EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

The Impulse Particle-In-Cell Method

Sergio Sancho[1,2], Jingwei Tang[2], Christopher Batty[3], Vinicius C. Azevedo[2]

[1]ETH Zürich, Switzerland [2]DisneyResearch|Studios, Switzerland [3]University of Waterloo, Canada

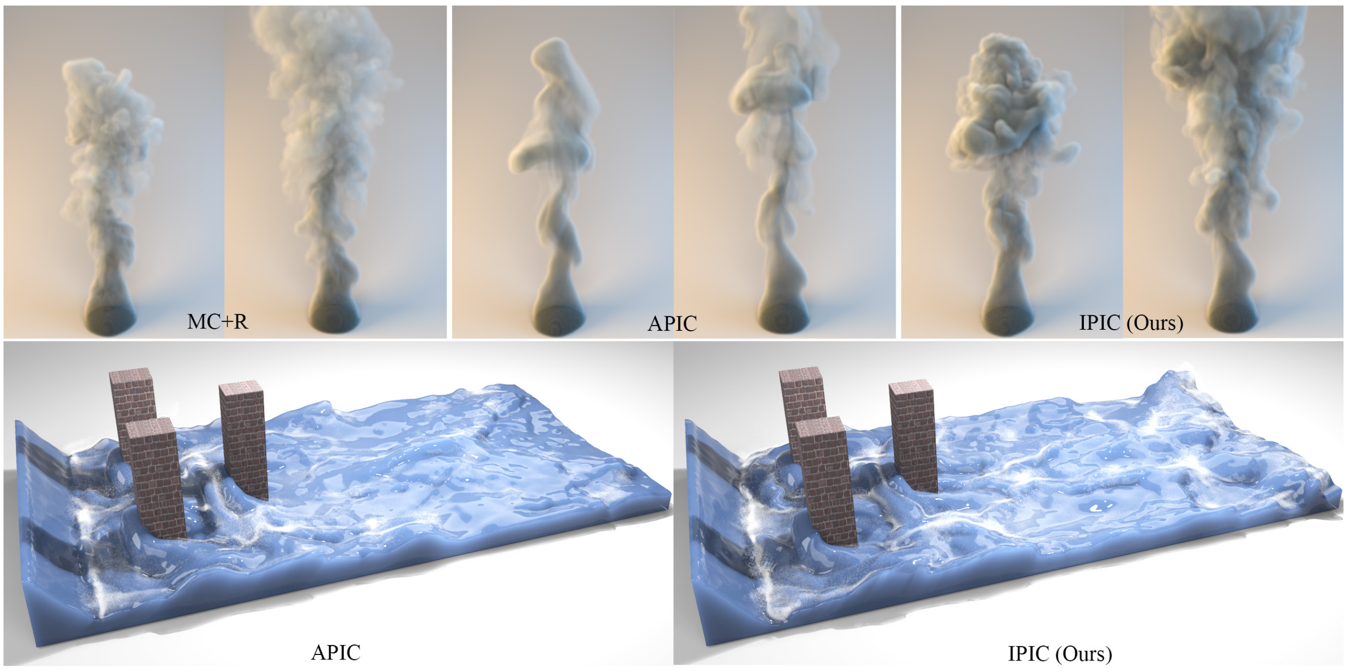


Figure 1: *We propose the Impulse Particle-In-Cell (IPIC) method, a novel extension of the Affine Particle-In-Cell (APIC) method. Our method can be used for both liquid and smoke simulations to better preserve circulation and vortical details.*

Abstract
An ongoing challenge in fluid animation is the faithful preservation of vortical details, which impacts the visual depiction of flows. We propose the Impulse Particle-In-Cell (IPIC) method, a novel extension of the popular Affine Particle-In-Cell (APIC) method that makes use of the impulse gauge formulation of the fluid equations. Our approach performs a coupled advection-stretching during particle-based advection to better preserve circulation and vortical details. The associated algorithmic changes are simple and straightforward to implement, and our results demonstrate that the proposed method is able to achieve more energetic and visually appealing smoke and liquid flows than APIC.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Physical simulation;***

1. Introduction

The chaotic, complex, and intricate behaviors of fluids can only properly manifest in virtual settings if the equations of motion are discretized and solved in a physically faithful manner. In the context of efficient time-splitting schemes, a significant challenge is to simultaneously satisfy conservation of momentum (force balance during advection) and conservation of mass (incompressibility). Early

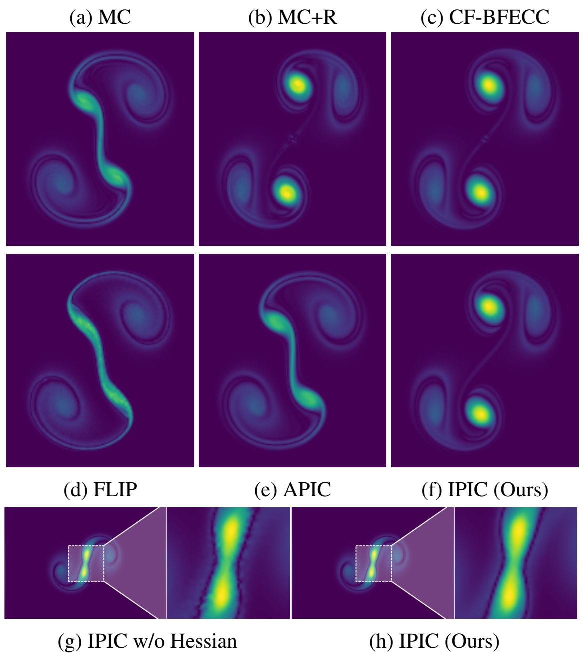


Figure 2: ***2D Taylor Vortices*** *simulations in resolution* 256×256. *MC (a), FLIP (d) and APIC (e) are simulated with* $\Delta t = 0.0125$ *in the first order integration scheme. MC+R (b), CF-BFECC (c) and our method (f) use* $\Delta t = 0.025$ *with a second order time integration scheme. We show results at* $t = 6$. *Ours, CF-BFECC and MC+R preserve the vortices similarly well, while MC, FLIP and APIC dissipate more. In (g) and (h), we show the effect of removing the Hessian term from our method at* $t = 2$. *The energy and vorticity are similarly preserved, but removing the Hessian term results in more noise.*

methods typically prioritized stability and conservation of mass over conservation of momentum, leading to notable energy dissipation [Sta99, FSJ01a]. In response, several techniques have been proposed over the years, ranging from flow enhancements that compensate missing information, such as vorticity confinement [SRF05, ZBG15] or synthetic turbulence [KTJG08, PTG12], to more complex schemes relying on unsplit formulations of Navier-Stokes [MCP*09] or higher order integrators [DL03, SFK*08, ZNT18].

One interesting class of recent approaches incorporates structure-preserving integrators [ETK*07, NWRC22] to better resolve the conservation of momentum. Such approaches augment the advection process by accounting for how the underlying space is itself deformed by the flow, but they have so far been limited to simulating only smoke (and not liquids) in purely Eulerian (i.e., grid-based) settings. In this paper we propose and evaluate a simple modification to such structure-preserving integrators to handle simulations of either smoke or liquid by integrating these ideas into popular hybrid particle/grid solvers, such as APIC [JSS*15].

Fundamentally, our proposed technique relies on constructing the Jacobian of the forward flow map for all particles that represent the fluid. The Jacobian of the Lagrangian forward flow map is then inverted on a per-particle basis and used to correct the particles' velocities. It thus agrees with the backwards flow map correction used in the semi-Lagrangian context by the Covector Fluids method of Nabizadeh et al. [NWRC22]. Our method is straightforward to implement and can be readily integrated into existing solvers, adding only a small computational overhead. We further present a practical technique to significantly improve on the stability of Covector Fluids, by detecting when the correction is likely to be inaccurate. The presented results demonstrate that the flows obtained by our technique are more energetic, better preserve vorticity, and produce more visually compelling details. Our contributions can be summarized as

- The introduction of a novel coupled advection-stretching particle-based method that makes use of the impulse gauge variable to preserve circulation and vortical details, with a simple and intuitive derivation.
- A modified APIC scheme that admits a per-particle velocity correction with a Hessian update to preserve the accuracy of the particle-to-grid transfer.
- The observation that instabilities in velocity-stretching happen because of errors in the construction of the Jacobian of the flow map. Thus, we propose a novel Jacobian-Aware blending scheme that significantly improves the stability of impulse-based methods.
- We demonstrate that our method can be applied in existing hybrid methods for liquid simulation with minor algorithmic changes in order to enhance vortices and rotational motion.

2. Related Work

Early approaches to fluid animation suffered from severe numerical dissipation that detrimentally affects the flow dynamics [Sta99], and several methods have been proposed to tackle this issue. Energy-preserving integrators [MCP*09] can simulate flows with close to zero numerical viscosity, but they require a costly non-linear solve at each time-step. Vortex methods are also known for their excellent conservation properties. They can be discretized in a Lagrangian fashion, by points [PK05, SRF05, Ang17], segments [TRLX22, XTZ*21], filaments [WP10, PCK*19] or sheets [PTG12, BKB12]; or in a hybrid format that combines grid-based velocity/streamfunction data with Lagrangian vortex elements [Leo80, CC91, KL95, ZBG15, CCB*08, FDB22]. In general, vortex methods suffer from complicated boundary treatments and additional costs or instabilities due to vortex stretching terms. Among other relevant works that aim to enrich the liveliness of simulations are vorticity confinement schemes and higher order interpolation [FSJ01b, SRF05], method of characteristic mapping [TP11, SIBA17, QZG*19], turbulence synthesis [KTJG08, PTSG09], and Lattice Boltzmann approaches [LMLD22].

Especially relevant to our work is the recent approach of Nabizadeh et al. [NWRC22], in which the authors propose advecting covectors instead of the usual velocity field. A discrete covector field is intrinsically connected with the underlying discretization mesh. This assumption requires updating transported covector quantities to account for the deformation of the space during transport. This approach has connections to impulse methods [Ose89, Cor95, SC96, FLX*22], velocity advection schemes [But93], and structure-preserving Lie integrators [ETK*07, McK07].

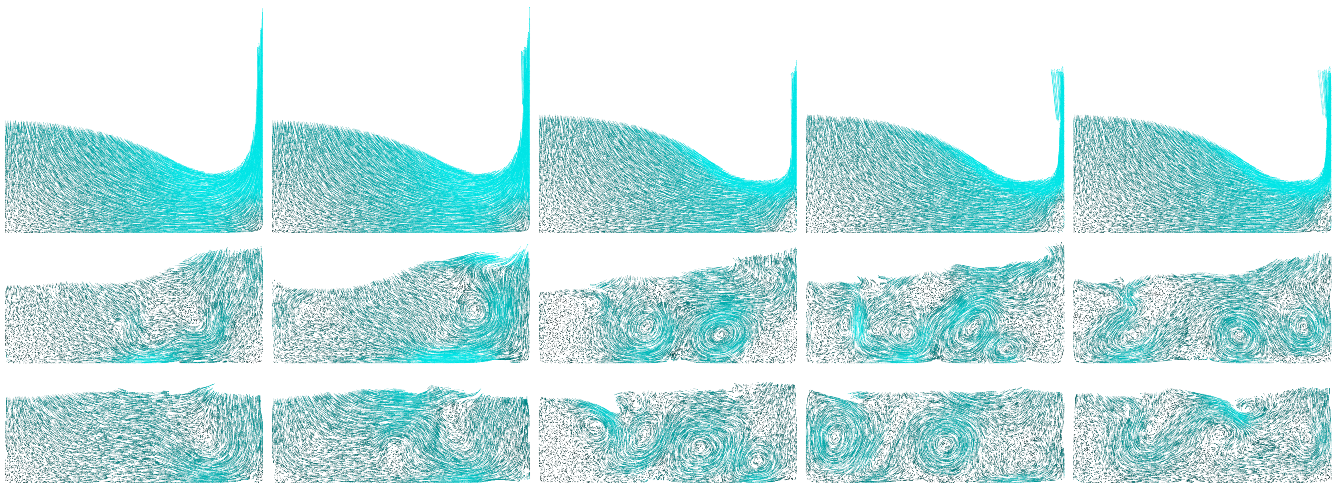


Figure 3: ***2D Dam Break*** *simulations at the resolution of* 64×64 *grid cells. All examples are initialized with 8 particles per cell. A second-order time integration scheme is used with* $\Delta t = 0.25$. *From top to bottom, we show results at time steps* $t = 25$, $t = 220$ *and* $t = 389.5$. *Our method (c) shows more vortical structures than FLIP (a) and APIC (b). Employing lower limiter values for* α *and* β *(d) allow less accurate Jacobian of the flow maps, but is able to provide a more vortical result at* $t = 220$. *IPIC without the Hessian correction (e) becomes less energetic, but is still more vortical than FLIP and APIC.*

Crucial to the work of Nabizadeh et al. [NWRC22] is the combination of more accurate semi-Lagrangian advection schemes (e.g., BFECC [DL03] or MacCormack [SFK*08]) with two-step time-splitting methods [ZNT18, NZT19]. Their approach, however, is limited to purely Eulerian grid-based settings, and does not naturally extend to modelling liquids. Recent work has explored the improvement of the impulse method's accuracy in a grid-based setting by using neural fields [DYZ*23].

Hybrid Lagrangian-Eulerian Fluids combine the capacity of particle-based representations to accurately model transport with the ability of grids to ensure discrete incompressibility [ZB05]. Hybrid methods are thus effective in capturing sub-grid details and have been widely adopted for liquid simulations. Such methods were used to model subgrid details [ABO16, CMSA20, TBBC*22], adopted in collocated variable schemes [GHMR*20] and adaptive grids [NNC*20], combined with SPH for small-scale effects [LTKF08, CIPT14], improved for efficiency [ATW13, FAW*16, SWT*18], enhanced with better particle distribution [AT11, UBH14, KLTB21, QLDJ22], and extended for two-phase flows [BB12, ATW15] and realistic whitewater simulation [WFS22].

Prominent among hybrid approaches is the popular Fluid Implicit Particle method (FLIP) [BR86, ZB05], which increments, rather than overwrites, the particles' velocities by transferring only the interpolated change in the grid velocities before and after pressure projection. FLIP, however, introduces noise due to the discrepancy in the information carried by the particles and the discretization grid. Jiang et al. [JSS*15, JST17] therefore proposed the Affine Particle in Cell method (APIC), which uses concepts of the Generalized Interpolation Material Point method (GIMP) [BK04, PCW07] to create transfers between particles and grid that preserve angular momentum. Due to its versatility and ability to maintain vortical structures, APIC has become a fundamental part of production pipelines for liquid simulations [FSN17]. APIC was further extended to model higher order quantity tracking [FGG*17], combined with a FLIP scheme [FGW*21], and extended in space with a Taylor expansion [NMM23]. For a more thorough analysis of the convergence and properties of the APIC method applied specifically to fluid scenarios, please refer to [DSS20].

3. Background

We begin by conducting a brief review of impulse-based methods and hybrid particle/grid fluid solvers. The motion of volume-preserving inviscid fluids is governed by the incompressible Euler equations, given by

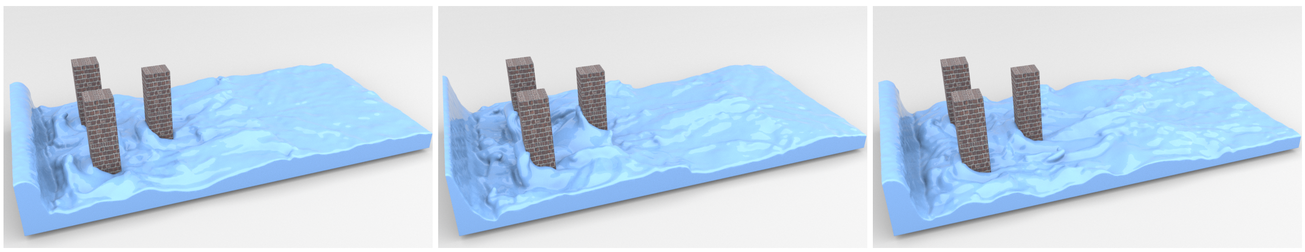
$$\frac{\partial \mathbf{u}(\mathbf{x})}{\partial t} + \mathbf{u}(\mathbf{x}) \cdot \nabla \mathbf{u}(\mathbf{x}) = -\frac{1}{\rho}\nabla p(\mathbf{x}) + \mathbf{f}(\mathbf{x}), \tag{1}$$

$$\nabla \cdot \mathbf{u}(\mathbf{x}) = 0, \tag{2}$$

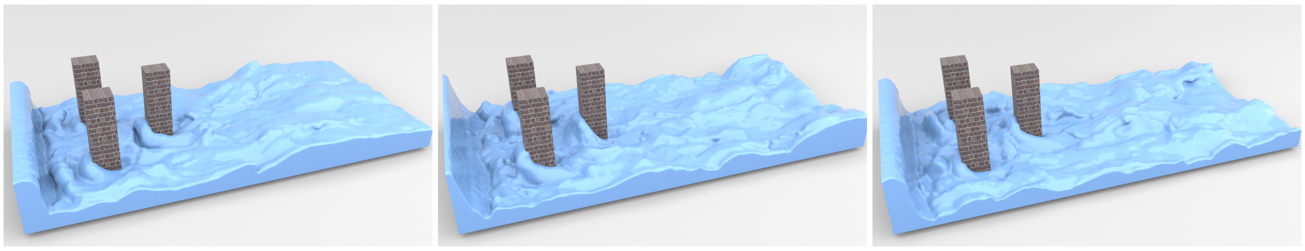
where $\mathbf{u}(\mathbf{x})$, $p(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$ denote the velocity, pressure and external force fields, respectively, while $\mathbf{x}$ represents the spatial coordinates of the domain and ρ the constant fluid density. Viscosity terms are often omitted due to the inherent dissipation of numerical solvers [ETK*07, MCP*09]. These equations can be discretized through a time-splitting method as

$$\frac{\partial \mathbf{u}(\mathbf{x})}{\partial t} = -\mathbf{u}(\mathbf{x}) \cdot \nabla \mathbf{u}(\mathbf{x}) + \mathbf{f}(\mathbf{x}), \tag{3}$$

$$\frac{\partial \mathbf{u}(\mathbf{x})}{\partial t} = -\frac{1}{\rho}\nabla p(\mathbf{x}), \text{ subject to } \nabla \cdot \mathbf{u}(\mathbf{x}) = \mathbf{0}. \tag{4}$$



(a) APIC



(b) IPIC (Ours)

Figure 4: ***3D Liquid Street*** *simulations with a resolution* $150 \times 75 \times 45$ *grid cells. A liquid wave generator causes waves that collide with obstacles. Both methods are simulated with* $\Delta t = 0.25$ *using a second-order integration scheme. From left to right, we show results at* $t = 75.25$, $t = 86.25$ *and* $t = 100.75$ *respectively. Our method (b) is able to produce more detailed wakes at the downstream of the obstacles when compared to APIC (a).*

3.1. Impulse-based methods

The impulse formulation [Cor95, SC96, FLX*22] of the equations of motion allows for a relaxation of the global incompressibility constraint during advection by employing an additional gauge variable. Given that the flow $\mathbf{u}$ is incompressible (i.e., $\nabla \cdot \mathbf{u} = 0$), the impulse gauge variable $\mathbf{m}$ is defined as

$$\mathbf{m} = \mathbf{u} + \nabla\phi, \tag{5}$$

where $\nabla\phi$ is the gradient of an arbitrary scalar field. This term can be reinterpreted as the gradient of pressure that will modify $\mathbf{m}$ to satisfy incompressibility. The resulting equations for mass conservation become

$$\begin{aligned} \nabla^2\phi(\mathbf{x}) &= \nabla \cdot \mathbf{m} && \mathbf{x} \in \Omega, \\ \frac{\partial\phi(\mathbf{x})}{\partial\mathbf{n}} &= 0 && \mathbf{x} \in \partial\Omega_b \ , \\ \phi(\mathbf{x}) &= 0 && \mathbf{x} \in \partial\Omega_f, \end{aligned} \tag{6}$$

where Ω represents the fluid domain, and $\partial\Omega_b$ and $\partial\Omega_f$ are its fluid-solid and fluid-air boundaries, respectively. Due to the change of variables induced by the new gauge variable, the inviscid equation for advection becomes (see Appendix for the derivation):

$$\frac{\partial\mathbf{m}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{m} + (\nabla\mathbf{u})^\top \mathbf{m} = 0. \tag{7}$$

The major difference between the impulse formulation and the original advection in Equation (3) is the extra $(\nabla\mathbf{u})^\top \mathbf{m}$ term, which accounts for the stretching of the advected velocity due to deformation of the space. Through operator splitting, this equation can be broken down into passive transport and velocity stretching as

$$\frac{\partial\mathbf{m}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{m} = 0, \quad \frac{\partial\mathbf{m}}{\partial t} + (\nabla\mathbf{u})^\top \mathbf{m} = 0. \tag{8}$$

By first solving the advection term $\mathbf{u} \cdot \nabla\mathbf{m}$ to obtain an intermediate $\mathbf{m}^*$, the velocity stretching term can then be discretized by backward Euler integration as $\mathbf{m}^{**} = \mathbf{m}^* - \Delta t (\nabla\mathbf{u})^\top \mathbf{m}^{**}$. Rearranging the terms of this equation shows the relationship between the velocity gradient and the advected stretched transport as

$$\mathbf{m}^{**} = (\mathbf{I} + \Delta t\, \nabla\mathbf{u})^{-\top}\, \mathbf{m}^*. \tag{9}$$

We will demonstrate how the expression $\mathbf{I} + \Delta t\, \nabla\mathbf{u}$ is connected with the definition of the flow map in the next section.

3.2. The Flow Map

A flow map represents the correspondences created by the passively transported grid variables that track fluid quantities, yielding a connection between an undeformed fluid domain and its deformed counterpart. Flow maps have been employed as part of a circulation-preserving vorticity-streamfunction solver [ETK*07], incorporated within a higher-order semi-Lagrangian scheme [NWRC22], integrated into Material Point Methods [SSC*13, JST*16], and used for visualizing flows through Lagrangian coherent structures, such as Finite-Time Lyapunov Exponents (FTLEs) [Hal01, Hal02]. The flow map $\Phi_t(\mathbf{x})$ that maps each fluid initial position $\mathbf{x}_0$ in the domain to its location at time t is defined by the integration of the flow velocity field as

$$\Phi_t(\mathbf{x}_0) = \mathbf{x}_0 + \int_0^t \mathbf{u}(\mathbf{x}(\tau), \tau)\, d\tau. \tag{10}$$

The Jacobian of the flow map, $\nabla\Phi_t(\mathbf{x})$, represents the deformation tensor of the fluid domain, and its derivative over time is directly connected with the velocity gradient [Cor95] as

$$\frac{\partial}{\partial t}\nabla\Phi_t(\mathbf{x}_t) = \nabla\mathbf{u}(\mathbf{x})\, \nabla\Phi_t(\mathbf{x}_t). \tag{11}$$

Covector Fluids [NWRC22] discretizes flow maps by integrating grid locations backwards in time, sampling extra points around the advected vector component (Figure 5, (b)). Instead, we adopt an approach common in FTLE visualizations, computing forward flow maps by sampling extra points around a given particle (Figure 5, (c)). Adopting a forward Euler time integration scheme, one can discretize Equation (11) as

$$\nabla\Phi_{n+1}(\mathbf{x}_{n+1}) = \nabla\Phi_n(\mathbf{x}_n) + \Delta t \nabla\mathbf{u}(\mathbf{x}_n)\nabla\Phi_n(\mathbf{x}_n), \tag{12}$$

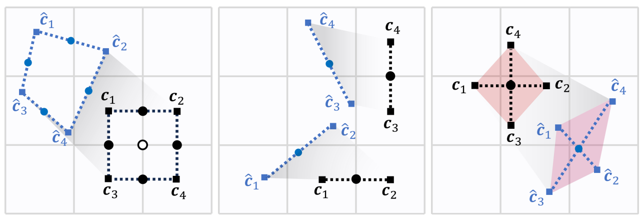
with n and Δt being the time step index and the time step size, respectively. Since the Jacobian of the flow map relative to the current time step is the identity $\nabla\Phi_n(\mathbf{x}_n) = \mathbf{I}$, we have

$$\nabla\Phi_{n+1}(\mathbf{x}_{n+1}) = \mathbf{I} + \Delta t \nabla\mathbf{u}(\mathbf{x}_n). \tag{13}$$

We will drop the subscript n for the rest of the paper, assuming that the flow map is always computed with respect to the previous time step. This derivation relates the flow map from Equation (13) to the impulse stretching in Equation (9) as

$$\mathbf{m}^{**} = (\nabla\Phi)^{-\top}\mathbf{m}^{*}. \tag{14}$$

These definitions set up a mathematical approach that simultaneously simplifies the formalism outlined by previous exterior calculus approaches, while clarifying how flow maps interact with classical impulse methods.



(a) *[ETK*07]* **(b)** *[NWRC22]* **(c)** *Ours*

Figure 5: *Differences in how flow maps are computed by distinct methods. The black color represents the undeformed state, while the blue color represents the deformed state. (a) Elcott et al. [ETK*07] use a streamfunction-vorticity formulation to backtrace circulation (black empty circle) sampled in the deformed space (blue dotted lines). (b) Nabizadeh et al. [NWRC22] employ a backwards flow map discretization: given a staggered grid location (black circle), the advected velocity vector is multiplied by a single row of the transposed Jacobian of the inverse flow map, which is geometrically equivalent to the approach of Elcott et al. (c) Our approach samples four additional points (c_1 to c_4) for each particle, which are then advected forward ($\hat{c}_1$ to $\hat{c}_4$). The full forward flow map Jacobian is computed using a finite-difference approach as in FTLE methods.*

3.3. Hybrid discretizations

Hybrid methods discretize the equations of motion with two distinct representations: Lagrangian particles are used to solve the advection terms (Equation (3)) and to track the evolution of the surface, while a background Eulerian grid is used to enforce incompressibility (Equation (4)). We represent discrete *grid-based* positions and velocities as $\mathbf{x}^{\boxplus}$ and $\mathbf{u}^{\boxplus}$ respectively, while their *particle-based* counterparts are represented by $\mathbf{x}^{\circ}$ and $\mathbf{u}^{\circ}$. For the simplicity in notation, in the following explanation we assume that all particles carry unit mass and that quantity transfers are normalized by the transferred mass at the computed position to conserve momentum.

The advection in a Lagrangian setting updates particle positions with a forward flow map computed from the grid velocities, while the attributes that these particles carry are kept unchanged. We denote the discrete particle-based flow map as $\Phi^{\circ} : \mathbb{R}^{N\times 3} \to \mathbb{R}^{N\times 3}$, with N being the number of particles. The advected velocities that satisfy momentum conservation on the grid are

$$\tilde{\mathbf{u}}^{\boxplus} \leftarrow \mathcal{I}_{p2g}(\mathbf{u}^{\circ}, \Phi^{\circ}(\mathbf{x}^{\circ}), \mathbf{x}^{\boxplus}), \tag{15}$$

where $\mathcal{I}_{p2g}$ is a particle-to-grid transfer function. The standard Particle-in-Cell (PIC) method defines the $\mathcal{I}_{p2g}$ routine as

$$\mathcal{I}^{\text{PIC}}_{p2g}(\mathbf{u}^{\circ}, \mathbf{x}^{\circ}, \mathbf{x}^{\boxplus}) = \sum_p w(\mathbf{x}^{\boxplus}, \mathbf{x}^{\circ}_p)\,\mathbf{u}^{\circ}_p, \tag{16}$$

with $w(\mathbf{x}^{\boxplus}, \mathbf{x}^{\circ}_p)$ representing the weights obtained from a pre-specified interpolation function. The summation in Equation (16) is conducted over the particles inside a small vicinity of each grid point. Since PIC suffers from severe energy dissipation, the Affine Particle-in-Cell (APIC) method modifies the particle-to-grid transfer to better preserve the velocity field. The corresponding $\mathcal{I}_{p2g}$ operation is

$$\mathcal{I}^{\text{APIC}}_{p2g}(\mathbf{u}^{\circ}, \mathbf{x}^{\circ}, \mathbf{x}^{\boxplus}, \mathbf{A}^{\circ}) = \sum_p w(\mathbf{x}^{\boxplus}, \mathbf{x}^{\circ}_p)\,\left(\mathbf{u}^{\circ}_p + \mathbf{A}^{\circ}_p(\mathbf{x}^{\boxplus} - \mathbf{x}^{\circ}_p)\right). \tag{17}$$

where $\mathbf{u}^{\circ}_p + \mathbf{A}^{\circ}_p(\mathbf{x}^{\boxplus} - \mathbf{x}^{\circ}_p)$ is the per-particle affine velocity contribution relative to a grid variable at $\mathbf{x}^{\boxplus}$. After transferring the particle velocities, mass conservation (Equation (4)) is solved on the grid. The resulting divergence-free velocity field is used to update the per-particle linear velocity and affine transformation as

$$\mathbf{u}^{\circ} = \sum_i w(\mathbf{x}^{\boxplus}_i, \mathbf{x}^{\circ})\,\mathbf{u}^{\boxplus}_i, \quad \mathbf{A}^{\circ} = \sum_i \nabla w(\mathbf{x}^{\boxplus}_i, \mathbf{x}^{\circ})\,\mathbf{u}^{\boxplus}_i. \tag{18}$$

The above equation assumes that the interpolation kernel $w(\mathbf{x}, \mathbf{y})$ is trilinear [PCW07, NMM23]. This assumption simplifies the computation of the affine velocity, which effectively models the per-particle matrix $\mathbf{A}^{\circ}$ as the interpolation of the velocity gradient at the given location, $\nabla\mathbf{u}(\mathbf{x})$.

4. An Impulse-based Hybrid Advection Scheme

Having outlined how the flow map relates to the impulse equations, and how hybrid discretizations can efficiently model advection, we can now propose a simple modification to hybrid schemes that incorporates structure-preserving deformations when advecting the impulse gauge variable. Accordingly, the particles are now defined to carry impulse (and its affine component), rather than velocity. The particle positions are first passively advected by the incompressible flow velocities, as usual. Then, per Equation (14), the per-particle impulse gauge variables are stretched by the inverse transposed Jacobian of the flow map, denoted $\mathcal{J}^{\circ} = (\nabla\Phi^{\circ})^{-\top}$, using

$$\mathbf{m}^{\circ} = \mathcal{J}^{\circ}\mathbf{u}^{\circ}. \tag{19}$$

We re-initialize the impulse variable at each step to be the divergence-free velocity $\mathbf{u}^{\circ}$, and denote the impulse gauge variable value after stretching as $\mathbf{m}^{\circ}$.

The next question is how the per-particle affine matrix $\mathbf{A}^\circ$ for the impulse should be stretched. When tri-linear kernels are used, the APIC transfer can be interpreted as a first-order Taylor approximation of a function centered around a Lagrangian coordinate [BK04, PCW07, NMM23] as

$$\mathbf{u}(\mathbf{x}^{\boxplus}) = \mathbf{u}^\circ + \nabla\mathbf{u}^\circ \Delta\mathbf{x} + \mathcal{O}\left((\Delta\mathbf{x})^2\right), \tag{20}$$

where $\Delta\mathbf{x} = (\mathbf{x}^{\boxplus} - \mathbf{x}^\circ)$ and $\nabla\mathbf{u}^\circ = \mathbf{A}^\circ$ (Equation (18)). Applying the Jacobian of the flow map to the APIC transfer yields

$$\mathbf{m}(\mathbf{x}^{\boxplus}) \approx \mathcal{J}^\circ\mathbf{u}^\circ + \nabla\left(\mathcal{J}^\circ\mathbf{u}^\circ\right)\Delta\mathbf{x}. \tag{21}$$

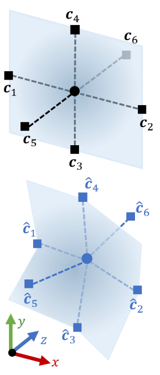
Assuming that the Jacobian of the flow map is locally constant around the transported particles, Equation (21) simplifies to

$$\mathbf{m}(\mathbf{x}^{\boxplus}) = \mathcal{J}^\circ\mathbf{u}^\circ + \mathcal{J}^\circ\nabla\mathbf{u}^\circ\Delta\mathbf{x}. \tag{22}$$

This means one can simply transform the affine matrix with the inverse transpose of the forward flow map Jacobian to get the stretched affine matrix:

$$\mathbf{A}^\circ_{\mathbf{m}} = \mathcal{J}^\circ\mathbf{A}^\circ. \tag{23}$$

This simple modification allows us to incorporate the per-particle stretched impulse $\mathbf{m}^\circ$ and matrix $\mathbf{A}^\circ_{\mathbf{m}}$ as inputs of the APIC transfer (Equation (17)). We emphasize that the *forward* flow map is necessary for particle-based advection, since the particle positions are evolved forward in time; previous approaches [ETK*07, NWRC22] employ semi-Lagrangian schemes instead, which integrate velocities by going *backwards* in time.



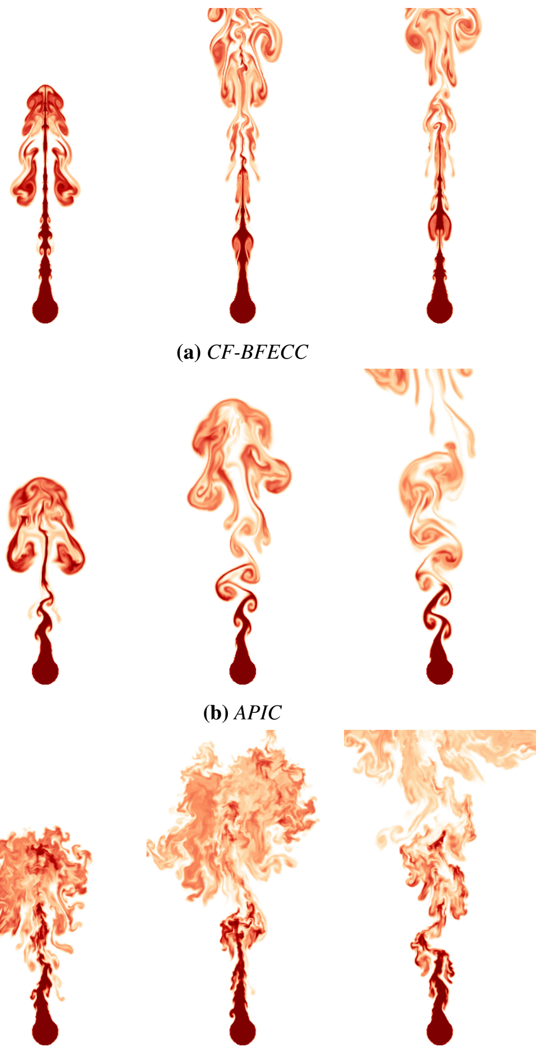
Our method requires the computation of the Jacobian of the per-particle flow map Φ° on an unstructured set of points. This Jacobian could potentially be computed based on the previous positions of neighbouring particles. Such an approach, however, would require querying closest particle information, which reduces the performance of the method. Instead, we choose to compute finite difference approximations from extra temporary sampled particles around the location of the Jacobian computation. The inset image shows the particle arrangement in 3D: two particles are created and displaced in each Cartesian direction. The inverse transpose of the forward flow map Jacobian $\mathcal{J}^\circ$ is computed in 3D by

$$\mathcal{J}^\circ_{3D} \approx \frac{1}{\delta x}\begin{bmatrix}(\hat{\mathbf{c}}_2 - \hat{\mathbf{c}}_1)^\top \\ (\hat{\mathbf{c}}_4 - \hat{\mathbf{c}}_3)^\top \\ (\hat{\mathbf{c}}_6 - \hat{\mathbf{c}}_5)^\top\end{bmatrix}^{-1}, \tag{24}$$

where δx is the separation of the extra temporary sampled particles and $\hat{\mathbf{c}}$ is the transported positions of the extra particles. In our implementations, we set δx to be the same as the grid spacing.

4.1. Non-constant Jacobian Flow Map

Modifying per-particle quantities with Equation (23) only works if the Jacobian of the flow map is locally constant around a particle. In


(a) *CF-BFECC*

(b) *APIC*

(c) *IPIC (Ours)*

Figure 6: ***2D Smoke Plume*** *simulations with resolution* 384×512 *grid cells. We employ* $\Delta t = 0.25$ *for CF-BFECC (a) and IPIC (c), while APIC (b) uses* $\Delta t = 0.125$ *(first-order integration). From left to right, we show results at* $t = 50.25$, $t = 63$ *and* $t = 87$ *respectively. Our method can express more detailed vortices than APIC while maintaining stability. CF-BFECC explodes early and fails to finish: the frame shown in its third column is at* $t = 66.5$ *and is the last frame before the simulation becomes unstable.*

the non-constant case, Equation (21) expands as

$$\mathbf{m}(\mathbf{x}^{\boxplus}) = \mathcal{J}^\circ\mathbf{u}^\circ + \mathcal{J}^\circ\nabla\mathbf{u}^\circ\Delta\mathbf{x} + \left(\nabla\mathcal{J}^\circ \cdot \mathbf{u}^\circ\right)\Delta\mathbf{x}. \tag{25}$$

The extra term comes from the gradient of the non-constant Jacobian: $\nabla\mathcal{J}^\circ$ is the Hessian of the inverse transposed flow map, a third order tensor that is single contracted with the vector $\mathbf{u}^\circ$. The last two terms of Equation (25) can be combined to derive a more convenient

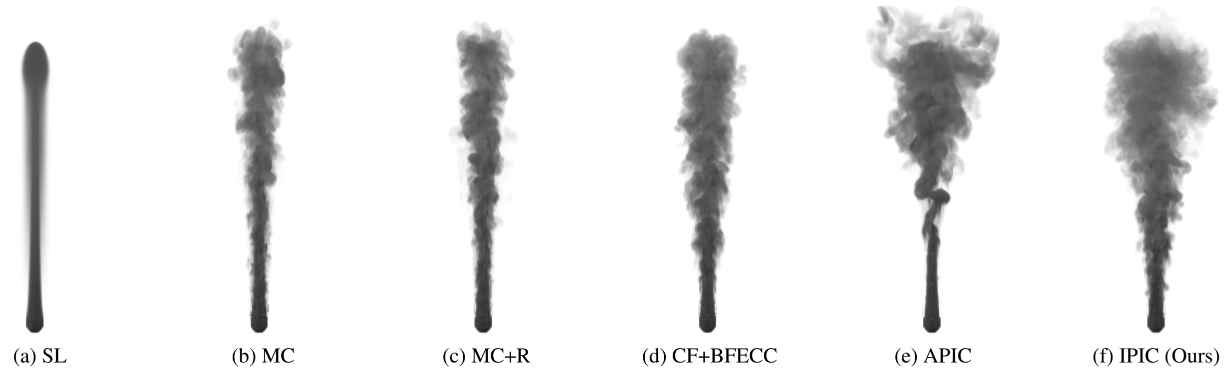


Figure 7: ***3D Smoke Plume*** *simulations at a resolution of* $128\times256\times128$ *cells. SL (a), MC(b), APIC (e) use* $\Delta t = 0.125$ *while MC+R (c), CF+BFECC (d) and ours (f) are simulated with* $\Delta t = 0.25$ *using a second-order integration scheme. We show results for all methods at* $t = 100$ *except for CF+BFECC, which explodes after* $t = 68$*. Our method exhibits more vortical details than the other methods while also maintaining better stability than CF+BFECC.*

expression to update the per-particle affine matrix as

$$\tilde{\mathbf{A}}^{\circ}_{\mathbf{m}} = \mathcal{J}^{\circ}\nabla\mathbf{u}^{\circ} + \nabla\mathcal{J}^{\circ} : \mathbf{u}^{\circ} = \mathcal{J}^{\circ}(\mathbf{A}^{\circ} - \mathbf{H}^{\circ}),$$
$$\text{with } \mathbf{H}^{\circ} = \begin{bmatrix} \left(m_x\nabla\frac{\partial\Phi_x}{\partial x} + m_y\nabla\frac{\partial\Phi_y}{\partial x} + m_z\nabla\frac{\partial\Phi_z}{\partial x}\right)^{\top} \\ \left(m_x\nabla\frac{\partial\Phi_x}{\partial y} + m_y\nabla\frac{\partial\Phi_y}{\partial y} + m_z\nabla\frac{\partial\Phi_z}{\partial y}\right)^{\top} \\ \left(m_x\nabla\frac{\partial\Phi_x}{\partial z} + m_y\nabla\frac{\partial\Phi_y}{\partial z} + m_z\nabla\frac{\partial\Phi_z}{\partial z}\right)^{\top} \end{bmatrix}, \tag{26}$$

where $\mathbf{m}^{\circ} = (m_x, m_y, m_z)^{\top}$ and $\nabla\frac{\partial\Phi_e}{\partial f}, e, f \in (x,y,z)$ represents the second order derivatives of the flow map.

The per-particle matrix $\mathbf{H}^{\circ}$ requires the evaluation of second derivatives of the flow map, and extra points need to be sampled to in order to compute these derivatives. In the supplementary material we include detailed diagrams that illustrate how we sample particles to approximate the Hessian, along with the derivations needed to obtain Equation (26). In summary, when the Jacobian of the flow map is not constant around the particle positions, the APIC transfer (Equation (17)) employs the stretched velocity $\mathbf{m}^{\circ}$ and modified affine matrix $\tilde{\mathbf{A}}^{\circ}_{\mathbf{m}}$. A full outline of a single time step of the IPIC advection-stretching is shown in Algorithm 1. For time integration, we employ a second-order multi-stepping algorithm [NWRC22], outlined in Algorithm 2.

4.2. Enforcing Stability by a Jacobian-Aware Blending

The original covector approach [NWRC22] can simulate fluids with intricate vortical details, but suffers from significant instability issues that severely limit the time step size of the algorithm. We propose a novel limiter on the Jacobian of the flow map that improves stability of impulse-based formulations. Our key insight is that velocity stretching is unstable when flow maps are not accurately constructed. Errors in the construction of the flow map are exacerbated in regions of higher turbulence or close to boundaries, where inaccuracies due to numerical integration of particle positions and the negative effects of not enforcing strictly divergence-free interpolants [CPAB22] are more severe.

The flow map error can be quantified by measuring the determinant of its Jacobian: for a strictly divergence-free velocity field coupled with an accurate position integrator, the determinant of the flow map should be equal to 1, indicating that the volume is perfectly conserved. To limit the amount of incorrect stretching induced by inaccurate flow maps, we apply a smoothed double-sided step function to modulate the strength of the stretching of Lagrangian quantities as

$$\begin{aligned} \mathbf{m}^{\circ} &= \xi(q^{\circ})\mathcal{J}^{\circ}\mathbf{u}^{\circ} + \left(1-\xi(q^{\circ})\right)\mathbf{u}^{\circ}, \\ \tilde{\mathbf{A}}^{\circ}_{\mathbf{m}} &= \xi(q^{\circ})\tilde{\mathbf{A}}^{\circ}_{\mathbf{m}} + \left(1-\xi(q^{\circ})\right)\mathbf{A}^{\circ}, \end{aligned} \tag{27}$$

with

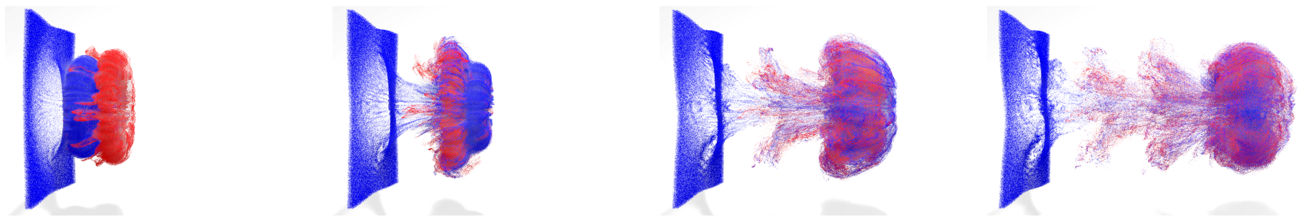
$$q^{\circ} = \frac{1 - ||\mathcal{J}^{\circ}| - 1| - \alpha}{\beta - \alpha}, \quad \xi(q^{\circ}) = \begin{cases} 0, & q \le 0 \\ 3q^2 - 2q^3, & 0 \le q \le 1 \\ 1, & 1 \le q \end{cases} \tag{28}$$

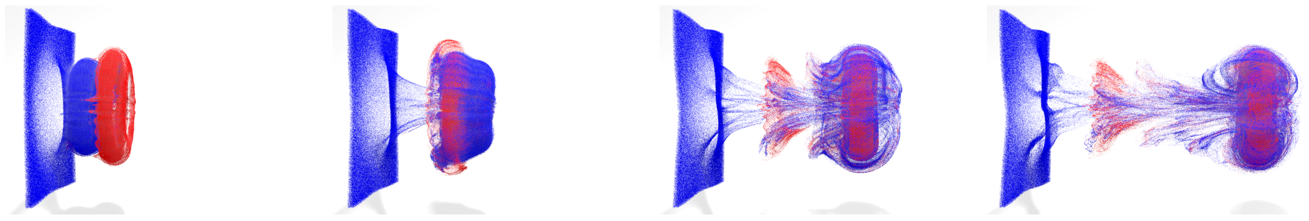
where α and β are parameters satisfying $0 \le \alpha < \beta \le 1$ that control the blending limits. In Section 5, we demonstrate the impact of the Jacobian-aware blending. When $\xi(q^{\circ})$ is far from one, the method is more strict in enforcing flow map correctness, and the behavior of the solver more closely mimics a standard APIC solver; with less strict enforcement, the method becomes less stable while also producing and maintaining more intricate, vortical details.

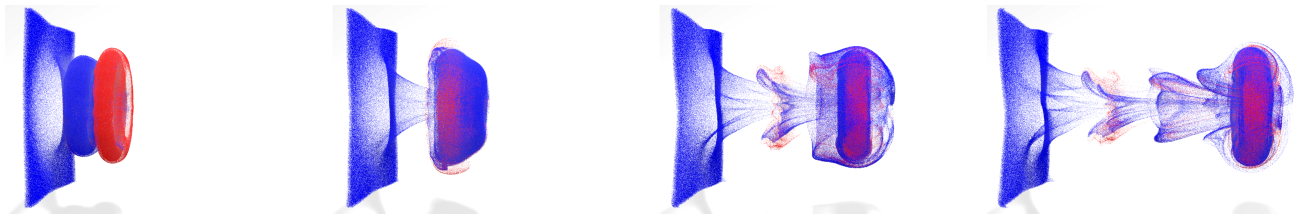
Moreover, when simulating liquids, the evaluation of the Jacobian of the flow map at the interface between air and fluid cells can become incorrect, since velocities from the fluid are extrapolated to the air. Since this creates simulations that have noticeable artifacts, we revert back to FLIP at the interface of fluid and air cells.

5. Results

We test the effectiveness of our method in several different scenarios, evaluating its performance and quality against established state-of-the-art methods. One advantage of our proposed hybrid approach is that it can be seamlessly integrated into liquid solvers, extending previous approaches [NWRC22] that were limited to smoke settings. We also reimplemented several methods in our pipeline to provide fair comparisons. The first-order (in time) semi-Lagrangian [Sta99] and MacCormack [SRF05], second-order


(a) MC+R


(b) CF+BFECC


(c) APIC

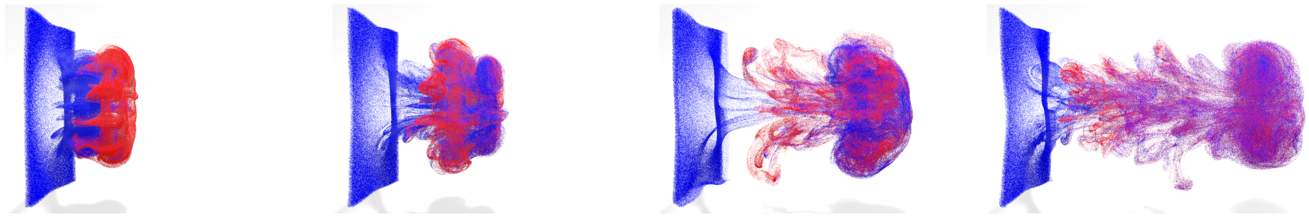

(d) IPIC (Ours)

Figure 8: ***Vortex Leapfrogging*** *simulations at a resolution of* $256 \times 128 \times 128$ *cells. MC+R (a), CF+BFECC (b) and our method (d) use* $\Delta t = 0.5$*, while APIC (c) employs* $\Delta t = 0.25$*. From left to right, we show results at* $t = 121$*,* $t = 258$*,* $t = 500.5$ *and* $t = 767.5$*, respectively. IPIC produces more detailed vortical structures when compared with other methods.*

Table 1: ***Summary of methods*** *used in this paper.*

Method	Acronym	Reference
Semi-Lagrangian	SL	[Sta99]
MacCormack	MC	[SFK*08]
Back-and-Forth Error Compensation and Correction	BFECC	[SFK*08]
Advection-Reflection	R	[NZT19]
Covector Fluids	CF	[NWRC22]
Fluid-Implicit-Particle	FLIP	[ZB05]
Affine Particle-In-Cell	APIC	[JSS*15]
Impulse Particle-In-Cell	IPIC	Our method

advection-reflection [NZT19], and second-order (in time) Covector Fluids [NWRC22] are the baseline implementations for simulating smoke examples; PIC, FLIP [ZB05] and APIC [JSS*15] provide baselines for liquid scenes. Table 1 shows all the implemented methods, and Table 2 lists experiment parameters and performance statistics. We refer the reader to the accompanying video for animated visualizations of our results.

The Impulse Particle-In-Cell method is implemented in Pytorch [PGM*19], extending a previous differentiable solver pipeline [TCCS21]. All simulations were performed on a system with an NVIDIA RTX3090 GPU with 24GB memory and an AMD Ryzen 7 5800X CPU with 64GB memory. Efficiently performing particle-to-grid operations implies increased memory costs, which can be prohibitive when evaluating our method on GPUs. We alleviate this requirement by sequentially computing the particle splatting operation in chunks, which can make particle-to-grid transfers less efficient. We use linear interpolation kernels for both grid-to-particle and particle-to-grid transfers and a third-order Runge-Kutta integrator for advancing positions.

5.1. Validation

2D Taylor Vortices Our 2D solver is validated using a *Taylor vortices* setup [McK07] that has been widely used for testing

Algorithm 1 A single step of the IPIC advection-stretching

Input: flow field $\mathbf{u}^{\boxplus}$; particle positions $\mathbf{x}^{\circ}$; grid positions $\mathbf{x}^{\boxplus}$; timestep Δt; non-constant Jacobian flag $\mathcal{H}$; stability parameters α and β.

```
 1: $\mathbf{u}^\circ, \mathbf{A}^\circ \leftarrow \mathcal{I}^{\text{APIC}}_{g2p}(\mathbf{x}^\circ, \mathbf{x}^\boxplus, \mathbf{u}^\boxplus)$
 2: for each particle do
 3:     $\mathbf{c}^\circ \leftarrow$ CreateJacobianParticles($\mathbf{x}^\circ$)
 4:     if $\mathcal{H}$ then
 5:         $\mathbf{h}^\circ \leftarrow$ CreateHessianParticles($\mathbf{x}^\circ$)
 6:         $\mathbf{h}^\circ \leftarrow$ RungeKutta($\mathbf{h}^\circ; \mathbf{u}^\boxplus, \Delta t$)
 7:     end if
 8:     $\mathbf{x}^\circ \leftarrow$ RungeKutta($\mathbf{x}^\circ; \mathbf{u}^\boxplus, \Delta t$)
 9:     $\mathbf{c}^\circ \leftarrow$ RungeKutta($\mathbf{c}^\circ; \mathbf{u}^\boxplus, \Delta t$)
10:     $\mathcal{J}^\circ \leftarrow$ ComputeJacobian($\mathbf{c}^\circ$)            ▷ Equation (24)
11:     $\mathbf{A}^\circ_\mathbf{m} \leftarrow \mathbf{A}^\circ$
12:     if $\mathcal{H}$ then
13:         $\mathbf{H}^\circ \leftarrow$ ComputeHessian($\mathbf{h}^\circ$)            ▷ Equation (26)
14:         $\mathbf{A}^\circ_\mathbf{m} \leftarrow \mathbf{A}^\circ - \mathbf{H}^\circ$
15:     end if
16:     $q^\circ \leftarrow \frac{1-||\mathcal{J}^\circ|-1|-\alpha}{\beta-\alpha}$
17:     $\mathbf{m}^\circ \leftarrow \xi(q^\circ)\mathcal{J}^\circ\mathbf{u}^\circ + (1-\xi(q^\circ))\,\mathbf{u}^\circ$
18:     $\mathbf{A}^\circ \leftarrow \xi(q^\circ)\mathcal{J}^\circ\mathbf{A}^\circ_\mathbf{m} + (1-\xi(q^\circ))\,\mathbf{A}^\circ$
19: end for
20: $\mathbf{m}^\boxplus \leftarrow \mathcal{I}^{\text{APIC}}_{p2g}(\mathbf{m}^\circ, \mathbf{x}^\circ, \mathbf{x}^\boxplus, \mathbf{A}^\circ)$
```

Output: Advected stretched impulse field $\mathbf{m}^{\boxplus}$

Algorithm 2 Second order time integration scheme

Input: Flow field $\mathbf{u}_n^{\boxplus}$; timestep Δt

```
1: $\tilde{\mathbf{m}}_{n+\frac{1}{2}} \leftarrow$ IPIC($\mathbf{m}_n; \mathbf{u}_n, \frac{\Delta t}{2}$)            ▷ Algorithm 1
2: $\mathbf{u}_{n+\frac{1}{2}} \leftarrow$ DivFreeProjection($\tilde{\mathbf{m}}_{n+\frac{1}{2}}$)
3: $\tilde{\mathbf{m}}_{n+1} \leftarrow$ IPIC($\mathbf{m}_n; \mathbf{u}_{n+\frac{1}{2}}, \Delta t$)            ▷ Algorithm 1
4: $\mathbf{u}_{n+1} \leftarrow$ DivFreeProjection($\tilde{\mathbf{m}}_{n+1}$)
```

Output: Velocity field $\mathbf{u}_{n+1}^{\boxplus}$

energy preservation properties in Computer Graphics. We generate two initial vortices separated by a distance of 0.81 using $\omega(\mathbf{x}) = \frac{U}{a}\left(2-\frac{r^2}{a^2}\right)\exp\left(\frac{1}{2}\left(1-\frac{r^2}{a^2}\right)\right)$, with $a = 0.3$ and $U = 1.0$. Vortex core positions are set so that they should theoretically remain separated through the course of the simulation; dissipative solvers are not able to preserve this property. Note that methods such as MacCormack and APIC are run with halved time steps to match the number of projection steps of second-order in time methods (Algorithm 2). Figure 2 provides a visualization of the resulting vorticity maps in each of the methods at time $t = 6$. We observed that Covector Fluids is very sensitive to MacCormack / BFECC extrema clamping modes, and minor changes to the clamping procedure can drastically decrease or increase the energy of the system.

3D Leapfrogging Vortex Rings An alternative evaluation of a solver's ability to conserve kinetic energy is to test a vortex leapfrogging example. Concentric vortex rings with different radii are initialized with equal circulations, and the expected behavior is that one ring will go through the other in alternating fashion. The true analytical solution for the inviscid case would reproduce this behavior indefinitely; for numerical solvers, we observe the number of times that each vortex ring goes through another before collapsing to a single ring. Figure 8 shows leapfrogging results for grid-based Advection-Reflection MacCormack (MC+R) and Covector Fluids methods, and for hybrid APIC and IPIC discretizations. APIC was simulated with a halved time step size, since all other methods are second-order in time. Thus, the number of pressure projections is the same across different methods. The sequence shows that IPIC produces more detailed vortical structures when compared to other methods, while also producing consistent ring motion. We compare the kinetic energy profile of different methods in Figure 9.

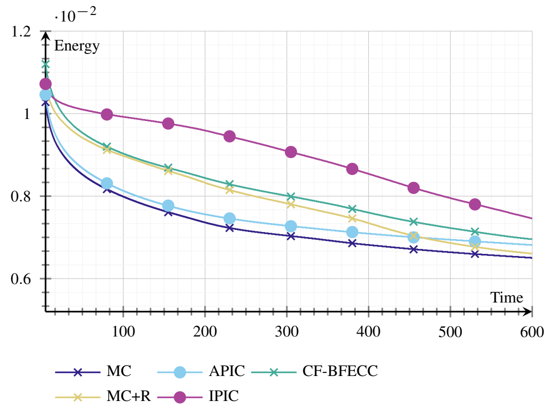


Figure 9: *Kinetic energy plot for different methods used in the 3D Vortex Leapfrogging example (Figure 8). IPIC better conserves the energy throughout the simulation.*

Table 2: ***Parameters and performance statistics.*** *All runtime statistics are reported in seconds per frame when second order integration schemes are used. For first order integration schemes, runtime is reported in seconds per two frames for a fair comparison.*

Scene	Resolution	# Part.	Ours	Ours w/o H	APIC
Taylor Vortices (Fig. 2)	256×256	1.05M	0.77	0.55	0.35
2D Plume (Fig. 6)	384×512	786k	0.71	0.56	0.44
2D Dam Break (Fig.3)	64×64	10k	0.18	0.17	0.15
Leapfrogging (Fig. 8)	256×128×128	16.78M	69.2	29.0	8.6
3D Plume (Fig. 7)	128×256×128	16.78M	67.8	28.6	7.4
3D Dam Break (Fig. 10)	128×100×40	471k	2.1	1.0	0.4
Liquid Sink (Fig. 11)	96×24×96	1.77M	5.3	2.2	0.7
Liquid Street (Fig. 4)	150×75×45	1.2M∼1.8M	7.2	2.9	0.8

5.2. Smoke

2D Smoke Plume We initialize a 2D smoke source in a rectangular domain. For all smoke examples, advection of the smoke concentration $\rho(\mathbf{x})$ is performed on the regular grid with the MacCormack advection scheme, instead of transporting densities through particles. We chose this setting so we could provide fair visual comparisons against grid-based methods that do not rely on particles for smoke concentration advection. A standard buoyancy force $\mathbf{f}(\mathbf{x}) = \gamma\rho(\mathbf{x})\mathbf{j}$ is added for all our smoke examples and for this particular 2D case we set $\gamma = 0.002$. Figure 6 shows a comparison between Covector Fluids, APIC and IPIC: our approach is not only better able to produce vortical structures, but is also stable.

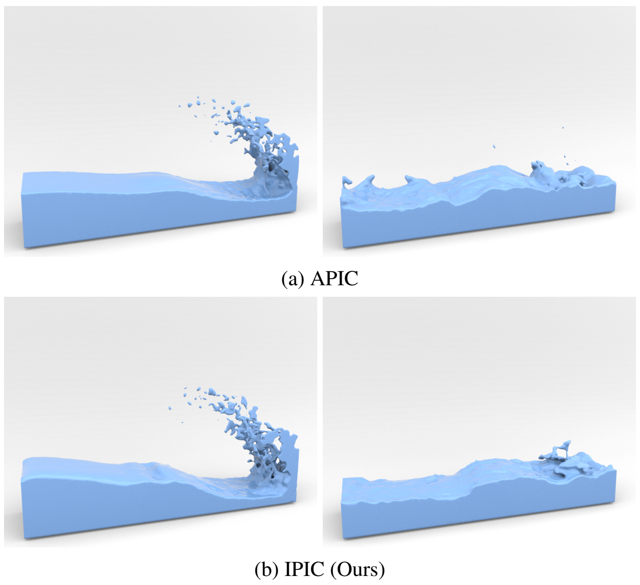
3D Smoke Plume Figure 7 shows a comparison of a 3D smoke simulation driven by buoyancy forces ($\gamma = 0.004$). This example is particularly interesting, since it demonstrates that different methods affect the spreading of turbulence throughout the simulation. All methods, except Covector Fluids, which blows up at $t = 68$, are stable. IPIC creates more turbulent structures that spread more realistically along with upward driving forces.

5.3. Liquids

2D Dam Break Existing hybrid liquid solvers can be easily extended with our particle-based algorithm, offering better circulation-preserving capabilities. In this example, we initialize a column of water in the left side of the domain. We show in Figure 3 a comparison between FLIP, APIC and IPIC: how our method is able to maintain more energetic vortical structures. Additionally, we compare the effect of the parameters of the Jacobian-aware blending: less strict (α=0.8, β=0.9) settings allow less accurate Jacobian of the flow map, making the simulation less stable. But it is able to provide a more vortical result at $t = 220$. More strict (α=0.99, β=0.998) parameters constrain the Jacobian better, and are also able to provide a much more vortical simulation than FLIP and APIC. Lastly, we also observe that the simulation is less energetic if we assume the Jacobian is constant around a particle position.

3D Dam Break and 3D Liquid Street Figure 10 shows a comparison between APIC and IPIC for a 3D dam break scenario. As we revert IPIC back to FLIP at fluid-air interfaces for stability, our result looks similar to APIC. Inspired by Yang et al. [YXZ*21], we create a 3D "liquid street" example that initializes a column of water that moves up and down at the left of the domain. IPIC is better able to create turbulent details behind obstacles and at boundary between liquid-air interfaces when compared to APIC.

3D Sink Lastly, we create a 3D sink example. A square domain is initialized with water, and on the bottom of the domain we add a hole that allows the water to flow through. We initialize the water velocities with a rotational velocity field. When compared to APIC, IPIC creates more interesting motions at the vortex centerline.


(a) APIC

(b) IPIC (Ours)

Figure 10: ***3D Dam Break*** *simulations at a resolution of* $128 \times 100 \times 40$ *with* $\Delta t = 0.25$*. The liquid block is initialized with 8 particles per cell. From left to right, we show results at* $t = 27.25$ *and* $t = 91.25$*. Our method (b) gives similar results to APIC (a). Because the flow map at interfaces is not correct due to the extrapolation of velocities, IPIC reverts back to FLIP on those regions.*

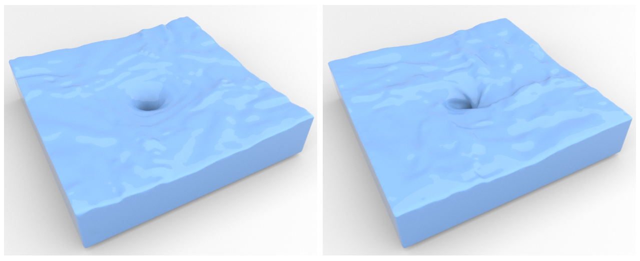

(a) *APIC* **(b)** *IPIC (Ours)*

Figure 11: ***3D Water Sink*** *simulations at a resolution of* $96 \times 24 \times 96$*. Each cell is initialized with 8 particles. Both methods use* $\Delta t = 0.25$ *with the second order integration scheme. We show the results at* $t = 150$*. Our method (b) produces richer small structures on the surface than APIC (a), especially near the central vortex.*

6. Conclusions

We have presented the Impulse Particle-In-Cell method, a novel hybrid discretization that combines the strength of particle-based transport with the versatility of grid-based incompressibility. Our method can model both smoke and liquid phases, and exhibits improved conservation of energy and vortical details compared to previous methods. Furthermore, we proposed a novel Jacobian-aware blending that increases the stability of the velocity-stretching step commonly used in related impulse-based/structure-preserving approaches.

The major limitation of IPIC is its reliance on sampling extra particles to compute the deformation flow map. This can introduce computational inefficiency and higher memory costs, especially when assuming that the Jacobian is not constant around particles. Moreover, flow maps can be inaccurate close to interfaces, and we rely on reverting back to simpler advection schemes in those scenarios. For future work, we believe that our method can benefit from a more thorough treatment of boundary conditions. In addition, the Jacobian-aware blending can be employed as a limiter for pure grid-based advection schemes that rely on velocity-stretching.

References

[ABO16] Azevedo V. C., Batty C., Oliveira M. M.: Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Transactions on Graphics 35*, 4 (jul 2016), 1–12. doi:10.1145/2897824.2925919. 3

[Ang17] Angelidis A.: Multi-scale vorticle fluids. *ACM Transactions on Graphics 36*, 4 (jul 2017), 1–12. doi:10.1145/3072959.3073606. 2

[AT11] Ando R., Tsuruno R.: A particle-based method for preserving fluid sheets. In *Proceedings - SCA 2011: ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, New York, USA, 2011), ACM Press, pp. 7–16. doi:10.1145/2019406.2019408. 3

[ATW13] Ando R., Thürey N., Wojtan C.: Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics 32*, 4 (jul 2013), 1. doi:10.1145/2461912.2461982. 3

[ATW15] Ando R., Thuerey N., Wojtan C.: A stream function solver for liquid simulations. *ACM Transactions on Graphics 34*, 4 (jul 2015), 53:1–53:9. doi:10.1145/2766935. 3

[BB12] Boyd L., Bridson R.: MultiFLIP for energetic two-phase fluid simulation. *ACM Transactions on Graphics 31*, 2 (apr 2012), 1–12. doi:10.1145/2159516.2159522. 3

[BK04] Bardenhagen S., Kober E.: The Generalized Interpolation Material Point Method. *CMES - Computer Modeling in Engineering and Sciences 5* (2004). 3, 6

[BKB12] Brochu T., Keeler T., Bridson R.: Linear-time smoke animation with vortex sheet meshes. *Computer Animation 2012 - ACM SIGGRAPH / Eurographics Symposium Proceedings, SCA 2012* (2012), 87–95. 2

[BR86] Brackbill J. U., Ruppel H. M.: 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. doi:10.1016/0021-9991(86)90211-1. 3

[But93] Buttke T. F.: *Velicity Methods: Lagrangian Numerical Methods which Preserve the Hamiltonian Structure of Incompressible Fluid Flow*. Tech. rep., 1993. doi:10.1007/978-94-015-8137-0_3. 2

[CC91] Chang C. C., Chern R. L.: A numerical study of flow around an impulsively started circular cylinder by a deterministic vortex method. *Journal of Fluid Mechanics 233*, 243 (1991), 243–263. doi:10.1017/S0022112091000472. 2

[CCB*08] Chatelain P., Curioni A., Bergdorf M., Rossinelli D., Andreoni W., Koumoutsakos P.: Billion vortex particle direct numerical simulations of aircraft wakes. *Computer Methods in Applied Mechanics and Engineering 197*, 13-16 (2008), 1296–1304. doi:10.1016/j.cma.2007.11.016. 2

[CIPT14] Cornelis J., Ihmsen M., Peer A., Teschner M.: IISPH-FLIP for incompressible fluids. *Computer Graphics Forum 33*, 2 (may 2014), 255–262. doi:10.1111/cgf.12324. 3

[CMSA20] Chen Y. L., Meier J., Solenthaler B., Azevedo V. C.: An extended cut-cell method for sub-grid liquids tracking with surface tension. *ACM Transactions on Graphics 39*, 6 (2020). doi:10.1145/3414685.3417859. 3

[Cor95] Cortez R.: *Impulse-Based Methods for Fluid Flow*. Tech. rep., 1995. 2, 4

[CPAB22] Chang J., Partono R., Azevedo V. C., Batty C.: Curl-Flow: Boundary-Respecting Pointwise Incompressible Velocity Interpolation for Grid-Based Fluids. *ACM Transactions on Graphics 41*, 6 (dec 2022), 1–21. doi:10.1145/3550454.3555498. 7

[DL03] Dupont T. F., Liu Y.: Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *Journal of Computational Physics 190*, 1 (sep 2003), 311–324. doi:10.1016/S0021-9991(03)00276-6. 2, 3

[DSS20] Ding O., Shinar T., Schroeder C.: Affine particle in cell method for MAC grids and fluid simulation. *Journal of Computational Physics 408* (2020), 109311. doi:10.1016/j.jcp.2020.109311. 3

[DYZ*23] Deng Y., Yu H.-X., Zhang D., Wu J., Zhu B.: Fluid simulation on neural flow maps. *ACM Trans. Graph. 42*, 6 (2023). 3

[ETK*07] Elcott S., Tong Y., Kanso E., Schröder P., Desbrun M.: Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics 26*, 1 (jan 2007), 4–es. doi:10.1145/1189762.1189766. 2, 3, 4, 5, 6

[FAW*16] Ferstl F., Ando R., Wojtan C., Westermann R., Thuerey N.: Narrow band FLIP for liquid simulations. *Computer Graphics Forum 35*, 2 (may 2016), 225–232. doi:10.1111/cgf.12825. 3

[FDB22] Frey M., Dritschel D., Böing S.: EPIC: The Elliptical Parcel-In-Cell method. *Journal of Computational Physics: X 14* (2022), 100109. doi:10.1016/j.jcpx.2022.100109. 2

[FGG*17] Fu C., Guo Q., Gast T., Jiang C., Teran J.: A polynomial particle-in-cell method. *ACM Transactions on Graphics 36*, 6 (nov 2017), 1–12. doi:10.1145/3130800.3130878. 3

[FGW*21] Fei Y. R., Guo Q., Wu R., Huang L., Gao M.: Revisiting integration in the material point method. *ACM Transactions on Graphics 40*, 4 (2021). doi:10.1145/3450626.3459678. 3

[FLX*22] Feng F., Liu J., Xiong S., Yang S., Zhang Y., Zhu B.: Impulse Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* (2022). doi:10.1109/TVCG.2022.3149466. 2, 4

[FSJ01a] Fedkiw R., Stam J., Jensen H. W.: Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001* (New York, NY, USA, aug 2001), ACM, pp. 15–22. doi:10.1145/383259.383260. 2

[FSJ01b] Fedkiw R., Stam J., Jensen H. W.: Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001* (New York, New York, USA, 2001), ACM Press, pp. 15–22. doi:10.1145/383259.383260. 2

[FSN17] Frost B., Stomakhin A., Narita H.: Moana: Performing water. In *ACM SIGGRAPH 2017 Talks, SIGGRAPH 2017* (New York, NY, USA, jul 2017), ACM, pp. 1–2. doi:10.1145/3084363.3085091. 3

[GHMR*20] Gagniere S., Hyde D., Marquez-Razon A., Jiang C., Ge Z., Han X., Guo Q., Teran J.: A hybrid lagrangian/eulerian collocated velocity advection and projection method for fluid simulation. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2020 39*, 8 (dec 2020), 1–14. arXiv:2003.12227, doi:10.1111/cgf.14096. 3

[Hal01] Haller G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena 149*, 4 (mar 2001), 248–277. doi:10.1016/S0167-2789(00)00199-8. 4

[Hal02] Haller G.: Lagrangian coherent structures from approximate velocity data. *Physics of Fluids 14*, 6 (jun 2002), 1851–1861. doi:10.1063/1.1477449. 4

[JSS*15] Jiang C., Schroeder C., Selle A., Teran J., Stomakhin A.: The affine Particle-In-Cell method. *ACM Transactions on Graphics 34*, 4 (jul 2015), 1–10. doi:10.1145/2766996. 2, 3, 8

[JST*16] Jiang C., Schroeder C., Teran J., Stomakhin A., Selle A.: The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses* (New York, NY, USA, jul 2016), ACM, pp. 1–52. doi:10.1145/2897826.2927348. 4

[JST17] Jiang C., Schroeder C., Teran J.: An angular momentum conserving affine-particle-in-cell method. *Journal of Computational Physics 338* (2017), 137–164. arXiv:1603.06188, doi:10.1016/j.jcp.2017.02.050. 3

[KL95] Koumoutsakos P., Leonard A.: *High-Resolution simulations of the flow around an impulsively started cylinder using vortex methods*, vol. 296. 1995. doi:10.1017/S0022112095002059. 2

[KLTB21] Kugelstadt T., Longva A., Thuerey N., Bender J.: Implicit Density Projection for Volume Conserving Liquids. *IEEE Transactions on Visualization and Computer Graphics 27*, 4 (apr 2021), 2385–2395. doi:10.1109/TVCG.2019.2947437. 3

[KTJG08] Kim T., Thürey N., James D., Gross M.: Wavelet turbulence for fluid simulation. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 50. 2

[Leo80] Leonard A.: Vortex methods for flow simulation. *Journal of Computational Physics 37*, 3 (1980), 289–335. doi:10.1016/0021-9991(80)90040-6. 2

[LMLD22] Li W., Ma Y., Liu X., Desbrun M.: Efficient kinetic simulation of two-phase flows. *ACM Transactions on Graphics 41*, 4 (2022). doi:10.1145/3528223.3530132. 2

[LTKF08] Losasso F., Talton J., Kwatra N., Fedkiw R.: Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 797–804. 3

[McK07] McKenzie A.: HOLA: a High-Order Lie Advection of discrete differential forms, with applications in fluid dynamics. URL: https://resolver.caltech.edu/CaltechETD:etd-05292007-100432. 2, 8

[MCP*09] Mullen P., Crane K., Pavlov D., Tong Y., Desbrun M.: Energy-preserving integrators for fluid animation. *ACM Transactions on Graphics 28*, 3 (jul 2009), 1. doi:10.1145/1531326.1531344. 2, 3

[NMM23] Nakamura K., Matsumura S., Mizutani T.: Taylor particle-in-cell transfer and kernel correction for material point method. *Computer Methods in Applied Mechanics and Engineering 403* (jan 2023), 115720. doi:10.1016/j.cma.2022.115720. 3, 5, 6

[NNC*20] Nakanishi R., Nascimento F., Campos R., Pagliosa P., Paiva A.: RBF liquids: An Adaptive PIC Solver Using RBF-FD. *ACM Transactions on Graphics 39*, 6 (2020), 1–13. doi:10.1145/3414685.3417794. 3

[NWRC22] Nabizadeh M. S., Wang S., Ramamoorthi R., Chern A.: Covector fluids. *ACM Transactions on Graphics 41*, 4 (jul 2022), 1–16. doi:10.1145/3528223.3530120. 2, 3, 4, 5, 6, 7, 8

[NZT19] Narain R., Zehnder J., Thomaszewski B.: A second-order advection-reflection solver. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 2*, 2 (jul 2019), 1–14. doi:10.1145/3340257. 3, 8

[Ose89] Oseledets V. I.: *On a new way of writing the Navier-Stokes equation. The Hamiltonian formalism*. Tech. Rep. 3, 1989. doi:10.1070/rm1989v044n03abeh002122. 2

[PCK*19] Padilla M., Chern A., Knöppel F., Pinkall U., Schröder P.: On bubble rings and ink chandeliers. *ACM Transactions on Graphics 38*, 4 (aug 2019), 1–14. doi:10.1145/3306346.3322962. 2

[PCW07] P. C. Wallstedt J. E. G.: Improved Velocity Projection for the Material Point Method. *Computer Modeling in Engineering & Sciences 19*, 3 (2007), 223–232. doi:10.3970/cmes.2007.019.223. 3, 5, 6

[PGM*19] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Köpf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang L., Bai J., Chintala S.: PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32* (dec 2019). URL: http://arxiv.org/abs/1912.01703, arXiv:1912.01703. 8

[PK05] Park S. I., Kim M. J.: Vortex fluid for gaseous phenomena. In *Computer Animation, Conference Proceedings* (New York, New York, USA, 2005), ACM Press, pp. 261–270. doi:10.1145/1073368.1073406. 2

[PTG12] Pfaff T., Thuerey N., Gross M.: Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics 31*, 4 (jul 2012), 1–8. doi:10.1145/2185520.2185608. 2

[PTSG09] Pfaff T., Thuerey N., Selle A., Gross M.: Synthetic turbulence using artificial boundary layers. *ACM Transactions on Graphics 28*, 5 (dec 2009), 1. doi:10.1145/1618452.1618467. 2

[QLDJ22] Qu Z., Li M., De Goes F., Jiang C.: The power particle-in-cell method. *ACM Transactions on Graphics 41*, 4 (jul 2022). doi:10.1145/3528223.3530066. 3

[QZG*19] Qu Z., Zhang X., Gao M., Jiang C., Chen B.: Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics 38*, 4 (2019), 1–12. doi:10.1145/3306346.3322945. 2

[SC96] Summers D. M., Chorin A. J.: *Numerical vorticity creation based on impulse conservation*. Tech. Rep. 5, 1996. doi:10.1073/pnas.93.5.1881. 2, 4

[SFK*08] Selle A., Fedkiw R., Kim B., Liu Y., Rossignac J.: An unconditionally stable MacCormack method. *Journal of Scientific Computing 35*, 2-3 (jun 2008), 350–371. doi:10.1007/s10915-007-9166-4. 2, 3, 8

[SIBA17] Sato T., Igarashi T., Batty C., Ando R.: A Long-term semi-lagrangian method for accurate velocity advection. *SIGGRAPH Asia 2017 Technical Briefs, SA 2017* (2017). doi:10.1145/3145749.3149443. 2

[SRF05] Selle A., Rasmussen N., Fedkiw R.: A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics 24*, 3 (jul 2005), 910–914. doi:10.1145/1073204.1073282. 2, 7

[SSC*13] Stomakhin A., Schroeder C., Chai L., Teran J., Selle A.: A material point method for snow simulation. *ACM Transactions on Graphics 32*, 4 (jul 2013), 1–10. doi:10.1145/2461912.2461948. 4

[Sta99] Stam J.: Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999* (New York, New York, USA, 1999), ACM Press, pp. 121–128. doi:10.1145/311535.311548. 2, 7, 8

[SWT*18] Sato T., Wojtan C., Thuerey N., Igarashi T., Ando R.: Extended narrow band FLIP for liquid simulations. *Computer Graphics Forum 37*, 2 (may 2018), 169–177. doi:10.1111/cgf.13351. 3

[TBBC*22] Tao M., Batty C., Ben-Chen M., Fiume E., Levin D. I.: VEMPIC: Particle-in-Polyhedron Fluid Simulation for Intricate Solid Boundaries. *ACM Transactions on Graphics 41*, 4 (2022). doi:10.1145/3528223.3530138. 3

[TCCS21] Tang J., C. Azevedo V., Cordonnier G., Solenthaler B.: Honey, I Shrunk the Domain: Frequency-aware Force Field Reduction for Efficient Fluids Optimization. *Computer Graphics Forum 40*, 2 (may 2021), 339–353. doi:10.1111/cgf.142637. 8

[TP11] Tessendorf J., Pelfrey B.: The Characteristic Map for Fast and Efficient VFX Fluid Simulations. *Computer* (2011). URL: http://jtessen.people.clemson.edu/papers_files/CMFluids.pdf. 2

[TRLX22] Tao R., Ren H., Liu J., Xiao F.: A Lagrangian vortex method for smoke simulation with two-way fluid–solid coupling. *Computers and Graphics (Pergamon) 107* (2022), 289–302. doi:10.1016/j.cag.2022.08.007. 2

[UBH14] Um K., Baek S., Han J.: Advanced Hybrid Particle-Grid Method with Sub-Grid Particle Correction. *Computer Graphics Forum 33*, 7 (oct 2014), 209–218. doi:10.1111/cgf.12489. 3

[WFS22] Wretborn J., Flynn S., Stomakhin A.: Guided bubbles and wet foam for realistic whitewater simulation. *ACM Transactions on Graphics 41*, 4 (2022). doi:10.1145/3528223.3530059. 3

[WP10] Weissmann S., Pinkall U.: Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers on - SIGGRAPH '10* (New York, New York, USA, 2010), ACM Press, p. 1. doi:10.1145/1833349.1778852. 2

[XTZ*21] XIONG S., TAO R., ZHANG Y., FENG F., ZHU B.: Incompressible flow simulation on vortex segment clouds. *ACM Transactions on Graphics 40*, 4 (2021). doi:10.1145/3450626.3459865. 2

[YXZ*21] YANG S., XIONG S., ZHANG Y., FENG F., LIU J., ZHU B.: Clebsch gauge fluid. *ACM Transactions on Graphics 40*, 4 (2021). doi:10.1145/3450626.3459866. 10

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3 (jul 2005), 965–972. doi:10.1145/1073204.1073298. 3, 8

[ZBG15] ZHANG X., BRIDSONY R., GREIFZ C.: Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics 34*, 4 (jul 2015), 52:1–52:8. doi:10.1145/2766982. 2

[ZNT18] ZEHNDER J., NARAIN R., THOMASZEWSKI B.: An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics 37*, 4 (aug 2018), 1–8. doi:10.1145/3197517.3201324. 2, 3