. In *Applied Parallel Computing. State of the Art in Scientific Computing*, Jack Dongarra, Kaj Madsen, and Jerzy Waśniewski (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 901–910.

Steve Lesser, Alexey Stomakhin, Gilles Daviet, Joel Wretborn, John Edholm, Noh-Hoon Lee, Eston Schweickart, Xiao Zhai, Sean Flynn, and Andrew Moffat. 2022. Loki: a unified multiphysics simulation framework for production. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–20.

Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. 2020. Kinetic-based multiphase flow simulation. *IEEE Transactions on Visualization and Computer Graphics* 27, 7 (2020), 3318–3334.

Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient kinetic simulation of two-phase flows. *ACM Transactions on Graphics* 41, 4 (2022), 114.

Wei Li, Kui Wu, and Mathieu Desbrun. 2024. Kinetic Simulation of Turbulent Multifluid Flows. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–17.

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-band topology optimization on a sparsely populated grid. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.

Haixiang Liu, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. 2016. A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–12.

Xu-Dong Liu, Ronald P Fedkiw, and Myungjoo Kang. 2000. A boundary condition capturing method for Poisson's equation on irregular domains. *Journal of computational Physics* 160, 1 (2000), 151–178.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 457–462.

Frank Losasso, Jerry Talton, Nipun Kwatra, and Ronald Fedkiw. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics* 14, 4 (2008), 797–804.

Scott P MacLachlan, Jok Man Tang, and Cornelis Vuik. 2008. Fast and robust solvers for pressure-correction in bubbly flow problems. *J. Comput. Phys.* 227, 23 (2008), 9742–9761.

Jan Mandel. 1988. Multi-Grid Methods and Applications (Wolfgang Hackbusch). *SIAM Rev.* 30, 3 (1988), 519–520.

Daniel F Martin, Phillip Colella, and Daniel Graves. 2008. A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions. *J. Comput. Phys.* 227, 3 (2008), 1863–1886.

A. McAdams, E. Sifakis, and J. Teran. 2010. A parallel multigrid Poisson solver for fluids simulation on large grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10).* Eurographics Association, Goslar, DEU, 65–74.

Viorel Mihalef, Samet Kadioglu, Mark Sussman, Dimitris Metaxas, and Vassilios Hurmusiadis. 2008. Interaction of two-phase flow with animated models. *Graphical Models* 70, 3 (2008), 33–42.

V. Mihalef, D. Metaxas, and M. Sussman. 2007. Textured Liquids based on the Marker Level Set. *Computer Graphics Forum* 26, 3 (2007), 457–466.

Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. 2009. Simulation of two-phase flow with sub-scale droplet and bubble effects. *Computer Graphics Forum* 28, 2 (2009), 229–238.

Shahab Mirjalili, Suhas S Jain, and Micheal Dodd. 2017. Interface-capturing methods for two-phase flows: An overview and recent developments. *Center for Turbulence Research Annual Research Briefs* 2017, 117-135 (2017), 13.

Marcus Mohr and Roman Wienands. 2004. Cell-centred multigrid revisited. *Computing and Visualization in Science* 7, 3 (2004), 129–140.

Joe J Monaghan. 1992. Smoothed particle hydrodynamics. *In: Annual review of astronomy and astrophysics. Vol. 30 (A93-25826 09-90), p. 543-574.* 30 (1992), 543–574.

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)* 32, 3 (2013), 1–22.

Mohammad Sina Nabizadeh, Ritoban Roy-Chowdhury, Hang Yin, Ravi Ramamoorthi, and Albert Chern. 2024. Fluid Implicit Particles on Coadjoint Orbits. *ACM Trans. Graph.* 43, 6, Article 270 (Nov. 2024), 38 pages.

Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector fluids. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

Juan Navarro, Sitararn Iyer, Peter Druschel, and Alan Cox. 2002. Practical, transparent operating system support for superpages. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 89–104.

Michael B Nielsen and Ole Østerby. 2013. A two-continua approach to Eulerian simulation of water spray. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Kevin M. Olson and Peter MacNeice. 2005. An Overview of the PARAMESH AMR Software Package and Some of Its Applications. In *Adaptive Mesh Refinement - Theory and Applications*, Tomasz Plewa, Timur Linde, and V. Gregory Weirs (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 315–330.

Stanley Osher and James A Sethian. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics* 79, 1 (1988), 12–49.

Ashish Panwar, Sorav Bansal, and K. Gopinath. 2019. HawkEye: Efficient Fine-grained OS Support for Huge Pages. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 347–360. https://doi.org/10.1145/3297858.3304064

Chuck Pheatt. 2008. Intel® threading building blocks. *Journal of Computing Sciences in Colleges* 23, 4 (2008), 298–298.

Stéphane Popinet. 2003. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. *Journal of computational physics* 190, 2 (2003), 572–600.

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

Ziyin Qu, Minchen Li, Fernando De Goes, and Chenfanfu Jiang. 2022. The power particle-in-cell method. *ACM Trans. Graph.* 41, 4, Article 118 (2022), 13 pages.

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Wouter Raateland, Torsten Hädrich, Jorge Alejandro Amador Herrera, Daniel T Banuti, Wojciech Pałubicki, Sören Pirk, Klaus Hildebrandt, and Dominik L Michels. 2022. Dcgrid: An adaptive grid structure for memory-constrained fluid simulation on the gpu. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 1 (2022), 1–14.

Mehdi Raessi and Heinz Pitsch. 2012. Consistent mass and momentum transport for simulating incompressible interfacial flows with large density ratios using the level set method. *Computers & Fluids* 63 (2012), 70–81.

M. Raw. 1996. Robustness of coupled Algebraic Multigrid for the Navier-Stokes equations. In *34th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Reno, Nevada.

Ronald A. Remmerswaal and Arthur E. P. Veldman. 2022. On Simulating Variability of Sloshing Loads in LNG Tanks. In *Proceedings of the ASME 2022 41st International Conference on Ocean, Offshore and Arctic Engineering (OMAE)*, Vol. 7: CFD and FSI. ASME, Hamburg, Germany, V007T08A012.

Bo Ren, Wei He, Chen-Feng Li, and Xu Chen. 2021. Incompressibility enforcement for multiple-fluid SPH using deformation gradient. *IEEE Transactions on Visualization and Computer Graphics* 28, 10 (2021), 3417–3427.

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 Transactions on Graphics (TOG)* 33, 5 (2014), 1–11.

Arch Robison, Michael Voss, and Alexey Kukanov. 2008. Optimization via Reflection on Work Stealing in TBB. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE, Miami, FL, USA, 1–8.

Ulrich Rüde. 1993. Fully adaptive multigrid methods. *SIAM J. Numer. Anal.* 30, 1 (1993), 230–248.

Ruben Scardovelli and Stéphane Zaleski. 1999. Direct numerical simulation of free-surface and interfacial flow. *Annual review of fluid mechanics* 31, 1 (1999), 567–603.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–12.

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 Transactions on Graphics (TOG)* 41, 4 (2022), 1–18.

William A Sirignano. 2010. *Fluid dynamics and transport of droplets and sprays*. Cambridge university press, Cambridge, United Kingdom.

B. Solenthaler and R. Pajarola. 2008. Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 211–218.

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

K_R Sreenivasan. 1991. Fractals and multifractals in fluid turbulence. *Annual review of fluid mechanics* 23, 1 (1991), 539–604.

Jos Stam. 2023. *Stable Fluids* (1 ed.). Association for Computing Machinery, New York, NY, USA, 779–786.

Alexey Stomakhin, Steve Lesser, Joel Wretborn, Sean Flynn, Johnathan Nixon, Nicholas Illingworth, Adrien Rollet, Kevin Blom, and Douglas Mchale. 2023. Pahi: A Unified Water Pipeline and Toolset. In *Proceedings of the 2023 Digital Production Symposium* (Los Angeles, CA, USA). Association for Computing Machinery, New York, NY, USA, Article 11, 13 pages.

Lucas Stringhetti. 2024. Houdini Water Simulation Techniques: A Detailed Breakdown. The VFX Media. Retrieved November 29, 2024, from https://www.thevfxmedia.com/articles/houdini-water-simulation-techniques-from-lucas-stringhetti-a-detailed-breakdown.

Hari Sundar, Rahul S Sampath, and George Biros. 2008. Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel. *SIAM Journal on Scientific Computing* 30, 5 (2008), 2675–2708.

Mark Sussman, Peter Smereka, and Stanley Osher. 1994. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational physics* 114, 1 (1994), 146–159.

JM Tang and C Vuik. 2007. Efficient deflation methods applied to 3-D bubbly flow problems. *Electronic Transactions on Numerical Analysis* 26 (2007), 330–349.

Osamu Tatebe. 1993. The Multigrid Preconditioned Conjugate Gradient Method. In *The Sixth Copper Mountain Conference on Multigrid Methods, Part 2*. NASA, Copper Mountain, Colorado, USA, 621—-634.

Jannis Teunissen and Ute Ebert. 2018. Afivo: A framework for quadtree/octree AMR with shared-memory parallelization and geometric multigrid methods. *Computer Physics Communications* 233 (2018), 156–166.

Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. 2000. *Multigrid*. Elsevier, Amsterdam, The Netherlands.

Dong-xu Wang, Jia-wen Sun, Jin-song Gui, Zhe Ma, De-zhi Ning, and Ke-zhao Fang. 2019. A numerical piston-type wave-maker toolbox for the open-source library OpenFOAM. *Journal of Hydrodynamics* 31, 4 (2019), 800–813.

Xinlei Wang, Yuxing Qiu, Stuart R Slattery, Yu Fang, Minchen Li, Song-Chun Zhu, Yixin Zhu, Min Tang, Dinesh Manocha, and Chenfanfu Jiang. 2020. A massively parallel and scalable multi-GPU material point method. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 30–1.

Xiaokun Wang, Yanrui Xu, Sinuo Liu, Bo Ren, Jiří Kosinka, Alexandru C. Telea, Jiamin Wang, Chongming Song, Jian Chang, Chenfeng Li, Jian Jun Zhang, and Xiaojuan Ban. 2024. Physics-based fluid simulation in computer graphics: Survey, research trends, and challenges. *Computational Visual Media* 10, 5 (October 2024), 803–858.

Daniel Weber, Johannes Mueller-Roemer, AndrÃ© Stork, and Dieter Fellner. 2015. A Cut-Cell Geometric Multigrid Poisson Solver for Fluid Simulation. *Computer Graphics Forum* 34, 2 (2015), 481–491.

Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. 2017. Infinite continuous adaptivity for incompressible SPH. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–10.

Chris Wojtan, Nils Thürey, Markus Gross, and Greg Turk. 2009. Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 Papers*. Association for Computing Machinery, New York, NY, USA, Article 76, 10 pages.

Kui Wu, Nghia Truong, Cem Yuksel, and Rama Hoetzlein. 2018. Fast Fluid Simulations with Sparse Volumes on the GPU. *Computer Graphics Forum* 37, 2 (2018), 157–167.

Feng Xiao. 2012. *Large Eddy Simulation of liquid jet primary breakup*. Ph. D. Dissertation. Loughborough University.

Han Yan and Bo Ren. 2023. High Density Ratio Multi-Fluid Simulation with Peridynamics. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–14.

Guan Heng Yeoh and Jiyuan Tu. 2019. *Computational techniques for multiphase flows*. Butterworth-Heinemann, Oxford, United Kingdom.

Yadong Zeng, Anqing Xuan, Johannes Blaschke, and Lian Shen. 2022. A parallel cell-centered adaptive level set framework for efficient simulation of two-phase flows with subcycling and non-subcycling. *J. Comput. Phys.* 448 (2022), 110740.

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

Weiqun Zhang, Andrew Myers, Kevin Gott, Ann Almgren, and John Bell. 2021. AMReX: Block-structured adaptive mesh refinement for multiphysics applications. *The International Journal of High Performance Computing Applications* 35, 6 (2021), 508–526.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.