Spatiotemporal FLIP for Fast Free-Surface and Two-Phase Simulation With Very Large Time Steps

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

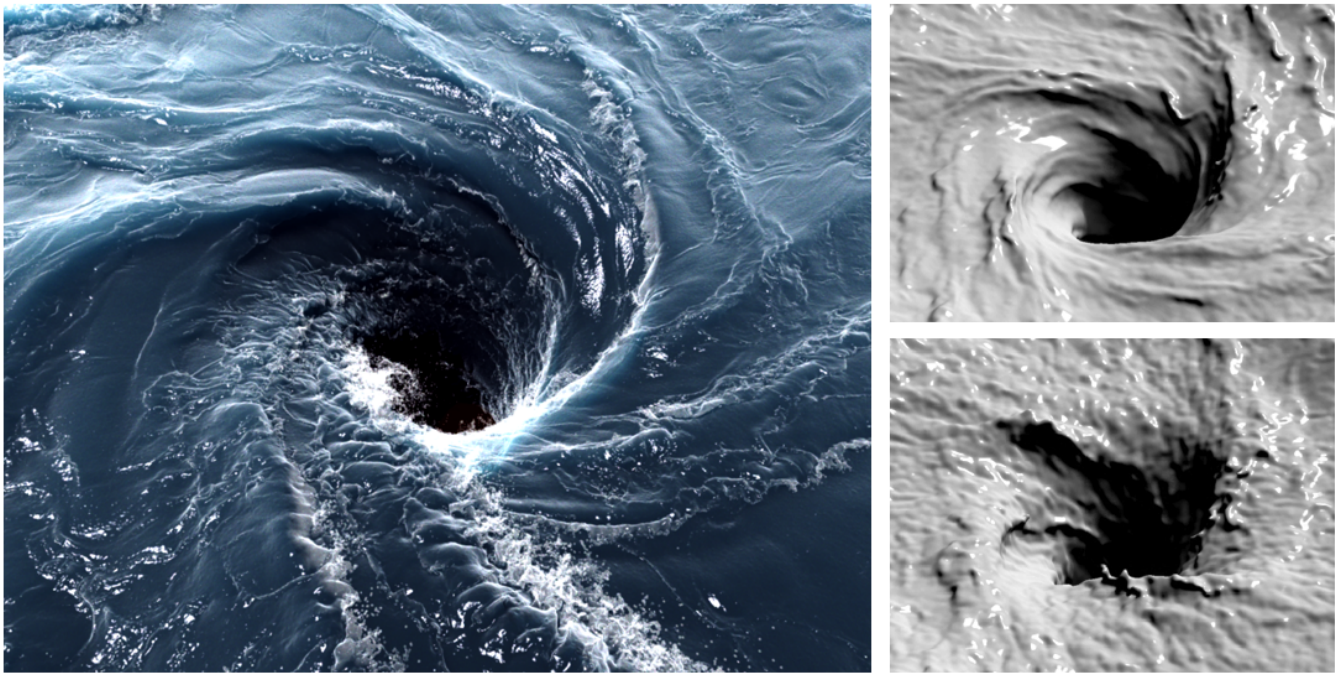
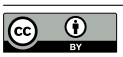


Fig. 1. Our spatiotemporal particle sampling preserves flow coherence at time steps several-fold larger than those typically used in liquid simulation, while incurring minimal computational overhead. Left: large-scale whirlpool with over 2 billion particles and $3072 \times 3072 \times 2048$ adaptive grid resolution, completed in 3.4 days at target CFL number 15. The same simulation would have taken an estimated 2 weeks with standard FLIP at CFL=2. Right: Free-surface comparison: our method (top) maintains detailed vortical flow and a smooth surface, whereas standard FLIP (bottom) remains numerically stable but develops severe aliasing-driven surface oscillations that rapidly dominate and erase meaningful flow detail.

We present ST-FLIP, a spatiotemporal extension of the Fluid-Implicit Particle (FLIP) method for incompressible free-surface and two-phase liquid simulation. ST-FLIP enables time steps up to an order of magnitude larger than those typically used in CFL-constrained solvers, while preserving detailed flow structures and visual fidelity. It addresses a common failure mode of large time steps in hybrid particle–grid liquid solvers: temporal undersampling of particle motion produces aliasing-driven free-surface artifacts after projection.

Our key idea is to interpret particles as samples in four-dimensional space-time: in addition to standard spatial jittering, we randomize particle positions along the time axis as well and perform particle-to-grid deposition using a separable 4D kernel. This yields a Monte Carlo estimator of per-step time-slab-integrated particle quantities. Although particles are treated as samples in 4D space-time, our approach works as a lightweight plug-in by collapsing to slab-integrated 3D grid fields for projection. Building on recent particle-based phase-field work, we reuse the particle-to-grid weight accumulators as a conceptual space–time phase field, providing variable-coefficient projection weights and eliminating the need for per-step surface reconstruction. The method can be easily integrated into existing FLIP/PIC or APIC solvers with negligible additional computational cost per time step. The effectiveness of our approach is demonstrated through a series of comparisons with state-of-the-art solvers, yielding several-fold speedups for multi-billion-particle simulations at high effective 3D resolutions on a single workstation.

Authors' Contact Information: Bernhard Braun, bernhard.braun@pbs.cit.tum.de, Technical University of Munich, Germany; Rene Winchenbach, rene.winchenbach@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.



ACM 1557-7368/2026/7-ART76
https://doi.org/10.1145/3811289

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

ACM Reference Format:
Bernhard Braun, Rene Winchenbach, Jan Bender, and Nils Thuerey. 2026. Spatiotemporal FLIP for Fast Free-Surface and Two-Phase Simulation With Very Large Time Steps. *ACM Trans. Graph.* 45, 4, Article 76 (July 2026), 20 pages. https://doi.org/10.1145/3811289

1 Introduction

Cinematic liquid effects span a vast range of scales–from bulk motion to thin sheets and spray–and meeting this visual bar routinely pushes simulations to extreme resolutions and prohibitive compute budgets. Hybrid Eulerian–Lagrangian methods such as PIC/FLIP/APIC [Bridson 2008; Jiang et al. 2015; Qu et al. 2022] have emerged as the backbone of many production pipelines [Side-Effects-Software 2024; Stringhetti 2024], in part because they combine detail-preserving advection with high numerical stability, in practice often allowing for time steps larger than those imposed by the CFL condition [Courant et al. 1928] without catastrophic numerical instability [Bridson 2015]. Taking time steps beyond the CFL-limit is attractive, especially in graphics, because 3D fluid simulation suffers from the $\approx O(n^4)$ scaling with grid resolution n: doubling the resolution not only increases the number of grid cells eightfold but also dictates halving the time step, for an overall runtime increase of about 16×. For high resolutions, CFL-conforming time steps can become orders of magnitude smaller than the video frame interval, substantially limiting the computational feasibility of large cinematic-scale simulations. Furthermore, large time steps can be essential for fast previews at high resolutions, accelerating feedback and turnaround cycles in VFX production.

However, numerical stability does not guarantee simulation accuracy or visual fidelity. In practice, the quality of the numerical solution can deteriorate rapidly with increasing *CFL number* (the maximum distance, in units of grid cells, that fluid parcels travel during a time step). As shown in the left column of Figure 2 and the left columns of Figure 8, temporal aliasing and unphysical ripples and waves are amplified by the pressure solve and quickly degrade the physical structure of the flow. This is especially critical for liquid or multiphase simulations, where interface kinematics (e.g., splashes and breaking waves) are highly nonlinear and topologically complex.

In this work, we present a space-time extension of standard FLIP, which we call ST-FLIP, that supports CFL numbers and, consequently, time steps an order of magnitude larger than those typically used in standard liquid solvers, while retaining comparable simulation quality and long-term stability of flow features. While our method, like any solver, cannot completely eliminate quality loss at larger time steps, the degradation is much more gradual (Figure 2, right column), yielding plausible results even at CFL numbers greater than ten, delivering multi-fold speedups in wall-clock time.

Another key design goal is compatibility with globally adaptive time stepping: our spatiotemporal sampling and staggered-time update remain well-defined and robust when Δt changes from step to step, which is essential for production-scale simulation. Importantly, our scheme can be easily integrated into existing PIC/FLIP/APIC solver pipelines and incurs negligible computational overhead: it requires no additional grids, no additional linear solves, and no extra passes over particles or fields. In Section 4, we demonstrate

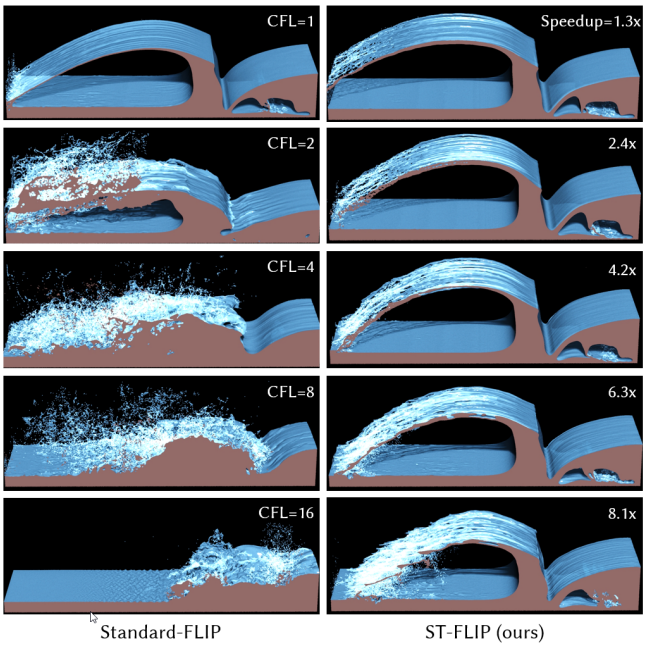


Fig. 2. Dam break simulated with FLIP (left column) and our method, ST-FLIP (right column). ST-FLIP maintains flow coherence even at very large time steps, speeding up simulation times multi-fold (upper right corners show speedup factors compared to the CFL = 1 FLIP baseline).

its versatility by integrating ST-FLIP into the recently published high performance PF-FLIP algorithm [Braun et al. 2025]. PF-FLIP is already optimized for large time steps via adaptively sub-stepped higher-order advection. Even on this strong baseline, incorporating ST-FLIP yields a speedup of about a factor of two for large-scale, high-resolution *two-phase* simulations, which are numerically significantly more challenging than the free-surface case.

The basic idea of our approach is illustrated in Figure 3, which depicts particles moving along world lines in space-time. Standard FLIP (left column) samples spatial cells using a set of spatially jittered particles, but at CFL > 1 the motion over a step leaves obvious gaps in spatiotemporal coverage. To address this, we propose to jitter particle samples not only in space but also in time, thus effectively sampling four-dimensional space–time cells instead of the usual three-dimensional spatial cells. As the right-hand side of Figure 3 shows, this leads to a much more uniform sampling coverage of space-time (and, consequently, of space). We achieve this by extending the standard particle-to-grid (P2G) transfer to four-dimensional space–time, orthogonally augmenting the usual 3D spatial kernel with a carefully designed additional 1D temporal kernel that distributes each particle's contribution along the time dimension through a staggered-in-time stochastic advection scheme. Although our particles live in 4D space-time, we keep the solver a lightweight drop-in by collapsing the conceptual 4D slab of a step's time interval to slab-integrated 3D grid fields for pressure projection.

Our scheme leverages the well-known insensitivity of Monte Carlo integration to dimensionality [Caflisch 1998], allowing the

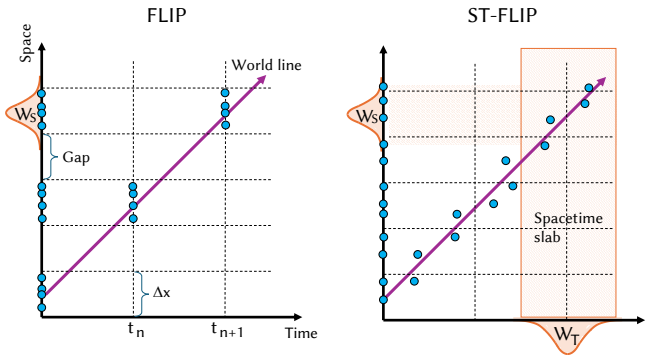


Fig. 3. Moving particles along world lines in space-time at CFL = 2 (uniform speed= $2\Delta x/\Delta t$). Left: standard FLIP moves particles instantaneously to the next discrete time point, leaving large gaps in space-time coverage. Right: ST-FLIP samples the 4D space-time much more uniformly while using the same number of particles. In addition to the standard (spatial) particle-to-grid transfer kernel W_S, ST-FLIP also uses a transfer kernel W_T for the temporal dimension.

additional temporal dimension to be covered with essentially *the same number of particles per cell* as in standard FLIP. While increasing the dimension does increase estimator variance, going from 3D to 4D only moderately increases noise; importantly, this noise is largely unbiased and uncorrelated and is therefore amenable to standard surface-smoothing and denoising post-processing techniques [Bhatacharya et al. 2011; Clarenz et al. 2003; Müller 2009]; see also Appendix B. We note that this is similar in spirit to Monte Carlo methods in rendering, such as distributed ray tracing and path tracing, where samples are distributed not only over the 2D pixel area but also, simultaneously, over additional dimensions (e.g., time and lens position) to efficiently capture effects like motion blur and depth of field [Cook et al. 1984; Kajiya 1986].

Another key feature of our method, inherited from our generalization of PF-FLIP, is that it does not require the geometrical construction of a liquid surface, whether explicit by meshing or implicit as a signed distance field. Instead we draw inspiration from the recently proposed PF-FLIP and re-use the particle-to-grid weight accumulators as a *phase field* in the spirit of *diffuse interface methods* [Anderson et al. 1998; Jacqmin 1999]. ST-FLIP extends this idea to an interface that is not only diffuse in space but also over a (short) period in time. Notably, we obtain this space-time phase-field virtually for free by re-using the weight accumulators from our 4D-P2G step. Besides computational efficiency, this "volumetric" approach to free-surface and two-phase simulation also provides a theoretical basis for the pressure projection as a space-time extension to the variational framework of Batty et al. [2007]. Whenever a sharper representation of the surface is needed (e.g., for final rendering), we perform standard surface reconstruction (Appendix B) directly from the particles, whose time-jittered positions we re-synchronize on-the-fly to the global time step to avoid unwanted motion blur. This is typically done only at output frames (typically every 3–10 time steps), not every step.

Contributions. In summary, we make the following contributions:

- A spatiotemporal extension for widely used FLIP-like particle-grid solvers that supports adaptive time steps an order of magnitude larger than those used by prior methods at comparable visual fidelity, while integrating easily into existing solvers with minimal runtime overhead.
- A free-surface and two-phase interface representation derived by reusing P2G weight accumulators as a space–time extension of PF-FLIP, providing pressure projection coefficients and avoiding per-step surface reconstruction.

2 Related Work

Fluid simulation has been a major topic of study in computer graphics for over twenty years [Bridson 2008; Losasso et al. 2004; Stam 1999], and we refer to the comprehensive textbook of Bridson [2015] and recent survey works [Ren et al. 2018; Wang et al. 2024] for an in-depth overview.

Hybrid Particle–Grid Liquid Simulation. Hybrid Eulerian-Lagrangian methods are widely used for liquid simulation in graphics, combining an Eulerian grid (e.g., for pressure projection and other PDE-based updates) with Lagrangian particles for advection and free-surface tracking. Most of such schemes trace back to particle-in-cell (PIC) methods for fluid dynamics [Harlow 1964], where velocities are transferred from particles to a background grid, updated with a finite-difference PDE solve, and then interpolated back to the particles. Although robust, classical PIC is strongly dissipative and rapidly damps vortical motion. To reduce this dissipation, the FLIP method [Brackbill and Ruppel 1986; Zhu and Bridson 2005] updates particle velocities using grid velocity increments rather than absolute values, preserving small-scale detail at the cost of increased noise. APIC [Jiang et al. 2015] enriches each particle with an affine velocity field, yielding low-dissipation yet smooth and stable transfers at the cost of increased storage requirements. PolyPIC generalizes this idea to higher-order polynomial velocity fields [Fu et al. 2017], and more recent methods such as IPIC [Sancho et al. 2024] leverage flow maps and an impulse-gauge formulation to further improve detail preservation.

To meet increasing resolution and runtime requirements, a complementary line of work has focused on scalability through spatial adaptivity [Ando et al. 2013; Ferstl et al. 2014; Losasso et al. 2004; Raateland et al. 2022; Setaluri et al. 2014] and faster, more scalable pressure Poisson solvers [McAdams et al. 2010; Setaluri et al. 2014; Shao et al. 2022]. These advances are orthogonal to our focus on large time steps.

Recently, Braun et al. [2025] introduced a particle-based phase-field representation for two-phase simulation that reuses particle-to-grid (P2G) mass accumulation on a purely spatial grid. Our method builds on this idea and extends it to a *space–time* mass estimator, coupled to our large-time-step ST-FLIP formulation.

Large Time Steps. Although larger time steps can substantially reduce total simulation cost, comparatively few liquid simulation methods in graphics are designed explicitly for the regime of very large time steps (i.e., allowing for CFL $\gg$ 1). Here, CFL denotes the well-known Courant–Friedrichs–Lewy number [Bridson 2015; Courant et al. 1928] which measures the maximum distance (in units

of cell spacing Δx) that fluid parcels (particles) travel during a single time step Δt.

Many classical Eulerian methods based on semi-Lagrangian advection [Stam 1999] are unconditionally stable and, in principle, can use time steps corresponding to arbitrarily large CFL numbers. However, unconditional stability in this context only guarantees that the simulation will not catastrophically diverge ("blow up") but in practice, the visual *quality* of the numerical solution of liquid simulations often deteriorates once the CFL number exceeds about 2–3 [Koike et al. 2020; Lentine et al. 2012].

By contrast, purely Lagrangian schemes (e.g., SPH) [Ihmsen et al. 2014; Koschier et al. 2022; Monaghan 1992], explicit VOF-based CFD methods Hirt and Nichols [1981]; Kleefsman et al. [2005], and kinetic approaches such as LBM [Li et al. 2020, 2022; Wang et al. 2025a] typically impose time-step restrictions equivalent to CFL ≤ 1 for both stability and accuracy. Hybrid particle–grid methods (e.g., PIC/FLIP variants) can remain numerically stable for a wide range of time steps, but—similar to semi-Lagrangian approaches—their accuracy and visual quality can become unacceptable as the CFL number grows (Figure 2, left column).

A number of recent advances have introduced more sophisticated particle–grid formulations and representations, including PolyPIC [Fu et al. 2017], flow-map-based methods [Chen et al. 2025; Wang et al. 2025b; Zhou et al. 2024], impulse- and gauge-variable formulations (IPIC) [Sancho et al. 2024], covector-fluid methods [Nabizadeh et al. 2022] and coadjoint-orbit-based CO-FLIP [Nabizadeh et al. 2024]. These approaches primarily target improved preservation of vortical-flow structure and reduced numerical dissipation, often in the context of single-phase (e.g., smoke or gas) scenarios. Large time steps are typically not the primary design objective in these methods, and reported experiments are commonly conducted near CFL ≈ 1.

Only a limited number of works directly target liquid simulation at very large CFL values (e.g., on the order of ten and above). In the Eulerian/semi–Lagrangian context Chentanez and Müller [2012] couple an interface sharpening scheme with mass-conserving semi–Lagrangian advection, involving extra scatter- and reweighting grid-passes. Reported CFL numbers are in the range of 8–32 for examples with a maximum grid resolution of 256×128^2. Lentine et al. [2012] similarly combine mass- and momentum-conserving advection [Lentine et al. 2011], a particle level set (PLS) method, and a sharpened color-function transport scheme to enable time steps corresponding to reported CFL values in the range 10–40. Although the latter method uses particles for low-dissipative surface representation, the velocity advection itself remains a semi-Lagrangian scheme, which can introduce substantial dissipation of flow detail. In addition, the overall algorithm couples multiple numerical components; the authors report that this increases per-step runtime relative to a standard baseline—approximately doubling the simulation time per step in their reported setting. In both works, the reported CFL values are computed from the maximum fluid velocity over the entire simulation, so per-frame CFL values may be substantially smaller during calmer periods of the flow. In contrast, in our experiments we always report the user-specified *target CFL* which is a fixed and reproducible measure, more representative of the overall time-step regime (see Figure 7).

More relevant to our context of *low-dissipation* hybrid particle-grid methods, Kugelstadt et al. [2019] relaxed time-step restrictions via implicit density projection (IDP) to address FLIP-typical uneven particle distributions. In their experiments, maximum CFL numbers of 20 are reported for a 2D dam-break example. While this method can be easily integrated into existing solvers, it adds notable runtime overhead due to the need to solve an additional pressure Poisson equation per time step, which can be expensive at high resolutions.

Spatially Adaptive Time Stepping. Another approach to increasing the *effective* time step is spatially adaptive time stepping that goes beyond locally sub-stepped advection by adapting the entire update—including pressure projection—in a spatially varying manner. Koike et al. [2020] propose such a method in the Eulerian setting and report speedups corresponding to an effective overall CFL of up to 6. However, because the method uses semi-Lagrangian advection for both velocity and the free surface, it inherits the associated dissipative behavior. Moreover, time-step-based domain decomposition introduces nontrivial coupling issues for boundary conditions between different "time levels" in the pressure solve: since the pressure equation is elliptic, it cannot be consistently partitioned across multiple time steps without violating strict incompressibility.

Spatiotemporal Deposition in Plasma PIC. In some plasma-physics particle-in-cell codes [Birdsall and Langdon 2005; Villasenor and Buneman 1992], current is deposited from particles to the grid by accounting for particle motion over the entire time step, rather than instantaneously as in classical PIC, which is effectively equivalent to assigning each particle a temporal shape function spanning Δt. This deterministic multi-deposition scheme differs from the present work in that our algorithm retains instantaneous deposition of mass and velocity, relying instead on randomization of particle positions and Monte Carlo integration to achieve spatiotemporal coverage. Consequently, our space-time splatting requires only a single deposition per particle per time step, which is attractive in practice because neighborhood deposition kernels are often memory bandwidth- and synchronization-bound.

Positioning Summary. Prior approaches for very large time steps (CFL $\gtrsim 10$) in graphics target single-phase simulation and either accept increased velocity dissipation (semi-Lagrangian velocity advection) and/or incur significant runtime-overhead (e.g., additional Poisson solves) and implementation complexity (coupling multiple schemes). In contrast, we support two-phase simulations as well and treat Lagrangian particles as space-time samples to obtain low-dissipation at very large time steps with a simple FLIP-extension that can be easily integrated in existing solvers while incurring minimal per-step runtime overhead.

3 Method

This section proceeds as follows: after defining the governing equations and setting up preliminaries and notation, we describe the spatiotemporal sampling machinery that enables large adaptive time steps: particle advection and our 4D-to-3D deposition with temporal jitter and residual carryover (Sections 3.3–3.5). We then describe how these temporally stratified quantities are used for

the diffuse interface and pressure projection (Sections 3.6–3.7), and finally summarize the full algorithm (Section 3.10).

3.1 Goal and Governing Equations

Incompressible Free-Surface and Two-phase Flow. Our goal is to numerically solve the Euler equations (treating the flow as nominally inviscid, as is common in graphics)

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u}\cdot\nabla)\vec{u} - \frac{\nabla p}{\rho} + \frac{\vec{f}}{\rho} \tag{1}$$

$$\nabla\cdot\vec{u} = 0, \tag{2}$$

together with advection of a phase indicator $\chi_\ell \in \{0,1\}$:

$$\partial_t \chi_\ell + \vec{u}\cdot\nabla\chi_\ell = 0, \qquad \rho = \rho_\ell\,\chi_\ell + \rho_g\,(1-\chi_\ell), \tag{3}$$

for a heavy liquid (water) and a light gas (air) in a three-dimensional domain $\Omega \subset \mathbb{R}^3$ with $\rho \in \{\rho_\ell, \rho_g\}$, t, $\vec{u}$, p, and $\vec{f}$ denoting density, time, velocity, pressure, and external forces. Subscripts ℓ and g refer to the liquid and gas phases, respectively.

Boundary and Interface Conditions. On solid obstacles $\partial\Omega_s$ we impose the no-through condition $\vec{u}\cdot\vec{n} = 0$ on $\partial\Omega_s$. At the liquid–gas interface Γ, the kinematic condition coincides with (3); in the inviscid, no-surface-tension setting the normal stress is continuous, hence p is continuous across Γ.

Free-Surface Model. We treat the common case of *free-surface* flow as the limit $\rho_g \to 0$, where we take $p = p_{\text{atm}}$ on Γ and, via gauge transformation $p \leftarrow p - p_{\text{atm}}$, set Dirichlet $p = 0$ on Γ.

3.2 Preliminaries and Notation

We denote the current time level by t_n with variable step size $\Delta t_n = t_{n+1} - t_n$. For spatial discretization, we use a standard MAC grid [Harlow et al. 1965] over a domain $\Omega \subset \mathbb{R}^3$ with cell size Δx, cell centers $\{\vec{x}_c\}$, and face centers $\{\vec{x}_f\}$. Particles $p = 1,\dots,N$ carry mass m_p, position $\vec{x}_p^n$, and velocity $\vec{u}_p^n$. We assume that, initially, each cell contains N_{ppc} particles on average (e.g., eight for standard FLIP in 3D). We use $\text{clamp}(x,a,b) := \min(\max(x,a),b)$ and the standard cubic ramp $\text{smoothstep}(a,b;x) := s^2(3-2s)$ with $s = \text{clamp}\left(\frac{x-a}{b-a}, 0, 1\right)$.

3.3 Advection

At the end of each time step, we integrate particle trajectories, obtaining new positions

$$\vec{x}_p^{n+1} = \mathcal{A}(\vec{u}^{n+1}, \vec{x}_p^n, \Delta t_{p,\text{act}}^n). \tag{4}$$

$\mathcal{A}(\vec{u},\vec{x},\Delta t)$ is an advection operator, advancing a position $\vec{x}$ through velocity field $\vec{u}$ for a time interval Δt. Typically, $\mathcal{A}$ is a higher-order multistage ODE integrator (e.g., RK3). $\vec{u}^{n+1}$ is the end-of-step divergence-free advector velocity field and $\Delta t_{p,\text{act}}^n$ is a per-particle time step as defined below (Equation 11).

Locally Sub-Stepped Advection. One way to improve temporal adaptivity beyond globally adaptive time stepping is to use substeps within the advection operator such that the maximum distance traveled by a particle satisfies a *local CFL bound* $\text{CFL}_{\text{local}}$.

Sub-stepped advection is not new [Bridson 2015], so for fairness, we employ it (with $\text{CFL}_{\text{local}} = 1$ throughout) not only in our algorithm but in all implementations of previous methods that we compare against. One consequence of sub-stepped advection is that the speedup gained from larger time steps no longer scales linearly with the *global* CFL number, because the runtime of a global simulation time step now depends on Δt. In practice, this effect is moderate because (i) only fast-moving particles require more than one advection substep, and (ii) per-step runtime is typically dominated by particle-grid transfers and pressure projection – not advection.

External Forces. We apply external forces to the grid velocities using the global step size Δt_n, as in standard FLIP. For spatially uniform body accelerations (e.g., gravity), one could alternatively apply the update directly to particles using each particle's effective time interval, i.e., $\vec{u}_p \leftarrow \vec{u}_p + \Delta t_{p,\text{act}}^n \frac{1}{\rho}\vec{f}$, more in line with the interpretation of particles as spatiotemporal samples. In our experiments, this produced no measurable differences (likely because the per-step accumulation already corresponds to a time-step averaged update), so we retain the standard grid-based update to simplify integration into existing solvers and to naturally support spatially varying forces computed on the grid (e.g., surface tension).

3.4 Staggered Time Integration and Space–Time 4D → 3D Particle-to-Grid Transfer

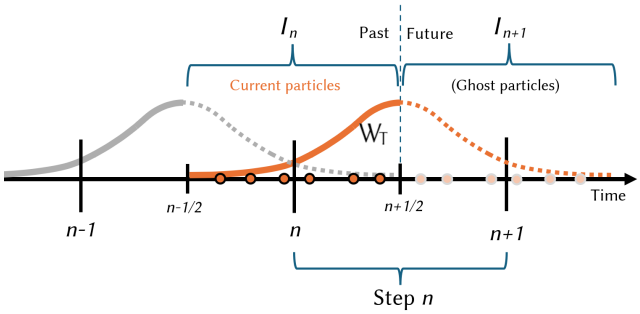


Fig. 4. Timeline overview: at the beginning of step n, particle sample times (orange dots) t_p^n are distributed over the current slab $I_n = [t_{n-1/2}, t_{n+1/2}]$ as a result of the previous stochastic advection step. Orange: current slab integration with W_T, grey: previous step. Dotted: virtual completion of the one-sided time-kernel with conceptual ghost-particles reflected from the past to the future-staggered time slab (See Section 3.8).

We augment each particle p with a scalar time attribute t_p^n, stored as an offset $\delta t_p^n \in [-0.5\Delta t_{n-1}, 0.5\Delta t_{n-1}]$, relative to the current global time t_n, so that each particle represents a sample $(\vec{x}_p, t_p)$ in space-time rather than only in space (Figure 3). Here, Δt_{n-1} is used because the current space-time positions are the result of the previous step. Being a small offset, δt_p^n can be stored in half-precision, adding only two bytes per particle in our implementation.

At the beginning of step n, we interpret each particle as located at its *jittered sample time*

$$t_p^n := t_n - \delta t_p^n,$$

with $\vec{x}_p^n$ already advanced by the previous step's advection using a particle-specific time step $\Delta t_{p,\text{act}}^{n-1}$ (Equation (11), Figure 4). As

a result, particle sample times are distributed over the staggered interval (time slab)

$$I_n := [t_n - \frac{1}{2}\Delta t_{n-1},\ t_n + \frac{1}{2}\Delta t_{n-1}], \tag{5}$$

centered at the current step time t_n.

Let $W_S : \mathbb{R}^3 \to \mathbb{R}_{\geq 0}$ be a compactly supported spatial transfer kernel and $W_T : \mathbb{R} \to \mathbb{R}_{\geq 0}$ a compact temporal kernel with

$$\int_{\mathbb{R}^3} W_S(\vec{r})\,\mathrm{d}\vec{r} = 1, \qquad \int_{\mathbb{R}} W_T(\tau)\,\mathrm{d}\tau = 1,$$

over dimensionless (grid-normalized) coordinates $\vec{r}$ and τ. We denote the separable space–time P2G transfer kernel

$$W_{ST}(\vec{x}, \tau) \ = \ W_S(\vec{x})\, W_T(\tau). \tag{6}$$

4D Monte Carlo Estimator. Classical PIC/FLIP estimate face mass and momentum *instantaneously*. In contrast, we target *time-slab-integrated* per-face quantities q_f :

$$\overline{q}_f = \int_{-1/2}^{1/2} q_f(t_n + \tau\Delta t_{n-1}) W_T(\tau)\,\mathrm{d}\tau, \tag{7}$$

where $q_f(t) = \int q(x_f + r\Delta x, t) W_S(r) dr$ is the standard spatially P2G-integrated quantity $q(t)$ evaluated at the center of the staggered MAC grid face f. $\tau = (t - t_n)/\Delta t_{n-1}$ is the normalized time coordinate centered at the current step n and the integral is over I_n. We normalize by Δt_{n-1} because the current space-time particle samples are the result of the *previous* advection step. Throughout, the overline denotes this *per-step* slab integration/collapse operator over I_n; note that it is *not* a recursive average across steps.

Evaluating (7) exactly is expensive; instead we define a 4D Monte-Carlo estimator for mass and momentum

$$\begin{aligned} \hat{m}^n(\vec{x}, t) &= \sum_p m_p\, W_S\Big(\frac{\vec{x}_p^n - \vec{x}}{\Delta x}\Big)\, W_T\Big(\frac{t_p^n - t}{\Delta t_{n-1}}\Big), \\ \hat{\boldsymbol{P}}^n(\vec{x}, t) &= \sum_p m_p\, \vec{u}_p^n\, W_S\Big(\frac{\vec{x}_p^n - \vec{x}}{\Delta x}\Big)\, W_T\Big(\frac{t_p^n - t}{\Delta t_{n-1}}\Big), \end{aligned} \tag{8}$$

where $\vec{u}_p^n$ denotes the current particle velocity as updated in the previous G2P step (Section 3.9). For the default jitter strength $\gamma = 1$ and in the absence of clamping (Section 3.5), the normalized particle sample times, evaluated at t_n, satisfy $\theta_p := (t_p^n - t_n)/\Delta t_{n-1} = -\delta t_p^n/\Delta t_{n-1} \sim \mathrm{Uniform}(-1/2,\ 1/2)$. Therefore

$$\mathbb{E}\big[q_f(t_n + \theta_p\,\Delta t_{n-1})\ W_T(\theta_p)\big] = \int_{-1/2}^{1/2} q_f(t_n + \tau\,\Delta t_{n-1})\ W_T(\tau)\,\mathrm{d}\tau.$$

Hence, Equation (8) is an unbiased Monte Carlo estimator of the kernel-normalized slab integral in Equation (7) for deposited mass and momentum for constant Δt_n. With adaptive Δt, clamping (Equation (11)) can introduce a small bias, but sharp step-size changes are rare in our settings, as discussed below in Section 3.5. For (spatially adaptive) $\gamma < 1$, θ_p concentrates towards 0, yielding a locally narrower effective temporal integration window; we analyze the resulting weight change for W_T in Section 3.10.

In principle, we could store $\hat{m}$ and $\hat{\boldsymbol{P}}$ on a "thin" 4D grid on the current time slab to enable higher-order-in-time reconstruction and we experimented with doing so to obtain linear-in-time interpolated velocities in the P2G and G2P steps. This, however, yielded no significant improvement of visual simulation quality while multiplying storage requirements. We therefore assume piecewise constant-in time representation and only keep a single set of 3D MAC-grid fields per step (4D→3D) by evaluating the slab-integration estimators (8) at the middle of the current time slab $t = t_n$

$$\begin{aligned} &\overline{m}_f := \hat{m}^n(\vec{x}_f, t_n),\ \overline{P}_f := \hat{P}^n(\vec{x}_f, t_n), \\ &\vec{u}_f^{\star} := \frac{\overline{\mathbf{P}}_f}{\overline{m}_f} \quad \text{for } \overline{m}_f > \epsilon_m, \end{aligned} \tag{9}$$

with provisional face velocities $\vec{u}_f^{\star}$ and a small threshold $\epsilon_m > 0$ (10^{-9} in our implementation) used to flag under-sampled faces for velocity extrapolation as usual.

3.5 Temporal Jitter and Residual Carryover

To cover the time interval with a single sample per particle and step, we *jitter* the per-particle advection time and keep track of a running residual so long–term advancement does not drift.

Let $s_p^n \sim \mathrm{Uniform}(0, 1)$ and $\xi_p^n = s_p^n - \frac{1}{2}$. With jitter strength $\gamma \geq 0$ (default $\gamma = 1$), newly jittered time step

$$\delta t_{p,\mathrm{jit}}^n = \gamma\, \xi_p^n\, \Delta t_n, \tag{10}$$

and per–particle residual δt_p^n ($\delta t_p^0 = 0$), we obtain the actual per-particle time step $\Delta t_{p,\mathrm{act}}^n$ to be passed to the advection operator in Equation (4) as

$$\begin{aligned} \Delta t_{p,\mathrm{act}}^n &= \mathrm{clamp}\Big(\Delta t_n + \delta t_p^n + \delta t_{p,\mathrm{jit}}^n,\ 0,\ 2\Delta t_n\Big), \\ \delta t_p^{n+1} &= \Delta t_n + \delta t_p^n - \Delta t_{p,\mathrm{act}}^n. \end{aligned} \tag{11}$$

Adaptive Time Stepping and Clamping. In the absence of clamping, Equation (11) reduces to $\delta t_p^{n+1} = -\xi_p^n\,\Delta t_n$ (for $\gamma = 1$), so particle sample times remain uniformly distributed over the current time slab; consequently producing slab-stratified sampling in time with $\mathbb{E}[\Delta t_{p,\mathrm{act}}^n] = \Delta t_n$. Although our adaptive time stepping tends to equalize successive time step lengths (by always subdividing the remaining video frame time into even parts), $\Delta t_{p,\mathrm{act}}^n$ may become negative or too large in cases of abrupt changes between Δt_{n-1} and Δt_n, so we clamp $\Delta t_{p,\mathrm{act}}^n$ to $[0, 2\Delta t_n]$ as in Equation (11). Importantly, the residual equals the global/particle time mismatch, $t_n - t_p^n = \delta t_p^n$; since the global CFL condition enforces $\Delta t_n \leq \Delta t_{\max}$, we show in Appendix A that $|t_p^n - t_n| = |\delta t_p^n| \leq \Delta t_{\max}/2$ for all n, preventing random-walk drift of particle sampling times. So clamping in this way does not introduce particle-time drift, but it may perturb the ideal quadrature of our MC estimators. However, clamping is rare in practice: During the violent, large-CFL dam-break stress test (Figure 18, bottom) less than 1% of particle updates required clamping and of those, the maximum clamping amount was less than 20%.

Advancing the Space–Time Position. At the end of the simulation step, $\Delta t_{p,\mathrm{act}}^n$ is fed into the advection operator (Equation (4)) to advance the particle to its new space–time position

$$\begin{aligned} \vec{x}_p^{n+1} &= \mathcal{A}\big(\vec{u}^{n+1}, \vec{x}_p^n, \Delta t_{p,\mathrm{act}}^n\big), \\ t_p^{n+1} &= t_p^n + \Delta t_{p,\mathrm{act}}^n. \end{aligned} \tag{12}$$

Note that our staggered-in-time scheme effectively combines the two necessary advection steps, (i) undoing the previous jitter for stratification/preventing drift and (ii) advancing to the new jittered position into a single standard advection step with a *combined* local advection time $\Delta t^n_{p,\mathrm{act}}$, facilitating seamless integration into existing FLIP/PIC simulation pipelines without any additional particle passes. Although in theory it might be advantageous to use the old velocity field for (i), doing so did not yield any measurable improvement in our experiments.

Monte Carlo Benefits. One way to see the benefit of the proposed scheme is to consider that classical solvers approximate a large time step with a (low-CFL constrained) sequence of *deterministic* "substep grids", i.e., n_{sub} evaluations equally-spaced in time. In contrast, our Monte Carlo sampling advances each particle at a *random* instant t_i inside the large step; the quadrature error is $O(1/\sqrt{N_{\mathrm{ppc}}})$ and, crucially, the $1/\sqrt{N_{\mathrm{ppc}}}$ rate is *independent of the dimension* of the integral [Caflisch 1998], although the variance prefactor may increase. In our setting, we found that extending from 3D to 4D does not require increasing the standard particle count (Section 4.5). By letting the usual eight particles per cell "fan-out" along time, we trade eight times n_{sub} deterministic-in-time samples for eight unbiased 4-D samples (see also Figure 3 and experimental validation in Section 4.5). The fact that our samples are effectively distributed along the spatio-temporally highly correlated characteristics of the velocity field further reduces the estimator's variance.

Beneficial Implications For Particle Management. The improved particle coverage produced by our spatiotemporal sampling removed the need for particle re-seeding or culling in our tests. In all experiments shown, we keep a fixed particle set, except for liquid inflow sources and irreversibly stuck-in-obstacles cases (less than 0.01% of particles per step).

3.6 Space–Time Phase Field from Mass Accumulators

Building on the particle-based phase field used in PF-FLIP, we extend it to a space–time phase field $\phi^{\mathrm{st}}(\mathbf{x}, t)$ defined over the current 4D slab $\Omega \times I_n$. Using the Monte Carlo estimators introduced above, ϕ^{st} can be evaluated at arbitrary $(\mathbf{x}, t)$ within this slab. In the implementation, however, we do not store a full 4D discretization; instead, we evaluate and store only a single slab-collapsed value per face center, denoted $\overline{\phi}_f$. For brevity, we restrict the following definition to the free-surface case $\rho_g = 0$ and refer to Equation (7) in [Braun et al. 2025] for a full two-phase spatial formulation that extends to our spatiotemporal setting by an analogous derivation.

$$\phi^{\mathrm{st}}(\vec{x}, t) := C\left(\frac{\hat{m}(\vec{x}, t)}{\eta_\phi m_0}\right) \in [0, 1]. \tag{13}$$

Here, m_0 is a reference mass for the space–time kernel, defined as the expected value of the mass estimator $\hat{m}$ under a spatio-temporally uniform particle distribution with N_{ppc} particles per cell:

$$m_0 := \mathbb{E}\left[\hat{m}(\vec{x}_f, t_n)\right].$$

In practice, we pre-compute m_0 by applying W_{ST} to a small particle-filled grid patch with $\tau \sim \mathrm{Uniform}(-\frac{1}{2}, \frac{1}{2})$ (averaging the resulting $\hat{m}$ over a few random realizations for stability); this is robust to normalization constants, kernel shape, and discretization details. The parameter η_ϕ (default value 0.5) provides a simple form of interface noise control. Smaller values steepen the phase transition and more aggressively level small wells and bumps caused by uneven particle distributions, similar in effect to "artificial surface tension", whereas larger values preserve finer interface detail at the cost of increased sensitivity to such noise. The function C is a simple saturation/clamping nonlinearity; in all our experiments, we use $C(x) = \min(\sqrt{x}, 1)$ and found the results largely insensitive to the exact choice of C.

Slab-Integrated Phase Field. From the space-time phase field estimator we derive face-centered, slab-integrated estimates of the 3D phase field $\bar{\phi}^n_f = \phi^{\mathrm{st}}(\vec{x}_f, t_n) \in [0, 1]$, as a discrete, smoothed approximation of the discontinuous physical phase indicator $\chi_\ell \in \{0, 1\}$ and the corresponding face densities $\rho(\overline{\phi}_f) = \rho_\ell\,\overline{\phi}_f + \rho_g\,(1 - \overline{\phi}_f)$. As mentioned above, these values are simply the (scaled and clamped) kernel weight accumulators that are computed anyway during the particle-to-grid transfer for normalizing the deposited quantities. While standard hybrid particle–grid methods discard them afterwards, we reuse them to obtain both a phase indicator field and the face coefficients for the variable-density pressure Poisson equation essentially for free, eliminating the need for a per-step surface reconstruction.

Relationship to Classical Phase-Field Methods. Classical Eulerian phase-field methods, typically based on Cahn–Hilliard-type equations [Jacqmin 1999; Jain 2022], evolve a fourth-order PDE and usually impose an interface thickness of several grid cells. As with other Eulerian interface-tracking schemes (VOF, level set), it is nontrivial to maintain strict consistency between the discretizations used for phase/mass and for momentum [Arrufat et al. 2021; Raessi and Pitsch 2012]. In contrast, the particle-based phase field, which we adopt from PF-FLIP and extend to space-time, requires no additional grids or solves and supports sharper interfaces with a small effective thickness of $\epsilon_\phi = 1.5\,\Delta x$. Furthermore, the coupling of the discrete interface indicator and momentum transfer through the same P2G accumulators prevents mass–momentum inconsistencies by construction. The trade-off is FLIP-typical noise, but this is usually acceptable for the kind of turbulent, "splashy" flows often simulated in graphics, especially given the fact that the noise is mostly uncorrelated and thus amenable to standard denoising methods [Bhatacharya et al. 2011; Clarenz et al. 2003; Yu and Turk 2013] (Appendix B).

3.7 Projection as an *Instantaneous* Constraint on the Slab-Integrated Field

We treat pressure as a Lagrange multiplier that enforces incompressibility on the slab-integrated provisional velocity $\vec{u}^\star$ from Equation (9). Specifically, we solve the variable-coefficient *pressure Poisson equation* (PPE)

$$\nabla\cdot(\beta\,\nabla p) \;=\; \frac{1}{\Delta t_n}\,\nabla\cdot(\alpha\vec{u}^\star), \tag{14}$$

with discrete *face coefficients*

$$\beta_f = \frac{\alpha_f}{\max(\rho(\overline{\phi}_f), \epsilon_\rho)}, \tag{15}$$

and update

$$\vec{u}_f^{\,n+1} = \vec{u}_f^{\star} - \Delta t_n \frac{\beta_f}{\alpha_f} (\nabla p)\Big|_f \tag{16}$$

to obtain a divergence-free velocity field at time level t_{n+1}. Here, ϵ_ρ is a small positive constant to guard against division by zero and ill-conditioning of the PPE and $\alpha_f > 0$ are solid face apertures, i.e., the fraction of a cell face that is not covered by solid [Batty et al. 2007; Ng et al. 2009].

In the absence of a sharp geometric surface representation for the liquid–which would be difficult to maintain, or even to define consistently, in our 4D diffuse-interface setting–our derivation does not rely on the commonly used ghost-fluid method (GFM [Kang et al. 2000; Liu et al. 2000]). We instead pose the pressure projection in a volumetric form, following the variational approach proposed by Batty [2010]; Batty et al. [2007] for fluid interfaces. In this view, the projection is a mass-weighted L^2 projection of $\vec{u}^\star$ onto the discrete incompressibility constraint $\nabla\cdot(\alpha\,\vec{u}) = 0$. Concretely, $\vec{u}^{\,n+1}$ minimizes

$$\frac{1}{2}\sum_f \alpha_f\, \rho(\overline{\phi}_f) \left\|\vec{u}_f - \vec{u}_f^{\star}\right\|^2 \tag{17}$$

subject to $\nabla\cdot(\alpha\,\vec{u}) = 0$. The corresponding Lagrange multiplier p is a single, instantaneous (not slab-integrated) pressure field for the step, and the Euler–Lagrange equations yield (14) and (16) with $\beta_f = \alpha_f/\rho(\overline{\phi}_f)$.

Solving the PPE. To numerically solve the pressure Poisson equation in (14), we adapt the approach based on the phase field values. For free-surface simulations, we tag cells as empty (air) if their phase field (as sampled at the cell centers) $\phi_c < 0.5$ and treat them as Dirichlet boundaries with $p = 0$. These cells are removed from the linear system, which only increases the matrix diagonal entries for boundary-adjacent liquid cells while keeping off-diagonal terms homogeneous, preserving the system's symmetry, positive definiteness and conditioning properties. This allows efficient solution using standard preconditioned conjugate gradient methods. For fully two-phase flows, the variable-density formulation leads to an inhomogeneous PPE with potentially severe ill-conditioning due to high density ratios ($\rho_l/\rho_g \gg 1$), necessitating specialized solvers such as adaptive multigrid preconditioners [Braun et al. 2025].

3.8 Choice of Deposition Kernels

We use a separable kernel based on a 1D variant of the *poly6* kernel [Müller et al. 2003] $W_{\text{poly6}}(r) = \max(0, (1-r^2)^3)$ for the spatial 3D weighting function

$$W_S(\vec{r}) = \left(\frac{35}{32}\right)^3 \prod_{d=1}^{3} W_{\text{poly6}}(r_d). \tag{18}$$

This kernel has higher polynomial order while maintaining an evaluation cost comparable to that of a trilinear kernel. We prefer the separable formulation over the SPH-typical spherical support because it better fits our purpose of sampling cubic staggered control volumes on a MAC grid.

For the temporal kernel W_T, we use the one-sided shape function

$$W_T(\tau) = \frac{35}{16}\, W_{\text{poly6}}\left(\tau - \frac{1}{2}\right) \mathbf{1}_{\{\tau\le 1/2\}}, \tag{19}$$

i.e., a half-step shifted and truncated poly6 kernel that peaks at $\tau = \frac{1}{2}$, so that the latest samples within the current slab I_n receive the most weight. Equivalently, W_T is the renormalized *past half* of a symmetric kernel centered at the forward slab boundary $t_{n+\frac{1}{2}}$ (Figure 4). This matches applying the symmetric kernel to the current particle time samples augmented with virtual *time-mirrored* copies across $t_{n+\frac{1}{2}}$ (reflection boundary correction [Silverman 1986]). Although the symmetric completion spans two slabs, the forward half is virtual/mirrored; the effective window remains I_n (Figure 4). Figure 5 shows that using a naive symmetric temporal kernel centered at t_n can introduce oscillatory free-surface artifacts at large CFL, while the one-sided kernel does not. While a formal proof or complete analysis is not currently available, a useful way to understand the choice is phase consistency with end-of-step projection: the solver produces u^{n+1} after projection, so deposition should not over-emphasize older samples within the slab. A symmetric kernel centered at t_n biases deposited interface/momentum toward earlier times, introducing lag which can amplify post-projection oscillations. W_T corresponds to a reflection-boundary-corrected estimator of a forward-centered kernel near $t_{n+1/2}$, aligning the estimator with the forward step and empirically reducing phase error.

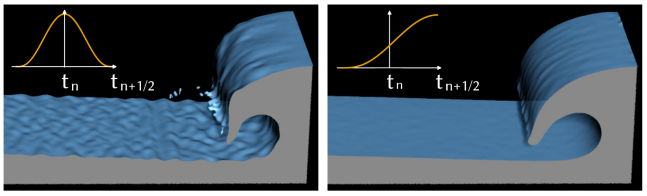


Fig. 5. Naive symmetric vs. one-sided temporal weighting at large CFL. Left: symmetric kernel centered at t_n produces oscillatory surface artifacts. Right: one-sided kernel (Equation (19)) avoids them.

3.9 Further Considerations

Relationship to the Ghost Fluid Method (GFM). As in GFM [Kang et al. 2000; Liu et al. 2000], the liquid-air interface determines the system matrix of the PPE. The difference is how the off-diagonal coefficients of the variable-coefficient Poisson equation are formed: GFM uses geometry-based fractions θ_f (1D-distances) from a sharp interface reconstruction, whereas we use a *volumetric* coefficient (Equation (15)) directly from our time-slab averaged phase field, obtained from the already computed 4D P2G weight accumulators (9). This effectively eliminates the conventional per-step surface reconstruction while at the same time providing an interpretation of interfacial pressure projection in a *volumetric* framework which –unlike sharp surface geometry– naturally extends to 4D.

Slab-Integrated Deposition vs. Viscous Diffusion. Although we integrate deposits over the current time slab, this operation does not introduce additional spatial smoothing beyond the standard transfer kernel W_S, and it is not a recursive temporal filter across simulation steps. We therefore expect it to suppress temporal aliasing artifacts without acting like an added viscosity term. Moreover, using larger global time steps reduces the number of particle-grid transfer cycles per unit physical time, which can reduce accumulated numerical dissipation compared to small-CFL-conforming stepping (Section 4.4).

Grid-to-Particles (G2P). Particle velocities are updated using the projected slab-integrated grid velocity $\vec{u}^{n+1}$ spatially interpolated at the particle's current spatial position (which is associated with its jittered time). Experimentally, we found that additional time interpolation of velocities did not significantly improve visual simulation quality (while requiring storage for an additional velocity field), so in practice, the standard FLIP/PIC G2P operator can be used without modification.

Surface Tension. We employ a standard Continuum Surface Force (CSF) model [Brackbill et al. 1992] to simulate surface tension with interface curvature estimated from a cubic B-spline-smoothed local reconstruction of the phase field. However, surface tension is a millimeter-scale phenomenon while our method is mostly targeted at large-scale simulations. Therefore, surface tension was turned off in our experiments, except for the glugging simulation in Figure 23.

3.10 Basic Algorithm as a Lightweight Extension for Existing Solvers

Algorithm 1 illustrates how our scheme integrates into standard FLIP, PIC, APIC or PF-FLIP solvers by introducing a single new particle attribute and modifying the main advection and P2G deposition loops. APIC integration is straightforward: the standard affine matrix is retained as an additional per-particle attribute, and the mass, momentum, and affine P2G contributions are multiplied by the same W_T. The function Jitter() in line 26 implements the jittering scheme as defined in Equation (10). Deviates ξ_p^n are efficiently generated by using per-thread PRNGs, seeded with a hash over the thread-id, the global step number and a global seed. Because our random deviations affect only the smallest spatio-temporal scales, runs are reproducible given a fixed global seed; across seeds, we observed only negligible differences at small scales.

Integrating ST-FLIP does not require additional linear solves or extra passes over fields: all changes amount to a few additional floating-point operations performed on the fly in loops that already traverse particles and grid nodes, where the relevant data are in cache. In addition to the benefit of enabling larger time steps, integrating ST-FLIP can make the solver also significantly faster per step at the same CFL number (Section 4) because it removes the need for the (often expensive) per-step surface reconstruction (line 17).

Adaptive Jitter Attenuation for Calm Regions. To reduce Monte Carlo noise on locally quiescent free surfaces (Figure 6), we choose the jitter strength $\gamma_p \in [0, 1]$ adaptively per particle based on the *local CFL number* $\|\vec{u}_p\|\Delta t/\Delta x$ as a measure of local flow activity (Algorithm 1 line 25). In theory this requires also adapting the temporal kernel in the deposition stage (line 10) and re-scaling m_0, but we did not find this necessary in practice, because for our one-sided poly6-Kernel W_T, the mean weight under a narrowed distribution produced by $\gamma < 1$ is $\int_{-0.5}^{0.5} W_T(\gamma\tau)\mathrm{d}\tau = (945 + 105\gamma^2 - 21\gamma^4 - 5\gamma^6)/1024 \in [945/1024, 1]$. Hence, the average temporal weighting and ϕ^{st} differ from the $\gamma = 1$ case by at most 7.7% and 3.9%, respectively. Moreover, in the limit $\gamma \to 0$, w_{temporal} collapses to the constant $W_T(0)$ and the normalized face velocity (Equation (9)) reduces to the standard instantaneous P2G transfer (up to a uniform scaling that cancels in the ratio). Thus, adaptive jitter attenuation provides a continuous interpolation between full spatiotemporal slab sampling and classical single-time-level deposition ($\gamma = 0$) while the boundedness result of Appendix A remains valid since $\gamma_p \in [0, 1]$.

Fig. 6. Drop falling into a quiescent pool of water simulated at CFL = 15. Adaptive γ attenuation (right) reduces noise in the calm water surface region while preserving the splash details.

Globally Adaptive Time Stepping. We assume that the base solver uses a standard CFL-based adaptive time step [Bridson 2015; Molemaker et al. 2008], quantizing each step so that the remaining time to the next video frame is always covered by an integer number of equal substeps (lines 4–8). This guarantees that frame boundaries are reached exactly and avoids tiny catch-up steps, while keeping time step changes gradual.

Temporary Re-Synchronization. After finishing a video frame (typically every 3–10 time steps), we reconstruct a high-quality surface directly from the particles (Appendix B). To avoid distortions in the reconstructed surface, we temporarily re-synchronize the time-jittered particle positions to align with the global frame time. This is done on the fly during particle rasterization for rendering only and does not alter the actual simulation state.

4 Results

In this section, we present the experiments conducted to demonstrate and evaluate our method. All tests and simulations were run on a single AMD workstation (Ryzen Threadripper PRO, 4.0 GHz) with 32 cores and 256 GB of RAM. To ensure fair and meaningful comparisons, we

(i) implemented all methods included in comparisons within the same base solver framework, using the open-source MSBG (Multiresolution Sparse Block Grid) library and the adaptive multigrid Poisson solver from Braun et al. [2025];
(ii) enabled adaptive time stepping and locally sub-stepped advection for all methods.

Reported grid resolutions refer to the *effective* resolution of the adaptive/sparse grid discretization (implemented with MSBG). Table 2 summarizes resolutions, particle counts and target CFL numbers of

Algorithm 1: ST-FLIP: Simulating one video frame of length Δt_{frame} with target CFL number $\text{CFL}_{\text{target}}$. Additional steps (beyond the standard FLIP algorithm) are marked (+) in green; removable FLIP steps are marked (-) in orange.

1 **typedef struct** *Particle*
2 float *pos*[3]; float *vel*[3]; float *advectedScalars*[...]
3 (+) float16 δt ▷ *Partial advection time carried over into next step*
4 **while** $\Delta t_{\text{frame}} > 0$ **do**
5 $\Delta t_{\text{prev}} \leftarrow \Delta t$
6 $\Delta t \leftarrow \text{MaxStep}(\Delta t_{\text{frame}}, \text{MaxVelocity}(), \text{CFL}_{\text{target}})$
7 $\Delta t \leftarrow \Delta t_{\text{frame}} / \lceil \Delta t_{\text{frame}} / \Delta t \rceil$ ▷ *subdivide remainder in even parts*
8 $\Delta t_{\text{frame}} \leftarrow \Delta t_{\text{frame}} - \Delta t$
9 **foreach** *particle p* **do** ▷ *P2G: particles-to-grid*
10 (+) $w_{\text{temporal}} \leftarrow W_{\text{T}}(-p.\delta t / \Delta t_{\text{prev}})$ ▷ *Eq.* (19)
11 **foreach** *MAC face center* $\boldsymbol{x}_f$ *in neighborhood of p* **do**
12 $w \leftarrow W_{\text{S}}((\boldsymbol{x}_f - p.pos)/\Delta x)$
13 (+) $w \leftarrow w_{\text{temporal}}\, w$
14 DepositWeightedContribution($\mathbf{x}_f$, p, w)
15 **end**
16 **end**
17 ▷ *Obtain face coefficients for the PPE* (-) $\{\beta_f\} \leftarrow$ ConstructSurfaceForGFM() ▷ *No surface necessary*
18 (+) $\{\beta_f\} \leftarrow$ GetFromP2GWeightAccumulators() ▷ *Eq.* (15)
19 AddExternalForceToVelocity()
20 SolvePressureAndProjectVelocity($\{\beta_f\}$) ▷ *Eqs.* (14),(16)
21 ExtrapolateVelocity()
22 GridToParticles()
23 **foreach** *particle p* **do** ▷ *Advect particles*
24 $\Delta t_{\text{act}} \leftarrow \Delta t$
25 (+) $\gamma \leftarrow \text{smoothstep}(0, 1; \|p.vel\| \Delta t / \Delta x)$ ▷ *local CFL≤1*
26 (+) $\Delta t_{\text{act}} \leftarrow \text{clamp}(\Delta t_{\text{act}} + p.\delta t + \text{Jitter}(\Delta t, \gamma), 0, 2\Delta t)$
27 (+) $p.\delta t \leftarrow \Delta t + p.\delta t - \Delta t_{\text{act}}$ ▷ *Eq.* (11)
28 Advect(p, Δt_{act}) ▷ *Multistep ODE Solver*
29 **end**
30 **end**
31 **foreach** *particle p* **do** ▷ *Rasterize for export/rendering*
32 *posRender* $\leftarrow p.pos$
33 (+) *posRender* $\leftarrow$ Advect(posRender, $p.\delta t$) ▷ *Un-jitter*
34 ParticleToRenderGrid(*posRender*) ▷ *e.g. for SDF construction*
35 **end**

all experiments. Our solver does not inherently rely on GPU acceleration (although any GPU-based solver can be augmented with it via the Algorithm 1 recipe), which is important for our main target use case of very large-scale scenarios whose memory requirements can exceed the limited RAM capacities compared to CPUs.

Methods other than ours use the standard Ghost Fluid Method (GFM) for free-surface treatment. All timings are wall-clock simulation time, excluding rendering. Speedup factors are always reported in terms of total simulation time (or average simulation time per video frame).

Rendering. For rendering we do not use the phase field ϕ but reconstruct an implicit *render surface* $\psi \in [0, 1]$ from the temporarily re-synchronized particles (Algorithm 1, lines 31–34) by splatting a union-of-spheres density onto a grid with twice the resolution of the simulation grid, followed by k_ψ iterations of feature-preserving mean curvature flow (MCF) smoothing [Clarenz et al. 2003; Museth 2014] (see Appendix B for implementation details). We emphasize that this or a similar form of surface smoothing post-processing is also necessary for standard FLIP due to inherent noise in the particle distribution [Bhatacharya et al. 2011; Yu and Turk 2013]. The 0.5 iso-contour of ψ is then rendered by a standard ray marcher. High-curvature areas of the interface are rendered with foam-like scattering coefficients. Apart from this, no post-processing effects or procedural textures are applied.

Parameters. Unless stated otherwise, we used the same values for all method parameters throughout all experiments: Jitter base-strength $\gamma = 1$, phase-transition steepness $\eta_\phi = 0.5$, render-surface regularization iterations $k_\psi = 30$, PIC/FLIP-blend factor $\alpha_{\text{FLIP}} = 0.98$ and $\epsilon = 10^{-4}$ error tolerance for the pressure Poisson solve.

Reported CFL Numbers. We report CFL numbers in terms of the user-specified target CFL number ($\text{CFL}_{\text{target}}$) passed to Algorithm 1, line 6, with adaptive time stepping; for brevity of notation, we refer to $\text{CFL}_{\text{target}}$ interchangeably as "CFL" throughout the paper.

Given $\text{CFL}_{\text{target}}$, the function MaxStep() computes the maximum admissible time step based on an estimate of the current maximum velocity magnitude [Bridson 2015]. At this point, however, the true maximum velocity in the upcoming step is not yet known, so the estimate uses the maximum from the previous step, which can significantly underestimate the actual maximum during splashy intervals of the simulation (Figure 7). The reporting of *target* CFL numbers differs from previous work, which often reports only the global maximum CFL over the entire simulation [Koike et al. 2020; Kugelstadt et al. 2019; Lentine et al. 2012], a metric whose interpretation may tend to overestimate the effective overall speedup because many steps in calmer phases may operate at CFL values substantially smaller than the global maximum.

4.1 Laminar Flow

In the first experiment, we evaluate ST-FLIP with a simple laminar flow arising from the very early phase of the standard dam-break test case [Marrone et al. 2016]. Figure 8 shows the liquid surface at the non-dimensional time $T = t\sqrt{g/H} = 2.3$, shortly before the water front hits the container wall. As the CFL number increases (top to bottom row), standard FLIP exhibits temporal aliasing artifacts and oscillations that manifest as unphysical surface waves. While APIC is more robust against uncorrelated noise than FLIP, it too suffers from temporal aliasing at large time steps. In contrast, ST-FLIP effectively maintains a smooth, artifact-free flow even for very large time steps corresponding to CFL values as high as 20. At very high CFL numbers, the Monte Carlo nature of ST-FLIP does introduce additional surface noise; however, this noise is low-amplitude and largely uncorrelated, making it visually less objectionable and well suited to removal by standard post-processing surface smoothing and denoising, which we investigate in the next section more closely.

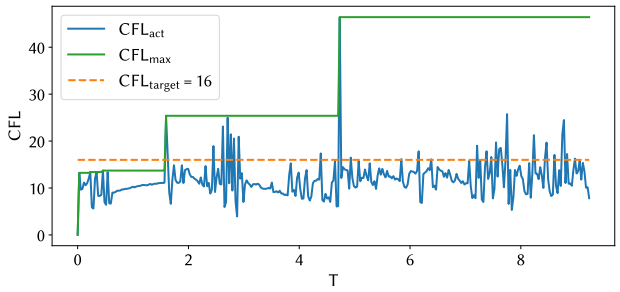


Fig. 7. Actual, maximum and target CFL number over simulation time for the dam-break example (Figure 2). Adaptive time stepping may overshoot the target because the step size Δt must be chosen before the current-step maximum velocity is known. Throughout the paper, we report the prescribed $\mathrm{CFL}_{\mathrm{target}}$ as the meaningful, user-controlled stability parameter rather than the single global maximum CFL over the entire simulation, which can be dominated by brief splashy intervals.

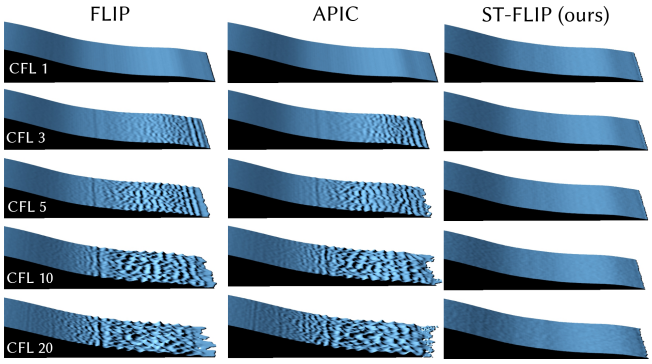


Fig. 8. Dam-break water front, simulated with increasing time steps (top to bottom row). Standard FLIP (left) and APIC (middle) exhibit temporal aliasing artifacts manifesting as unphysical waves, while our method (ST-FLIP, right) maintains a smooth flow, even at very high CFL numbers.

4.2 Influence of Post-Processing Surface Smoothing

To verify that our large-CFL comparisons are not dominated by render-surface smoothing, we vary the number of MCF iterations k_ψ. As shown in Figure 9, ST-FLIP noise at CFL = 20 is largely suppressed already at our default k_ψ=30, while FLIP's CFL = 20 aliasing ripples persist even at k_ψ=100. Figure 10 quantifies this trend using an RMSE on the reconstructed unit normal field against the smoothed CFL = 1 reference: ST-FLIP improves rapidly with smoothing, whereas FLIP remains substantially worse. Together, these results support the interpretation that ST-FLIP trades aliasing-driven, structured free-surface artifacts for predominantly uncorrelated noise that can be removed with standard post-processing, and that our large-CFL improvements are not an artifact of the smoothing stage.

4.3 Standard Dam-Break

Next, we evaluate our method on the fully developed, highly energetic dam-break flow, using the standard setup widely adopted in graphics and CFD as described by Marrone et al. [2016]. Figure 2

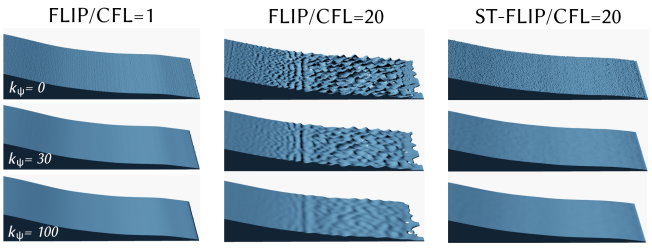


Fig. 9. Effect of MCF surface smoothing as a post-process. Columns show FLIP at CFL = 1 (left), FLIP at CFL = 20 (middle), and ST-FLIP at CFL = 20 (right). Rows (top to bottom) increase smoothing strength k_ψ (0, 30, 100). At CFL = 20, MCF smoothing substantially suppresses the largely uncorrelated Monte Carlo surface noise produced by ST-FLIP, with most noise removed already at k_ψ = 30. In contrast, the structured aliasing artifacts in FLIP persist even under strong smoothing (k_ψ = 100).

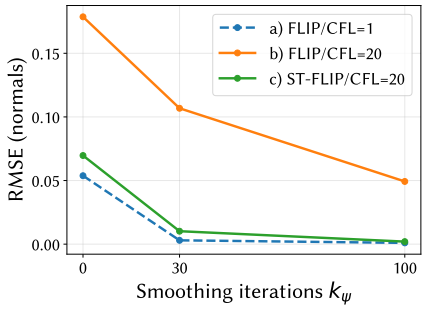


Fig. 10. Quantitative counterpart to Figure 9: surface normal RMSE versus smoothing iterations k_ψ, measured against the smoothed CFL = 1 reference (FLIP, k_ψ = 100).

shows snapshots of the simulation at (non-dimensional time) $T = 8$, for target CFL numbers from 1 to 16 for both FLIP and ST-FLIP. As the CFL number increases, the simulation quality of standard FLIP quickly deteriorates: temporal aliasing and unphysical surface waves (already apparent early in the simulation as shown in Figure 8) destroy important flow features such as the pronounced arc of the main splash-up lip and the hollow region beneath the water surface in the lower right part of the domain. In contrast, ST-FLIP maintains coherent large-scale flow structures and provides visually plausible results even for very large time steps corresponding to CFL = 16. Although ST-FLIP, like any method, cannot completely prevent the loss of quality with increasing time step sizes, the degradation is much less severe and more gradual. Effective speedups relative to the CFL = 1 FLIP-baseline are shown in the upper right of each ST-FLIP snapshot. Increasing the CFL number roughly translates linearly into speedup-gains up until about CFL = 4, after which the gain flattens (mainly due to sub-stepped advection, as discussed in Section 3.3, and additional iterations for a wider velocity extrapolation band). In summary, ST-FLIP achieved speedups of roughly 2x–4x with slight and up to 8x (at CFL=16) with moderate loss of quality vs. the CFL = 1 reference; while standard FLIP exhibits large flow distortions already for CFL=2 and above.

Table 1 breaks down the runtime of an average time step of our method compared to standard FLIP at the same target CFL number of 16. As can be seen, ST-FLIP does not add measurable overhead to the solver's main stages, except for a slightly more expensive advection, which is more than offset by the fact that ST-FLIP time steps do not require a surface reconstruction stage.

Table 1. Per-stage runtime breakdown (in seconds) for an average time step in the simulation of Figure 2. We compare ST-FLIP to standard FLIP at the same target CFL number of 16.

Method	Total	P2G	Surface	Project	Advect	Other
FLIP	2.3	0.5	0.48	0.52	0.52	0.28
ST-FLIP	1.86	0.52	—	0.48	0.6	0.26

For a quantitative assessment, Figure 11 shows the evolution of liquid volume and kinetic energy for the CFL = 16 case: ST-FLIP exhibits lower energy loss and better volume conservation than FLIP, with a volume loss rate of 0.7% per non-dimensional time unit compared with 3.5% for FLIP (orange curves). Unlike FLIP, which resolves only the primary upward splash (first energy peak), ST-FLIP captures multiple subsequent splashes as additional pronounced peaks in the energy curve (marked by arrows on the solid blue curve).

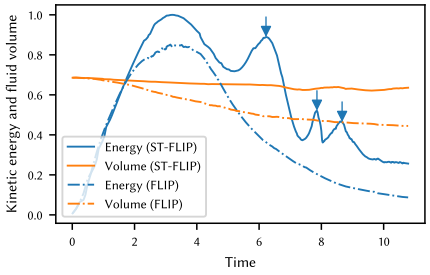


Fig. 11. Time histories of normalized kinetic energy and fluid volume (fraction of domain volume) for the dam-break simulation at CFL = 16, showing that ST-FLIP exhibits lower energy loss and better volume conservation than FLIP. Unlike FLIP, which resolves only the primary splash-up (first peak), ST-FLIP captures multiple subsequent splash-ups as pronounced peaks in the energy curve.

Validation Against a Physical Experiment. To quantitatively validate that ST-FLIP preserves physical fidelity, we simulate the widely used dam-break-with-obstacle setup of Kleefsman et al. [2005], for which water-height time series were measured at several probe locations. Figure 12 compares the water height at sensor H2 from the physical experiment (dashed) to ST-FLIP at increasingly large target CFL numbers (4, 8, 16). ST-FLIP captures the overall evolution and timing of the height signal, including the main impact-induced peaks and subsequent evolution; the first peak is somewhat over-predicted at lower CFL, but the response remains stable and qualitatively consistent as the time step is increased.

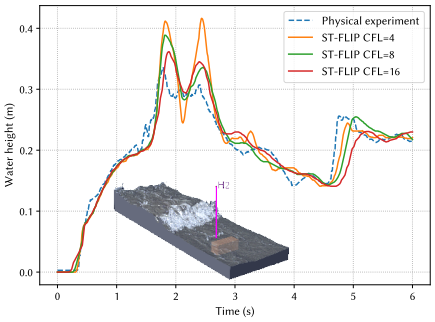


Fig. 12. Dam-break with obstacle benchmark of Kleefsman et al. [2005]: water height at gauge H2 over physical time. Dashed: experimental measurement; solid: ST-FLIP simulation with target CFL = 4, 8, 16. Even at very large time steps, ST-FLIP reproduces the main features and timing of the measured height evolution of the physical experiment.

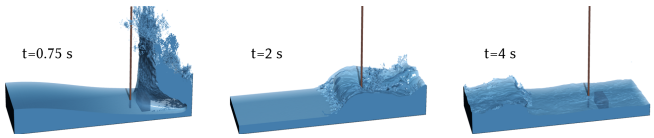


Fig. 13. Snapshots of our simulation of the Kleefsman et al. [2005] experiment at CFL = 16. See Figure 12 for a quantitative assessment (same gauge position at H2)

.

4.4 Time-Slab Integration vs. Viscous Diffusion

As discussed in Section 3.7, W_{T} does not act as a *recursive* temporal filter across simulation steps and should therefore not significantly increase the effective viscosity, i.e., the rate at which small-scale flow features are dissipated by the solver. To assess this, we monitor the evolution of *enstrophy*

$$\mathcal{E} = \frac{1}{2}\int_{\Omega} \|\nabla \times \vec{u}\|^2 \, \mathrm{d}V,$$

defined as half the integral of the squared vorticity over the domain. Enstrophy serves as a sensitive metric for capturing the damping of small-scale vortical flow structures [Donzis et al. 2008]. At equal nominal physical viscosity (assumed zero in our inviscid setting), slower decay of enstrophy is consistent with lower numerical dissipation. Figure 14 shows the evolution of enstrophy over a long-running dam-break simulation for different CFL numbers and different FLIP/PIC blends. The solid curve corresponds to a baseline run with CFL = 1 and $\alpha_{\mathrm{FLIP}} = 0.99$, where $\alpha_{\mathrm{FLIP}} = 1 - \alpha_{\mathrm{PIC}}$ controls the usual mix-in of PIC-style smoothing in FLIP [Bridson 2015]. As shown by the dotted curves, increasing artificial viscosity by decreasing α_{FLIP} leads to a faster loss of enstrophy, as expected. In contrast, increasing the *temporal support* of the slab-integrated deposition by raising the CFL number in ST-FLIP does not produce additional loss of vorticity: the dashed curves (higher CFL, same α_{FLIP}) largely coincide with or slightly improve upon the baseline. We attribute this mild improvement to the fact that larger time steps reduce the number of back-and-forth transfers between particles and grid. Figure 15 qualitatively illustrates vorticity decay in

ST-FLIP by showing 2D slices of the vorticity magnitude at time $T = 60$ for the baseline run (a), the case with increased PIC viscosity $\alpha_{\text{FLIP}} = 0.9$ (b), and the case with increased CFL = 10 at the original α_{FLIP} (c). While enstrophy is not a strict invariant in 3D turbulence, its decay is a practical proxy for the damping of small-scale vortical content and Figure 15 visualizes that this content is not replaced by grid-scale noise.

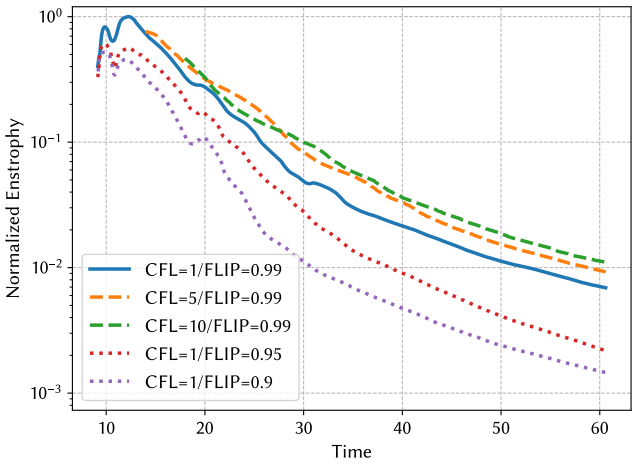


Fig. 14. Increasing the CFL number (dashed) does not increase dissipation of vortical features (no additional enstrophy loss). Dotted: increasing numerical viscosity / numerical diffusion via stronger PIC blending (smaller α_{FLIP}) accelerates enstrophy decay.

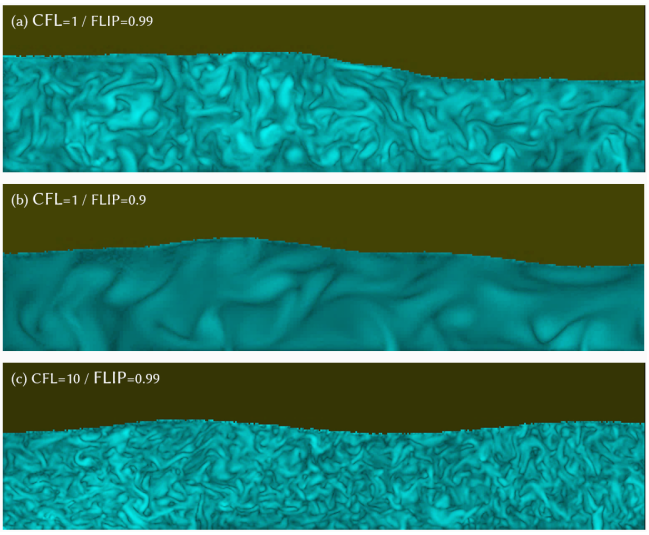


Fig. 15. Vorticity magnitude (2D slice) at T=60: (a) FLIP baseline (CFL = 1, $\alpha_{\text{FLIP}} = 0.99$) (b) increased numerical diffusion via stronger PIC blending ($\alpha_{\text{FLIP}} = 0.9$) suppresses small-scale vortices; (c) ST-FLIP at much larger time steps (CFL = 10) preserves small-scale vortical content comparable to (a), consistent with the enstrophy trends in Figure 14.

4.5 Varying the Number of Particles per Cell

As discussed in Section 3, ST-FLIP should not require significantly more particles (samples) per cell, even though each space–time cell is sampled in an additional (temporal) dimension. To assess this, we ran the dam-break test case for varying number of particles per cell N_{ppc} (from 1 to 16) and compared the resulting free-surface signed distance functions at time $T = 7$ against a high-particle-count ($N_{\text{ppc}} = 50$) reference obtained with the same method and CFL number. Figure 16 shows the resulting RMSE for standard FLIP (CFL = 1) and ST-FLIP (CFL = 10). For both methods, improvement saturates around $N_{\text{ppc}} = 8$, beyond which other errors (e.g., discretization and chaotic divergence of small-scale flow features) dominate. This is consistent with ST-FLIP not requiring significantly more than the standard choice of eight particles per cell.

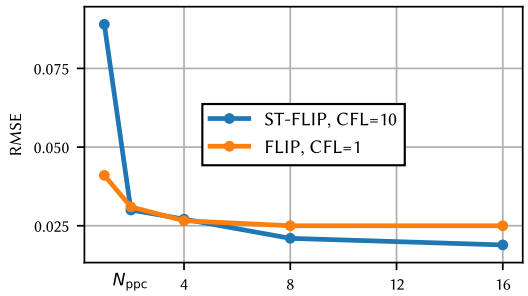


Fig. 16. Simulation quality vs. number of particles per cell. For both FLIP and ST-FLIP, improvement saturates at about the standard value of $N_{\text{ppc}} = 8$, indicating that ST-FLIP does not necessitate significantly more particles per cell although we sample an additional (time) dimension.

4.6 Comparison With APIC and Implicit Density Projection

We next compare ST-FLIP to two widely used modifications of hybrid particle–grid fluid solvers that target common failure modes under large time steps: Affine particle-in-cell (APIC) [Jiang et al. 2015] and Implicit Density Projection (IDP) [Kugelstadt et al. 2019]. APIC improves the particle–grid velocity transfer and typically reduces noise and numerical dissipation. IDP directly regularizes uneven particle distributions, which are a frequent issue at high CFL.

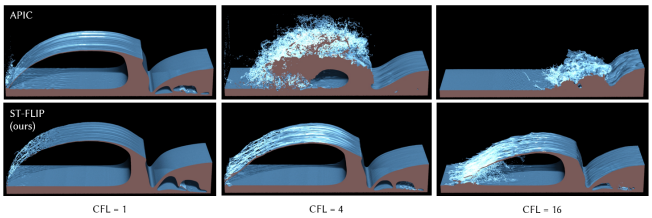


Fig. 17. Dam break simulated for increasing target CFL numbers with APIC (top row) and our method, ST-FLIP (second row). ST-FLIP maintains flow coherence even at very large time steps (CFL = 16) while being more efficient in terms of computation time and memory usage (see Figure 18 for performance metrics).

Figure 17 compares ST-FLIP with APIC for the standard 3D dam-break benchmark at target CFLs of 1, 4, and 16. In Figure 18 we

include IDP and the combination of IDP and APIC in the comparison at a target CFL of 16, compared against a high-fidelity reference computed with APIC at CFL = 1 shown in the first row. For each method the figure visualizes a 2D slice of the 3D deposited mass-density field (after particle-to-grid transfer) at the same simulation time $T = 7$, and reports wall-clock simulation time per video frame $\Delta t_{\text{frame}} = 0.1$ (excluding rendering), as well as peak memory footprint. At CFL = 16, standard FLIP exhibits severe degradation of simulation quality with highly uneven particle density distribution, failing to reproduce the characteristic splash-up arc and the cavity beneath the free surface. IDP and APIC+IDP noticeably improve small-scale density noise but still cannot reproduce the dominant flow features. All three prior methods also incur additional computational overhead compared to ST-FLIP: per-step surface reconstruction (FLIP, APIC, IDP); storing a 3x3 matrix per particle (APIC) or solving an additional Poisson equation (IDP). In contrast, ST-FLIP preserves the main flow structures at CFL = 16 much closer to the CFL = 1 reference, while being an order of magnitude faster and using ≈ 40% less memory than the APIC reference run. Compared to standard FLIP, ST-FLIP adds one half-precision particle scalar (δt) and no extra grids. As reported in Figure 18 this increased the peak memory footprint from 350 MB to 363 MB, i.e., by only 3.7%.

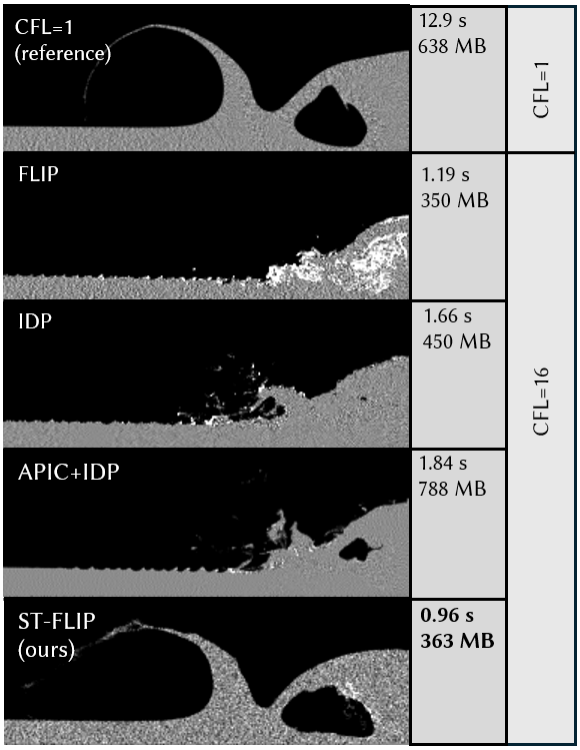


Fig. 18. 3D dam-break comparison at CFL = 16 (top: APIC/CFL = 1 reference). Each row shows a 2D slice of the 3D mass-density field, along with the corresponding simulation time per video frame (s) and peak memory footprint (MB). From top to bottom: CFL = 1 APIC reference, standard FLIP, IDP, APIC+IDP, and our ST-FLIP (all CFL = 16 except the reference). Standard FLIP exhibits severe degradation of the flow field and fails to reproduce the characteristic splash-up arc; IDP and APIC+IDP, while greatly suppressing density noise, increase surface-noise and still miss the main splash and large cavity. ST-FLIP preserves these features while being more than ten times faster. Also, APIC and IDP consume more memory due to per-particle matrix storage or auxiliary pressure gradient fields.

4.7 Comparison With Naive Jittering

A natural question is whether the large-CFL benefits of ST-FLIP can be obtained by a simpler modification that merely randomizes the per-particle advection time. To test this, we implemented a *naive temporal jittering* baseline in which each particle is advected with a randomly perturbed step size,

$$\Delta t^n_{p,\text{naive}} = \text{clamp}\Big(\Delta t_n + \gamma\, \tau^n_p\, \Delta t_n,\ 0,\ 2\Delta t_n\Big), \qquad \tau^n_p \sim \mathcal{U}\big(\big[-\tfrac{1}{2}, \tfrac{1}{2}\big]\big),$$

This baseline corresponds to using only the stochastic term of Equation (11) but omitting both (i) the residual carryover that defines a consistent particle time state, and (ii) the temporal kernel W_T in the deposition step. Figure 19 compares naive jittering to ST-FLIP in the dam-break benchmark at target CFL = 10. Although randomizing advection times breaks the strict regularity of particle trajectories, it does *not* yield a consistent space–time sampling of the step interval: particles are effectively advanced to different times, but these time offsets are neither tracked nor accounted for during deposition and projection. The mismatch between the particle state and the single-time-level grid solve can also accumulate over steps (a random-walk drift of the implicit particle time), producing locally inconsistent deposited mass/momentum. This manifests as pronounced high-frequency roughness and "grainy" surface artifacts (Figure 19a).

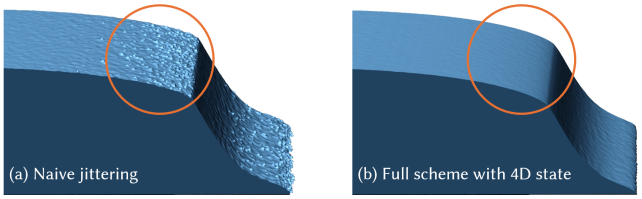


Fig. 19. Naive temporal jittering by randomizing the local per-particle advection time leads to marked artifacts due to temporal particle inconsistencies (a) which are avoided by our scheme that is based on a true spatiotemporal (4D) particle state (b). Shown is the free surface, early in the dam-break experiment at CFL = 10 for both methods.

4.8 Further Simulation Results

In this section, we present a range of liquid simulations that highlight the robustness, performance and versatility of our solver.

Turbulent Flow Around Solid Obstacle. In this test, we consider a setup designed to generate a turbulent wake behind a static obstacle. We again start from the standard 3D dam-break configuration of Marrone et al. [2016] and place a rectangular obstacle in the middle of the domain, so that the advancing sheet of water breaks around the obstacle and produces a highly vortical wake. Figure 20 compares FLIP and ST-FLIP for this scenario. The reference solution (Figure 20a) uses standard FLIP at CFL = 1 and exhibits a rich turbulent wake behind the obstacle. Increasing the time step to CFL = 20 with standard FLIP (b) leads to severe temporal aliasing: the wake structure is largely destroyed and replaced by noise and unphysical waves. In contrast, ST-FLIP at CFL = 20 (c) preserves the turbulent

wake and overall flow structure, producing results that are visually comparable to the CFL = 1 reference while speeding up total simulation time roughly by a factor of eight, consistent with the speedup of the dambreak case (Figure 2).

High-Speed Jet Impinging on a Thin Obstacle. A fundamental difficulty for schemes operating at CFL $\gg 1$ arises in the presence of thin obstacles, i.e., obstacles whose thickness is much smaller than the distance traveled by the liquid during a single time step. In a naive implementation, liquid may tunnel through a wall only a couple of cells thick when the leading edge of the flow advances, for example, 15 grid cells per time step. ST-FLIP nevertheless handles this case robustly because (i) locally sub-stepped advection (Section 3.3) maintains collision detection at $\mathrm{CFL}_{\mathrm{local}} = 1$, and (ii) spatio-temporal sampling distributes fluid samples over space-time (Fig. 3), thereby preserving a usable near-wall velocity field while free-slip/no-through pressure projection enforces the correct boundary conditions on the grid. Consequently, after collision detection and standard out-of-obstacle push-back, particles follow the obstacle surface rather than becoming trapped or accumulating. To evaluate this behavior, we simulated a fast water jet at CFL = 16 impinging on a plate only one grid cell thick and observed neither particle tunneling nor particle accumulation, as shown in Figure 21.

Large Whirlpool with Gravity Waves. To further broaden the set of flow regimes beyond standard benchmarks (e.g., dam breaks), we simulate a large-scale ocean whirlpool at high resolution. The domain is a $200 \times 200 \times 80$ m^3 body of water with a cylindrical outflow pipe (length 10 m, diameter 20 m) centered on the bottom boundary. The initial velocity field $\vec{u}(\vec{x}) = (0, 0, \omega_0) \times \vec{x}$ is prescribed as a solid-body rotation about the vertical axis through the origin with angular speed $\omega_0 = 0.1$ rad s^{-1}. Due to interactions with the planar side walls of the box-shaped domain, the flow transitions into a complex superposition of a spiraling vortical outflow and large gravity waves, most clearly visible in the accompanying video.

Figure 22 shows snapshots from a 512^3 simulation and compares different time-step sizes. FLIP at CFL = 3 clearly resolves the vortical and wave motion but already shows small, unphysical ripples on the free surface (left). At CFL = 30, the aliasing ripple artifacts largely destroy any details and the structure of the overall flow (middle). In contrast, ST-FLIP at CFL = 30 avoids these artifacts and preserves the flow structures, while using ten times larger time steps (right).

Finally, Figure 1 (left) shows a snapshot of the whirlpool simulation at an effective grid resolution of $3072 \times 3072 \times 2048$ with more than two billion particles. Using our method at a target CFL = 15, the simulation completed in about 3.4 days. By comparison, the smaller time steps typical of prior FLIP/PIC-based free-surface methods (CFL $\approx$ 1–3) would increase the runtime several-fold, pushing the same setup into a ∼2-week range and rendering it impractical to run at this scale.

4.9 Two-Phase Simulations: ST-FLIP Integrated Into PF-FLIP

So far, our evaluation focused on single-phase free-surface flows. We now turn to *two-phase* simulations with high density contrast (liquid–air), which are substantially more challenging due to the strong coupling between phases and the need to preserve a visually plausible interface under large-scale deformation. To test ST-FLIP in this more demanding setting, we integrate it into PF-FLIP using Algorithm 1. PF-FLIP is a highly optimized two-phase FLIP framework for extreme-scale, spatially adaptive high-resolution simulations and has been demonstrated on multi-billion particle scenarios. It thus serves as a strong baseline as it also operates with relatively large time steps (CFL numbers up to ∼ 3–4) using globally adaptive time stepping and locally adaptive sub-stepped advection.

Glugging/Rising Bubbles. We first demonstrate the feasibility of ST-FLIP as a two-phase solver by running a canonical two-phase test, i.e., the "glugging" effect of water pouring through a spout that cannot be reproduced with standard single-phase liquid simulations. For this test, we simulated two containers, each with a square bottom of side length $0.5L$, connected by a cylinder of radius and length $0.05L$, with the initial water level (relative to the box bottom) in the upper container at H=0.5L, where $L = 1$ m is the domain size. Figure 23 shows snapshots for the same simulation time at increasingly larger CFL numbers. ST-FLIP remains roughly faithful to the CFL = 1 reference even for large CFL numbers. Note that the details of the outflow and the rising disintegrating bubble are governed by turbulent, hence chaotic flow so fine-scale breakup differs, but the macroscopic glugging pattern is preserved.

Large-Scale Dam Break. Next, we simulate a very large-scale dam break (domain size is 400 meters, simulated with 3 billion particles). At such scales and resulting high velocities, aerodynamic forces become an important factor, necessitating two-phase simulation in order to obtain a velocity field not only in the water but also in the air to drive mist and spray droplet dynamics for realistic, physics-based white water effects [Braun et al. 2025], e.g., the explosive water-impact spray plumes in Figure 24. We used the same setup as described in Figure 19 in the aforementioned paper and compared our method to PF-FLIP at CFL = 15. As can be seen in Figure 24, PF-FLIP at such high CFL numbers no longer reproduces the main splash-up spray plume when the water hits the opposite wall of the container. In contrast, ST-FLIP maintains the structure of the flow with only moderate loss of quality compared to the CFL = 3 (PF-FLIP) baseline (leftmost image), while finishing the simulation 3.1 times faster than the reference.

Large-Scale Dam Discharge. We re-simulated the *large dam discharge* scene (Figure 3 in [Braun et al. 2025]) both with basic PF-FLIP (Figure 25, top) and with ST-FLIP (Figure 25, bottom). With ST-FLIP, we are able to more than double the already high CFL number of the PF-FLIP baseline without noticeable loss of visual quality. This translates into an approximately 2× reduction in per-frame simulation time at comparable visual fidelity. Achieving such a speedup on top of an already highly optimized, large-step two-phase solver underscores that ST-FLIP provides a substantial and largely orthogonal performance benefit, while remaining compatible with production-scale two-phase simulation pipelines. Photograph © Imaginechina Ltd. / Alamy.

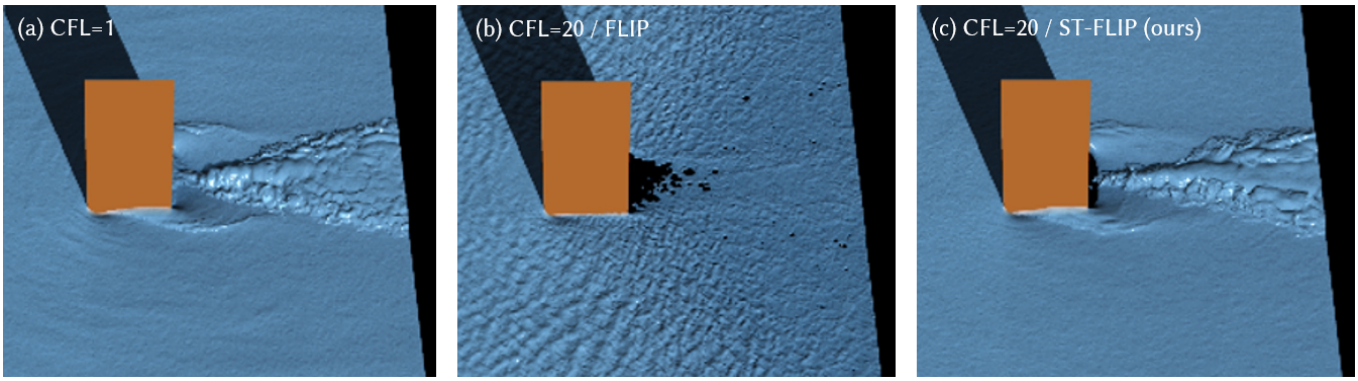


Fig. 20. Turbulent flow around obstacle. (a) FLIP at CFL = 1 (reference); (b) FLIP at CFL = 20, where temporal aliasing severely degrades the flow structure; (c) ST-FLIP at CFL = 20, which preserves the turbulent wake structure and yields results visually comparable to the reference, despite much larger time steps (8x speedup over the CFL = 1 reference).

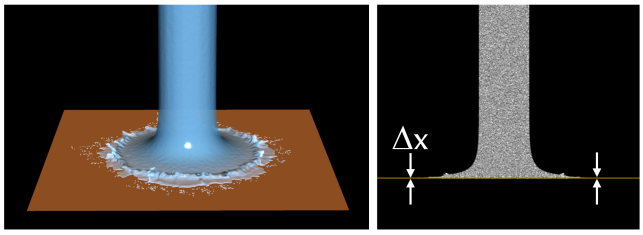


Fig. 21. Fast liquid jet hits a thin obstacle (left). 2D slice of particle density (right). ST-FLIP handles this at high CFL = 16 and a small obstacle thickness of Δx without particles "tunneling" through or "piling up" on the obstacle plate.

Table 2. Resolution and target CFL summary for our simulations

Figure	Effective resolution	Particles	Target CFL
1 (left)	3072×3072×2048	2.1 B	15
2	768×256×192	62 M	1–16
8	384×128×96	8 M	1–20
12	640×224×120	48 M	4–16
18,17	384×112×96	8 M	1–16
20	768×768×288	200 M	1–20
22	512×512×512	40 M	3–30
23	128×256×128	31 M	1–16
24	1600×1600×1600	3.1 B	3–15
25	2048×1024×1024	1.2 B	3–7

5 Limitations

While our method offers significant advantages for large-scale liquid and two-phase simulations, it is not without limitations. In particular, ST-FLIP, like any method, cannot completely prevent a loss of quality as the time step size increases. However, the degradation is much less severe and more gradual than with previous methods (Figure 2).

The FLIP-typical time-step-dependent loss of energy and angular momentum induced by the standard first-order time-splitting scheme is not addressed by our method. Nevertheless, ST-FLIP is, in principle, compatible with common countermeasures (e.g., advection-reflection [Zehnder et al. 2018], IVOCK [Zhang et al. 2015], or vorticity confinement) that can be applied to mitigate this issue.

Like other Monte Carlo integration-based approaches, our scheme can introduce significant amounts of noise, in particular at very high CFL numbers. This is often acceptable for the turbulent, naturally chaotic flows commonly simulated in graphics, especially given that the noise is mostly low amplitude and uncorrelated and thus amenable to standard surface denoising methods as shown in Section 4.2. This is analogous to how Monte Carlo methods in rendering trade integration speed for noise that can be reduced in post-processing.

This limitation is most apparent for calm water surfaces, where the ratio of signal to FLIP-noise is low and large time steps can amplify irregularities in the particle distribution. In our current implementation, we mitigate this by adaptively attenuating the jitter strength in regions with lower local flow velocity, but more sophisticated adaptivity and noise suppression schemes could be a worthwhile direction for future research.

Finally, ST-FLIP is less effective for surface-tension-dominated (i.e., very small-scale) flows, because surface tension imposes an additional restriction on the time step $\Delta t < O(\Delta x^{3/2})$ [Brackbill et al. 1992], limiting the potential for enlarging the velocity-based CFL number.

6 Conclusions and Outlook

We presented ST-FLIP, a spatiotemporal extension of hybrid particle–grid liquid solvers that targets free-surface and two-phase flows with very large time steps. The central observation behind our method is that many large-CFL artifacts in PIC/FLIP/APIC pipelines are fundamentally spatiotemporal sampling problems: when particles traverse multiple cells per step, instantaneous particle-to-grid deposition leaves gaps in space–time coverage, which manifests as temporal aliasing and unphysical surface waves after projection. ST-FLIP addresses this issue by treating particles as Monte Carlo samples in four-dimensional space-time. Concretely, we augment each particle with a time offset within the current step and perform particle-to-grid transfers using a carefully designed, separable space–time kernel. This produces a Monte Carlo estimator of per-step *time*

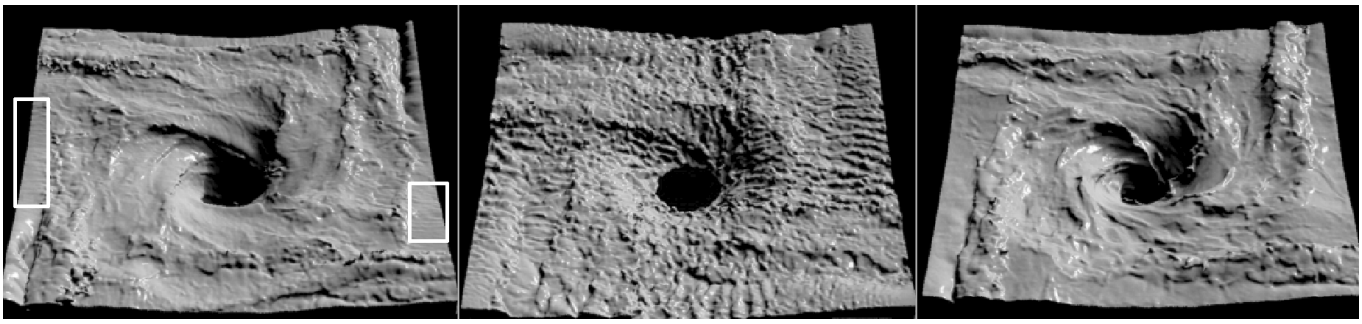


Fig. 22. Whirlpool simulation. Left: FLIP CFL = 3 resolves the flow well but exhibits small unphysical surface ripples, Middle: FLIP at CFL = 30 largely degrades any flow features. Note that the surface ripples are not real waves but unphysical aliasing artifacts, Right: Our method, ST-FLIP at CFL = 30 preserves the flow without artifacts, despite using ten times larger time steps with an effective per-frame speedup of approximately a factor of eight.

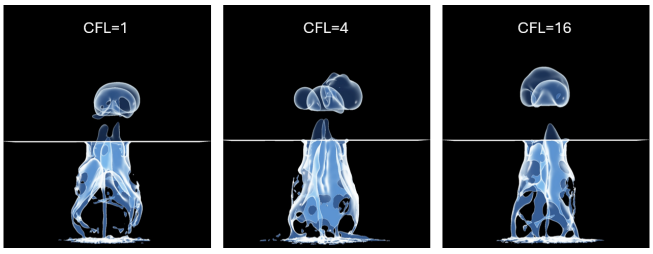


Fig. 23. Glugging simulation at increasingly larger CFL numbers. Left: CFL = 1 PF-FLIP reference. Middle and right: ST-FLIP at CFL = 4 and CFL = 16 respectively.

slab-integrated mass and momentum on the grid, effectively acting as temporal anti-aliasing while retaining the low-dissipation character of FLIP-style updates.

A second key benefit of the spatiotemporal formulation is an interface representation that is diffuse in both space and time. Building on recent particle-based phase field ideas, we reuse the spatiotemporal P2G weight accumulators as a space–time phase field, which provides variable-coefficient projection weights and removes the need for explicit per-step surface reconstruction. When a sharp surface is needed for export or final rendering, we reconstruct it directly from the particles only at output frames.

Importantly, ST-FLIP is designed as an attractively lightweight extension: it introduces only one new particle attribute and does not require additional grids, additional linear solves, or extra passes over particles/fields. In practice, this keeps the runtime dominated by the same components as the underlying solver (advection, transfers, and pressure projection), while enabling substantially larger stable and visually plausible time steps. Across our benchmarks and production-style scenarios, we observed multi-fold reductions in wall-clock time for target CFL numbers up to 30, with speedups of roughly 2×–8× at high effective 3D resolutions. Finally, integrating ST-FLIP into the already fast and highly optimized PF-FLIP two-phase solver yielded a same-quality speedup of about a factor of two in a large-scale, high-resolution two-phase setting, indicating that the approach can benefit strong baselines as well.

Outlook. ST-FLIP suggests several promising directions for future work. A natural next step is to reduce the variance of the temporal Monte Carlo estimator: replacing pseudo-random temporal jitter with low-discrepancy sequences (e.g., Halton or Sobol) together with suitable scrambling could improve temporal coherence and further suppress residual noise at fixed particle counts.

Another direction is to enrich the temporal representation. While our current implementation collapses the conceptual space–time slab of a step's time interval into a single step-integrated 3D grid, retaining a small number of distinct time levels as separate 3D grids could enable higher-order temporal reconstruction and potentially further reduce estimator noise.

More broadly, the underlying spatiotemporal particle–grid viewpoint is not specific to liquids and may translate to other particle or hybrid methods (e.g., MPM, SPH), where uneven sampling or large time steps similarly affect stability, accuracy, and visual quality.

References

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, 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.

Christopher Batty. 2010. *Simulating Viscous Incompressible Fluids with Embedded Boundary Finite Difference Methods.* Ph. D. Dissertation. The University of British Columbia, Vancouver.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 100–es. doi:10.1145/1275808.1276502

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.

C. K. Birdsall and A. B. Langdon. 2005. *Plasma Physics via Computer Simulation.* Institute of Physics Publishing, Bristol and Philadelphia. doi:10.1887/0750300255 Reprint of the 1985 McGraw-Hill edition, with corrections and new material.

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.

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, Article 42 (July 2025), 23 pages. doi:10.1145/3730854

Robert Bridson. 2008. *Fluid Simulation for Computer Graphics.* A. K. Peters, Ltd., USA.

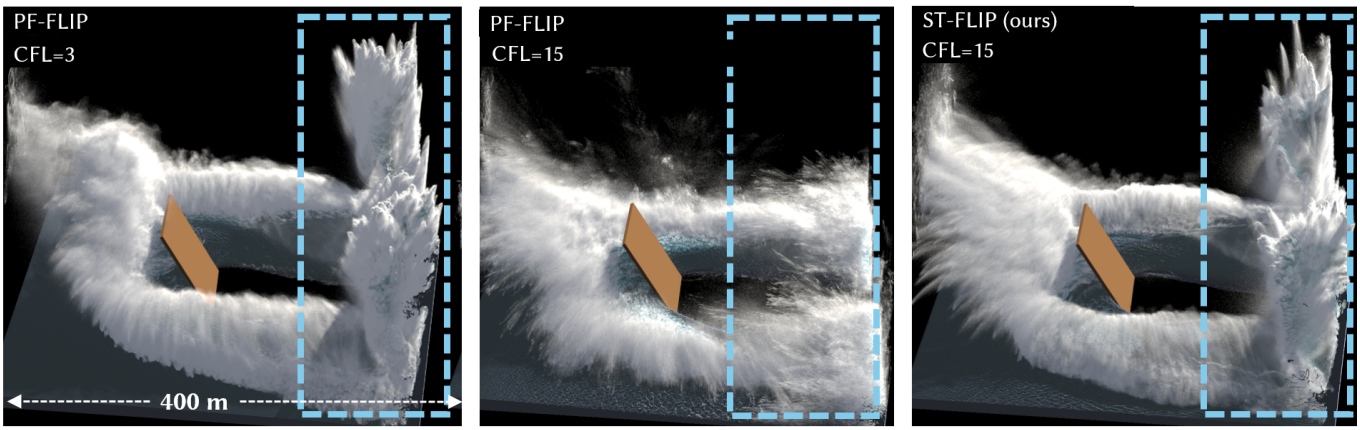


Fig. 24. Two-Phase simulation of a large-scale dam break with obstacle (domain size 400 meters). At CFL = 15, PF-FLIP (middle) no longer reproduces the main impact splash-up spray plumes, whereas ST-FLIP only moderately loses quality in the details but maintains the global structure of the flow.

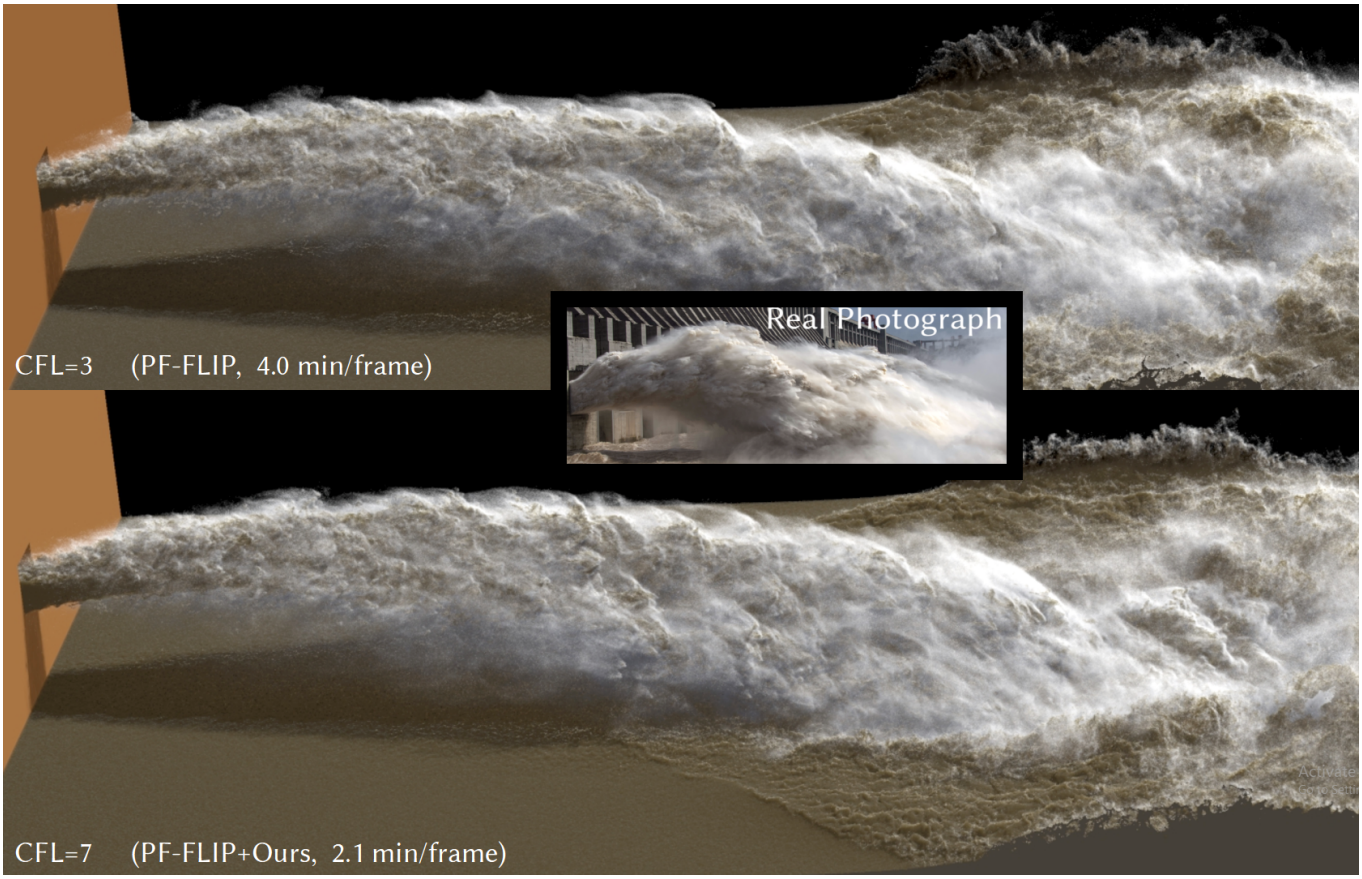


Fig. 25. Large-scale two-phase flow: Dam discharge using the same setup as Braun et al. [2025] at high effective resolution of 2048x1024x1024. Top: original PF-FLIP method with CFL = 3. Bottom: Augmented with our spatiotemporal extension, the same simulation can be run at more than twice the time step size without visible loss of quality, effectively doubling the simulation speed of the already fast, highly optimized PF-FLIP solver. Photograph © Imaginechina Ltd. / Alamy.

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

Russel E. Caflisch. 1998. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica* 7 (1998), 1–49. doi:10.1017/S0962492900002804

Duowen Chen, Zhiqi Li, Taiyuan Zhang, Jinjin He, Junwei Zhou, Bart van Bloemen Waanders, and Bo Zhu. 2025. Fluid Simulation on Compressible Flow Maps. *ACM Transactions on Graphics* 44, 4 (2025), to appear. to appear.

Nuttapong Chentanez and Matthias Müller. 2012. Mass-conserving eulerian liquid simulation. In *Proceedings of the 11th ACM SIGGRAPH / Eurographics Conference on Computer Animation* (Lausanne, Switzerland) *(EUROSCA'12)*. Eurographics Association, Goslar, DEU, 245–254.

Ulrich Clarenz, Gerhard Dziuk, and Martin Rumpf. 2003. On generalized mean curvature flow in surface processing. In *Geometric analysis and nonlinear partial differential equations*. Springer, Berlin, Heidelberg, 217–248.

Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed Ray Tracing. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '84)*. Association for Computing Machinery, New York, NY, USA, 137–145. doi:10.1145/800031.808590

Richard Courant, K. Friedrichs, and Hans Lewy. 1928. Über die partiellen Differenzengleichungen der mathematischen Physik. *Math. Ann.* 100 (1928), 32–74.

DA Donzis, PK Yeung, and KR Sreenivasan. 2008. Dissipation and enstrophy in isotropic turbulence: resolution effects and scaling in direct numerical simulations. *Physics of Fluids* 20, 4 (2008), –.

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.

Francis H Harlow. 1964. The particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys.* 3 (1964), 319–343.

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.

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.

Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. 2014. SPH fluids in computer graphics. *Computer Graphics Forum* 33, 1 (2014), 155–188.

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.

James T. Kajiya. 1986. The Rendering Equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. Association for Computing Machinery, New York, NY, USA, 143–150. doi:10.1145/15922.15902

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.

KMT Kleefsman, G Fekken, AEP Veldman, B Iwanowski, and B Buchner. 2005. A volume-of-fluid based simulation method for wave impact problems. *Journal of computational physics* 206, 1 (2005), 363–393.

Tatsuya Koike, Shigeo Morishima, and Ryoichi Ando. 2020. Asynchronous Eulerian liquid simulation. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, Wiley, Hoboken, NJ, 1–8.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2022. A Survey on SPH Methods in Computer Graphics. *Computer Graphics Forum* 41, 2 (2022), 737–760. doi:10.1111/cgf.14508

Tassilo Kugelstadt, Andreas Longva, Nils Thuerey, and Jan Bender. 2019. Implicit density projection for volume conserving liquids. *IEEE Transactions on Visualization and Computer Graphics* 27, 4 (2019), 2385–2395.

Michael Lentine, Mridul Aanjaneya, and Ronald Fedkiw. 2011. Mass and momentum conservation for fluid simulation. 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, 91–100. doi:10.1145/2019406.2019419

Michael Lentine, Matthew Cong, Saket Patkar, and Ronald Fedkiw. 2012. Simulating free surface flow with very large time steps. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Goslar, DEU, 107–116.

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.

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.

Salvatore Marrone, Andrea Colagrossi, Andrea Di Mascio, and David Le Touzé. 2016. Analysis of free-surface flows through energy considerations: Single-phase versus two-phase modeling. *Physical Review E* 93, 5 (2016), 053113.

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.

Jeroen Molemaker, Jonathan M Cohen, Sanjit Patel, Jonyong Noh, et al. 2008. Low Viscosity Flow Simulations for Animation.. In *Symposium on Computer Animation*, Vol. 9. ACM, New York, NY, USA, 18.

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.

Matthias Müller. 2009. Fast and robust tracking of fluid surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, New York, NY, USA, 237–245.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, San Diego, CA, USA, 154–159.

Ken Museth. 2014. A Flexible Image Processing Approach to the Surfacing of Particle-Based Fluid Animation (Invited Talk). In *Mathematical Progress in Expressive Image Synthesis I*, Ken Anjyo (Ed.). Mathematics for Industry, Vol. 4. Springer, Tokyo, 81–84. doi:10.1007/978-4-431-55007-5_11

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.

Yen Ting Ng, Chohong Min, and Frédéric Gibou. 2009. An efficient fluid–solid coupling algorithm for single-phase flows. *J. Comput. Phys.* 228, 23 (2009), 8807–8829.

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.

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.

Bo Ren, Xu-Yun Yang, Ming C Lin, Nils Thuerey, Matthias Teschner, and Chenfeng Li. 2018. Visual simulation of multiple fluids in computer graphics: A state-of-the-art report. *Journal of Computer Science and Technology* 33, 3 (2018), 431–451.

Sergio Sancho, Jingwei Tang, Christopher Batty, and Vinicius C. Azevedo. 2024. The Impulse Particle-In-Cell Method. *Computer Graphics Forum* 43, 2 (2024), e15022. doi:10.1111/cgf.15022

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.

Side-Effects-Software. 2024. Houdini. – –, – (2024), –.

B. W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London. https://nla.gov.au/nla.cat-vn375680 Accessed: 2026-01-09. Via National Library of Australia.

Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. doi:10.1145/311535.311548

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.

John Villasenor and Oscar Buneman. 1992. Rigorous charge conservation for local electromagnetic field solvers. *Computer Physics Communications* 69, 2-3 (1992), 306–316.

Haoxiang Wang, Kui Wu, Hui Qiao, Mathieu Desbrun, and Wei Li. 2025a. Kinetic Free-Surface Flows and Foams with Sharp Interfaces. *ACM Transactions on Graphics* to appear, to appear (2025), to appear.

Sinan Wang, Junwei Zhou, Fan Feng, Zhiqi Li, Yuchen Sun, Duowen Chen, Greg Turk, and Bo Zhu. 2025b. Fluid Simulation on Vortex Particle Flow Maps. *ACM Trans. Graph.* 44, 4, Article 91 (July 2025), 24 pages. doi:10.1145/3731198

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.

Ross T Whitaker. 2002. Isosurfaces and level-set surface models. *School of Computing, University of Utah* N/A, N/A (2002), –.

Jihun Yu and Greg Turk. 2013. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics (TOG)* 32, 1 (2013), 1–12.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–8.

Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–8.

Junwei Zhou, Duowen Chen, Molin Deng, Yitong Deng, Yuchen Sun, Sinan Wang, Shiying Xiong, and Bo Zhu. 2024. Eulerian-Lagrangian Fluid Simulation on Particle Flow Maps. *ACM Trans. Graph.* 43, 4, Article 76 (July 2024), 20 pages. doi:10.1145/3658180

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

A Boundedness of the Particle Time Deviation

Recall Equation (11):

$$\Delta t^n_{p,\mathrm{act}} = \mathrm{clamp}\Big(\Delta t_n + \delta t^n_p + \gamma\, \xi^n_p\, \Delta t_n, 0, 2\Delta t_n\Big),\ \ \delta t^{n+1}_p = \Delta t_n + \delta t^n_p - \Delta t^n_{p,\mathrm{act}}$$

with $\xi^n_p \in [-\frac{1}{2}, \frac{1}{2}]$ and $\gamma \in [0,1]$. Let $t_{n+1} = t_n + \Delta t_n$ and $t^{n+1}_p = t^n_p + \Delta t^n_{p,\mathrm{act}}$, $t^0_p = t_0$. By telescoping the residual update,

$$\delta t^n_p = \sum_{k=0}^{n-1} (\Delta t_k - \Delta t^k_{p,\mathrm{act}}) = t_n - t^n_p. \tag{20}$$

Thus bounding $|\delta t^n_p|$ directly bounds the particle/global time mismatch.

Lemma A.1 (Uniform bound; no random walk drift). *Assume the global CFL bound $\Delta t_n \le \Delta t_{\max}$ for all n. Then for all $n \ge 0$,*

$$|\delta t^n_p| \le \frac{\Delta t_{\max}}{2} \qquad \textit{and hence} \qquad |t^n_p - t_n| \le \frac{\Delta t_{\max}}{2}.$$

Proof. Set $H := \Delta t_{\max}/2$ and suppose $|\delta t^n_p| \le H$. Write $h := \Delta t_n$, $r := \delta t^n_p$, $\xi := \xi^n_p$, and $x := h + r + \gamma\, \xi\, h$. Let $a := \mathrm{clamp}(x, 0, 2h)$, so $r^+ := \delta t^{n+1}_p = h + r - a$. There are three cases:

(i) $0 \le x \le 2h$: $a = x$, hence $r^+ = -\gamma\, \xi\, h$, so $|r^+| \le h/2 \le H$.

(ii) $x < 0$: $a = 0$, hence $r^+ = h + r$. From $x < 0$ we get $r < -h(1 + \gamma\xi) \le -h/2$, so $r^+ \le h/2 \le H$. Also $r \ge -H$ and $h \le 2H$ imply $r^+ = h + r \ge -H$.

(iii) $x > 2h$: $a = 2h$, hence $r^+ = r - h$. From $x > 2h$ we get $r > h(1 - \gamma\xi) \ge h/2$, so $r^+ \ge -h/2 \ge -H$. Also $r \le H$ implies $r^+ = r - h \le H$.

Thus $r^+ \in [-H, H]$. Since $\delta t^0_p = 0$, induction yields $|\delta t^n_p| \le H$ for all n. The bound on $|t^n_p - t_n|$ follows from (20). □

B Surface Reconstruction for Rendering

For rendering, we reconstruct an implicit surface as the 0.5-isocontour of a field $\psi \in [0,1]$ directly from the particles by splatting a union-of-spheres (radius $0.5\Delta x$) density ψ_0 onto a grid with twice the resolution of the simulation grid, followed by k_ψ narrow-band iterations of a feature-preserving mean curvature flow in the standard level-set formulation

$$\psi^{n+1} = \psi^n + \Delta\tau\, g\, |\nabla\psi^n|\, \nabla \cdot \left(\frac{\nabla\psi^n}{|\nabla\psi^n|}\right),$$

where the pseudo-time step is chosen as $\Delta\tau = 1/(6 \max_{\mathbf{x}} g(\mathbf{x})) = \frac{1}{6}$ for $g > 0$ (since $g \le 1$), i.e. the diffusion-number stability limit for second-order level-set terms [Whitaker 2002]. The feature-preserving mask g is based on the quotient of ψ_0 to its smoothed version (as in self-quotient methods for local contrast normalization): $g = 1 - \mathrm{smoothstep}(\theta, \zeta\,; \psi_0/(G_\sigma \star \psi_0 + 10^{-10}))$ with gaussian blur G_σ of standard deviation $\sigma = 2\Delta x$. Here, $\zeta = 5$ and $\theta = 2$ are a fixed slope-factor and a noise-threshold respectively. This leaves k_ψ as the single free tuning parameter controlling the strength of surface smoothing/denoising.