An Impulse Ghost Fluid Method for Simulating Two-Phase Flows

YUCHEN SUN, Georgia Institute of Technology, USA
LINGLAI CHEN, Harvard University, USA
WEIYUAN ZENG, Zhejiang University, China
TAO DU, Tsinghua University, Shanghai Qi Zhi Institute, China
SHIYING XIONG, Zhejiang University, China
BO ZHU, Georgia Institute of Technology, USA

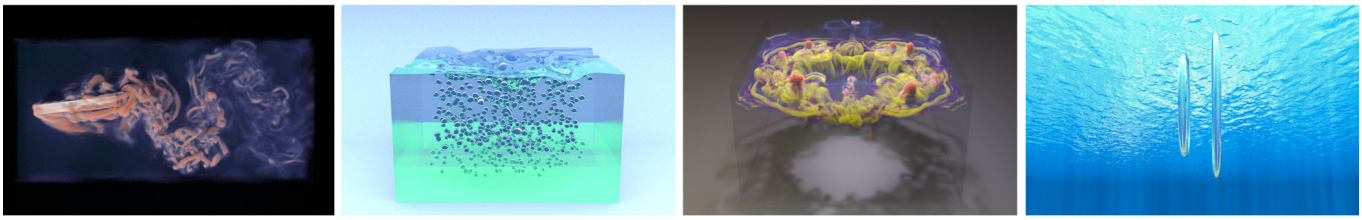


Fig. 1. Our Impulse Ghost Fluid Method preserves vortical structures and fluid volume across a broad spectrum of two-phase vortical flow simulations. We present four illustrative snapshots from simulations utilizing this method: a swinging fishtail viewed from above (*left*), rising bubbles in a tank (*middle left*), a rising bubble ring hitting the liquid-air interface (*middle right*) and a pair of leapfrogging bubble rings (*right*).

This paper introduces a two-phase interfacial fluid model based on the impulse variable to capture complex vorticity-interface interactions. Our key idea is to leverage bidirectional flow map theory to enhance the transport accuracy of both vorticity and interfaces simultaneously and address their coupling within a unified Eulerian framework. At the heart of our framework is an impulse ghost fluid method to solve the two-phase incompressible fluid characterized by its interfacial dynamics. To deal with the history-dependent jump of gauge variables across a dynamic interface, we develop a novel path integral formula empowered by spatiotemporal buffers to convert the history-dependent jump condition into a geometry-dependent jump condition when projecting impulse to velocity. We demonstrate the efficacy of our approach in simulating and visualizing several interface-vorticity interaction problems with cross-phase vortical evolution, including interfacial whirlpool, vortex ring reflection, and leapfrogging bubble rings.

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

Additional Key Words and Phrases: two-phase flow, flow map, gauge method

ACM Reference Format:
Yuchen Sun, Linglai Chen, Weiyuan Zeng, Tao Du, Shiying Xiong, and Bo Zhu. 2024. An Impulse Ghost Fluid Method for Simulating Two-Phase Flows. *ACM Trans. Graph.* 43, 6, Article 269 (December 2024), 12 pages. https://doi.org/10.1145/3687963

Authors' addresses: Yuchen Sun, yuchen.sun.eecs@gmail.com, Georgia Institute of Technology, USA; Linglai Chen, linglai27@yahoo.com, Harvard University, USA; Weiyuan Zeng, zengweiyuan35@gmail.com, Zhejiang University, China; Tao Du, taodu.eecs@gmail.com, Tsinghua University, Shanghai Qi Zhi Institute, China; Shiying Xiong, shiying.xiong@zju.edu.cn, Zhejiang University, China; Bo Zhu, bo.zhu@gatech.edu, Georgia Institute of Technology, USA.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2024 Copyright held by the owner/author(s).
0730-0301/2024/12-ART269
https://doi.org/10.1145/3687963

1 INTRODUCTION

Various two-phase flow phenomena manifest complex interactions between vorticity and interfaces, exemplified by cavitation bubble flow, toroidal bubble ring, cross-interface swimming, etc. Accurately simulating these phenomena requires capturing the long-term evolution of both vortical structures and interface features in a two-phase setting. Specifically, such captures should consider the phase change and jump conditions on a dynamically evolving surface separating water and air. Traditional fluid simulators discretized on an Eulerian grid face numerical dissipation when transporting vorticity and interfaces, evidenced by the volume loss of small bubbles and the dissipated vortical structures, hampering both the physical fidelity and the visual appearance of the simulation of these flow phenomena. Typically, auxiliary data structures such as particles [Patkar et al. 2013] are necessary to preserve these flow details, particularly in a turbulent setting.

Impulse-based methods have emerged in computer graphics and computational physics as a promising solution for solving incompressible flow systems exhibiting complex vortical evolution (e.g., see [Deng et al. 2023; Li et al. 2024; Nabizadeh et al. 2022; Zhou et al. 2024]). Equipped with an accurate, long-range flow map established across a time interval, impulse (covector) can be geometrically transported with this map along with its Jacobian. The combination of impulse and flow maps naturally preserves and develops complex vortical structures and achieves impressive results regarding their physical accuracy and visual complexity. However, most current impulse-based schemes focus on solving incompressible flow without a free surface (i.e., smoke simulation in graphics), and none of the existing methods can tackle multi-phase fluid.

The main challenge of employing impulse to simulate a two-phase system lies in tackling the jump conditions on the interface. Due

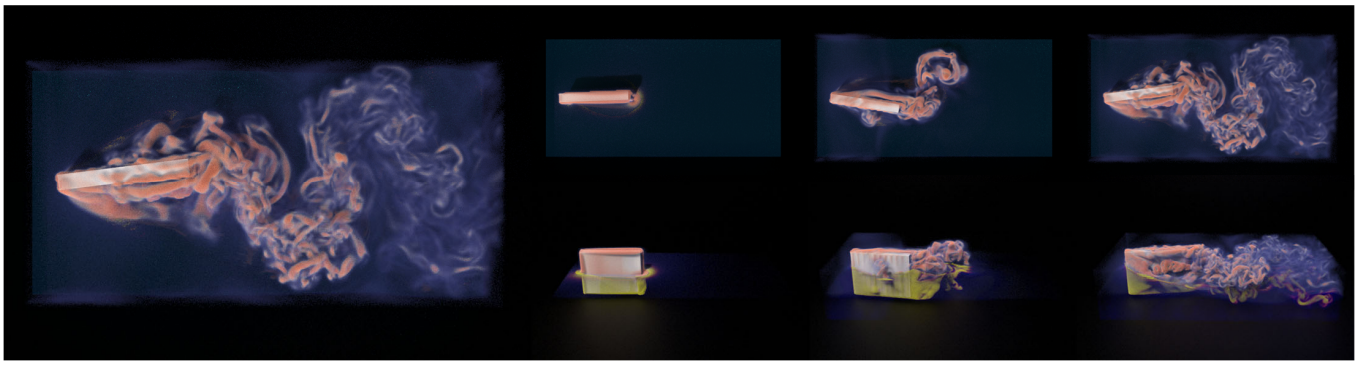


Fig. 2. A rotating fishtail in an inlet. Vorticity in two phases is visualized using different colormaps. Top: Top view. Bottom: Side view.

to the spatiotemporal nature of the gauge variable, the previously geometry-dependent jump condition (e.g., surface tension due to the local interfacial geometry) becomes history-dependent, which needs to be converted into a path integral along the trajectory of every virtual particle on the interface over the entire time interval of its flow map. Directly calculating this integral for each grid sample is impractical with an implicit interface representation such as a level set. Moreover, due to the dynamic nature of the problem, calculating such an integral requires performing additional inside/outside checks for each quadrature along the path according to the interface's position at each time instant, which affects the accuracy and robustness of the entire solver due to the accumulated numerical errors across the interface.

To address these challenges, this paper develops a novel Eulerian framework to simulate two-phase flows under a flow-map perspective. We mainly address the challenge of tackling the spatiotemporal jump conditions across a dynamic implicit interface with flow maps by devising a novel framework named Impulse Ghost Fluid Method (IGFM). To tackle the accumulated jump of the gauge variable across a dynamic interface, we propose a novel scheme named Path Integral Projection. Leveraging spatiotemporal buffers and flow maps, we use a path integral to calculate the history part of the gauge variable and solve for the emerging part with a varying-coefficient Poisson solver. The bidirectional flow map evolved in the process can also be employed to accurately track the level set. As a result, our framework achieves state-of-the-art transport accuracy of both vorticity and interface simultaneously under a single flow map discretized on an Eulerian grid. We demonstrate the efficacy of our approach by simulating a variety of two-phase vortical flow phenomena, such as interfacial whirlpool, vortex ring reflection, and leapfrogging bubble rings, with our particular focus on capturing the cross-interface vortical structures that have played an essential role in many cross-interface flow phenomena.

We summarize our main contributions as:

- A novel Impulse Ghost Fluid Method (IGFM) for simulating two-phase fluid.
- A novel Path-Integral-Projection scheme to tackle the history-dependent jump of gauge variable on an Eulerian grid.
- A novel Bidirectional-Marched-Flow-Map Level Set (BMFM-LS) method to enhance volume preservation.
- A two-phase flow solver to capture complex cross-interface vortex flow phenomena.

2 RELATED WORK

Two-phase Flows. In the graphics community, a popular approach for simulating two-phase flows is the Ghost Fluid Method (GFM) [Fedkiw et al. 1999]. This method employs a level set to track the interface and sharply captures the discontinuous jump in fluid density at the liquid-air boundary. It enforces incompressibility for both liquid and air through a pressure projection scheme. Hong and Kim [2005] first adapted the method for graphics to simulate bubbles, with a variety of work following. Mihalef et al. [2006] used Coupled Level Set and Volume-Of-Fluid (CLSVOF) method [Sussman and Puckett 2000] to address the volume loss of level set methods. A volume control method was developed by Kim et al. [2007]. They calculated a correction term according to volume error and applied it to pressure projection. To capture small-scale droplets and bubbles, Boyd and Bridson [2012] proposed MultiFLIP, combining GFM with the Fluid Implicit Particle (FLIP) method [Brackbill and Ruppel 1986; Zhu and Bridson 2005]. Besides these sharp-interface approaches, Song et al. [2005] and Zheng et al. [2006] employed diffuse-interface models where the fluid density is smoothly changing near the liquid-air interface. Researchers also explored ways to simulate bubbles without simulating the air part, such as stream function solver [Ando et al. 2015] and constraint-based model [Goldade et al. 2020]. In recent years, many works were also devoted to simulating two-phase flows with kinetic solver [Li and Desbrun 2023; Li et al. 2021, 2022] and mixture model [Yan and Ren 2023].

Gauge Methods. The concept of *impulse* was first introduced by Buttke [1992]. It is defined by reformulating the incompressible Navier-Stokes Equation with a gauge transformation [Buttke 1993; Oseledets 1989; Roberts 1972]. After that, researchers in computational physics have explored various applications of impulse, such as turbulence [Buttke and Chorin 1993], numeric stability [Weinan and Liu 2003], and solid boundary treatment [Summers 2000]. In the graphics community, Feng et al. [2023] developed an Eulerian impulse-based gauge method for vortex flow simulations. Sancho et al. [2024] developed a hybrid impulse method, extending APIC [Jiang et al. 2015] with gauge transformation. Nabizadeh et al. [2022]

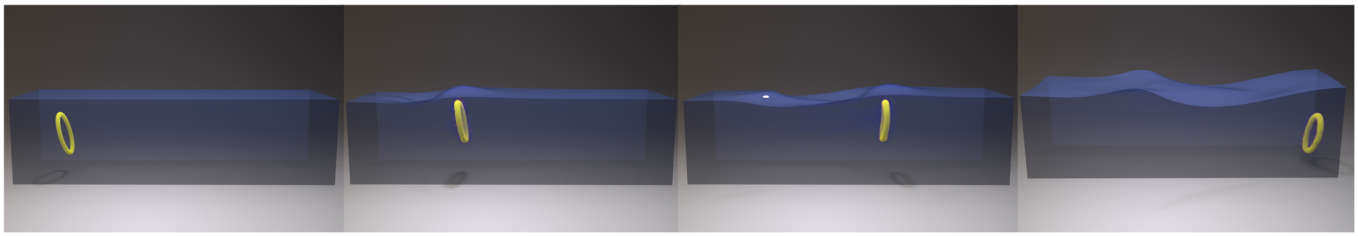


Fig. 3. A vortex ring obliquely approaches the liquid-air interface, creating a bump on the interface before reflecting back.

and Deng et al. [2023] found that the combination of impulse and flow maps significantly improves the preservation of vortical structures. While impulse demonstrates efficacy in smoke simulations, the difficulty of liquid-air boundary treatment hinders its application in two-phase flow simulations. The works of Saye [2016] and Saye [2017] pioneered in this direction, but were limited to a simplified velocity-impulse density formulation [Weinan and Liu 1997]. Recently, Li et al. [2024] proposed using path integral on particles to deal with the Dirichlet boundary condition of gauge variable. This motivates us to tackle the jump condition of gauge variable in two-phase flows by employing backward marched inverse flow maps to perform path integral on an Eulerian grid.

Flow-map Methods. Beginning with the pioneering work of Wiggert and Wylie [1976], flow-map methods (also known as reference-map methods or methods of characteristic mapping), have received consistent attention. These methods reduce the frequency of interpolations by tracking fluid quantities with a long-range mapping, thus diminishing numerical dissipation. Since Hachisuka [2005] introduced the idea to graphics community, many studies have been devoted to utilizing flow maps to increase the accuracy of velocity advection and preserve vortical flow structures. Sato et al. [2017] proposed backtracing a long-range flow map using semi-lagrangian and integrating pressure gradient along the path to correct velocity. Qu et al. [2019] developed a bidirectional mapping to better prevent dissipation. Based on this, Nabizadeh et al. [2022] combined the concept of flow map with the impulse fluid model [Cortez 1996]. In Neural Flow Map (NFM) [Deng et al. 2023], the authors proposed a neural buffer to memory-efficiently store intermediate velocity fields used for backtracing flow maps with high precision. Besides reducing numeric dissipation in velocity advection, recent work has also explored using flow maps for accurate interface tracking. Bellotti and Theillard [2019] leveraged flow maps to reduce volume loss in two-phase flow simulations, which is incurred by the numeric error in the advection and reinitialization of the level set [Osher and Sethian 1988]. Mercier et al. [2020] proposed combining flow maps with a gradient augmentation interpolation scheme. To reduce the volume loss incurred by reinitialization, Narita and Ando [2022] employed tiled characteristic maps to replace global reinitialization with local reinitialization. Li et al. [2023] developed an accurate quasi-Newton method for reinitialization and built up a high-order framework based on flow maps, which is named GARM-LS.

3 PHYSICAL MODEL

3.1 Two-Phase Flow

We take a sharp-interface model for two-phase flows, where the fluid density exhibits a discontinuous jump at the liquid-air interface. Liquid region and air region can be represented by a signed distance field, which is denoted as φ:

$$\begin{cases} \varphi < 0, & \text{in the liquid region } \Omega_L, \text{ with density } \rho_L, \\ \varphi > 0, & \text{in the air region } \Omega_A, \text{ with density } \rho_A, \\ \varphi = 0, & \text{at the liquid-air interface } \partial\Omega. \end{cases} \tag{1}$$

The flow dynamics are governed by the Euler equations for inviscid, incompressible fluid flow:

$$\begin{cases} \dfrac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\dfrac{1}{\rho}\nabla p + \boldsymbol{g}, & (2) \\ \nabla \cdot \boldsymbol{u} = 0, & (3) \\ [p] = \sigma\kappa \text{ on } \partial\Omega. & (4) \end{cases}$$

Here σ is the surface tension coefficient, κ is the mean curvature, and $\boldsymbol{g}$ is the gravitational acceleration.

3.2 Flow Map

Consider fluid moving according to a spatiotemporal velocity field $\boldsymbol{u}(\boldsymbol{q}, \tau)$ from time 0 to time t. We use $\boldsymbol{X}$ to denote the initial position of a material point and use $\boldsymbol{x}$ to denote the position of the material point at time t. The motion of the fluid in the time period can be represented by forward flow map $\boldsymbol{\phi}$, which is defined as:

$$\begin{cases} \dfrac{\partial \boldsymbol{\phi}(\boldsymbol{X}, \tau)}{\partial \tau} = \boldsymbol{u}[\boldsymbol{\phi}(\boldsymbol{X}, \tau), \tau], \\ \boldsymbol{\phi}(\boldsymbol{X}, 0) = \boldsymbol{X}, \\ \boldsymbol{\phi}(\boldsymbol{X}, t) = \boldsymbol{x}. \end{cases} \tag{5}$$

The inverse flow map $\boldsymbol{\psi}$ is defined as the inverse mapping of the forward flow map:

$$\begin{cases} \boldsymbol{\phi}(\boldsymbol{\psi}(\boldsymbol{x}, \tau), \tau) = \boldsymbol{x}. \\ \boldsymbol{\psi}(\boldsymbol{x}, 0) = \boldsymbol{x}, \\ \boldsymbol{\psi}(\boldsymbol{x}, t) = \boldsymbol{X}. \end{cases} \tag{6}$$

The jacobians of $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are represented by $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{T}}$, respectively.

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\phi}, \tau) = \frac{\partial \boldsymbol{\phi}(\boldsymbol{X}, \tau)}{\partial \boldsymbol{X}}, \ \boldsymbol{\mathcal{T}}(\boldsymbol{x}, \tau) = \frac{\partial \boldsymbol{\psi}(\boldsymbol{x}, \tau)}{\partial \boldsymbol{x}}. \tag{7}$$

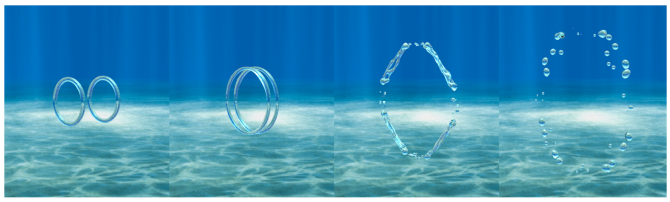


Fig. 4. Colliding bubble rings. The two bubble rings have opposite vorticity at the beginning and then approach and collide with each other.

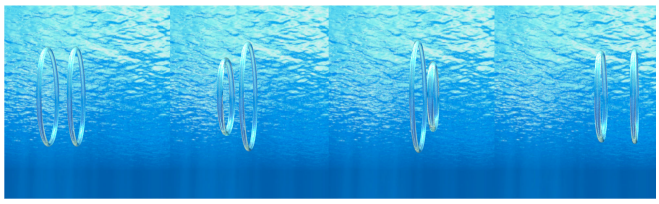


Fig. 5. Leapfrogging bubble rings. Initially, the two bubble rings have aligned vorticity. Induced flow makes them pass through each other repeatedly.

As proven in [Gonzalez and Stuart 2008], the temporal evolution of $\mathcal{F}$ and $\mathcal{T}$ is given by the following equations:

$$\frac{\mathrm{D}\mathcal{F}}{\mathrm{D}t} = \nabla \boldsymbol{u}\mathcal{F}, \ \frac{\mathrm{D}\mathcal{T}}{\mathrm{D}t} = -\mathcal{T}\nabla \boldsymbol{u}. \tag{8}$$

We define path integral operator $\mathcal{L}_a^b$, which maps a spatiotemporal field f to another spatiotemporal field $\mathcal{L}_a^b f$,

$$(\mathcal{L}_a^b f)(\boldsymbol{x}, t) = \int_a^b f(\boldsymbol{\phi}(\boldsymbol{\psi}(\boldsymbol{x}, t), \tau), \tau)\mathrm{d}\tau. \tag{9}$$

Consider a material point r which is located at $\boldsymbol{x}$ at time t. Its initial position is $\boldsymbol{\psi}(\boldsymbol{x}, t)$. From time a to time b, r moves from $\boldsymbol{\phi}(\boldsymbol{\psi}(\boldsymbol{x}, t), a)$ to $\boldsymbol{\phi}(\boldsymbol{\psi}(\boldsymbol{x}, t), b)$. $(\mathcal{L}_a^b f)(\boldsymbol{x}, t)$ represents the integral of the f value on the material point r over the time period.

3.3 Impulse Gauge Two-Phase Flow

For two-phase flows, impulse field $\boldsymbol{m}$ is defined through a gauge transformation

$$\boldsymbol{m} = \boldsymbol{u} + \frac{1}{\rho}\nabla\alpha. \tag{10}$$

Here α is chosen such that (1) $\boldsymbol{m}$ is initially equivalent to $\boldsymbol{u}$, (2) the material derivative of $\boldsymbol{m}$ equals to a stretching term[Cortez 1996],

$$\begin{cases} \boldsymbol{m}(\boldsymbol{x}, 0) = \boldsymbol{u}(\boldsymbol{x}, 0), \\ \dfrac{D\boldsymbol{m}}{Dt} = -(\nabla \boldsymbol{u})^T \boldsymbol{m}. \end{cases} \tag{11}$$

Note that the evolution of $\boldsymbol{u}$ and $\boldsymbol{m}$ are deterministic, thus $\nabla\alpha$ is unique, making the feasible solutions of α only vary by a constant. We derive that (in section A)

$$\alpha(\boldsymbol{x}, t) = \left(\mathcal{L}_0^t(p - \frac{1}{2}\rho|\boldsymbol{u}|^2 + \rho G)\right)(\boldsymbol{x}, t) \tag{12}$$

makes $\boldsymbol{m}$ satisfy Equation (11). Here G denotes the gravitational potential, which gives

$$\boldsymbol{g} = -\nabla G. \tag{13}$$

Across the liquid-air interface, α has a history-dependent jump,

$$[\alpha] = \mathcal{L}_0^t(\sigma\kappa) + (\rho_A - \rho_L) \cdot (\mathcal{L}_0^t G - \frac{1}{2}\mathcal{L}_0^t|\boldsymbol{u}|^2) \text{ on } \partial\Omega. \tag{14}$$

The jump of α at interface $[\alpha]$ is the difference between the unilateral limit of α on the air region and the unilateral limit on the liquid region. At time t, consider an interfacial point $\boldsymbol{x}^I$, an air point $\boldsymbol{x}^A$, and a liquid point $\boldsymbol{x}^L$. $\boldsymbol{x}^A$ and $\boldsymbol{x}^L$ are infinitely close to $\boldsymbol{x}^I$, and $[\alpha](\boldsymbol{x}^I, t)$ can be regarded as the difference between $\alpha(\boldsymbol{x}^A, t)$ and $\alpha(\boldsymbol{x}^L, t)$,

$$[\alpha](\boldsymbol{x}^I, t) = \alpha(\boldsymbol{x}^A, t) - \alpha(\boldsymbol{x}^L, t). \tag{15}$$

Consider material points r^I, r^A, r^L moving in the flow field. At time t, r^I, r^A, r^L are located at $\boldsymbol{x}^I$, $\boldsymbol{x}^A$, $\boldsymbol{x}^L$ respectively. Because t is finite, the distance between r^A and r^I, and the distance between r^L and r^I are still infinitely small in the time period from 0 to t. Therefore, at each time point, the difference between a physical quantity on r^A and r^L equals to the jump of the physical quantity on r^I. Furthermore, the difference between the temporal integral of a physical quantity on r^A and r^L equals to the temporal integral of the physical quantity's jump on r^I. Thus,

$$\alpha(\boldsymbol{x}^A, t) - \alpha(\boldsymbol{x}^L, t) = \left(\mathcal{L}_0^t([p] - \frac{1}{2}[\rho]|\boldsymbol{u}|^2 + [\rho]G)\right)(x^I, t). \tag{16}$$

And equation (14) can be obtained.

4 IMPULSE GHOST FLUID METHOD

A traditional velocity-based solver comprises two primary components: advection and projection. In advection, the velocity field is transported through a frozen background velocity field. In projection, the advected velocity field is made divergence-free by solving a Poisson equation. Similarly, an impulse-based solver updates the impulse to the current time, followed by projecting this impulse onto a divergence-free velocity field. As the mathematical form of the gauge transformation suggests, this projection is also equivalent to solving a Poisson equation. To avoid the problem that the distance between impulse and velocity keeps increasing and causes numeric instability at some point, the impulse is periodically reset to velocity, with the reset time selected as a new zero point (i.e. $t = 0$). We denote this scheme as the reinitialization of impulse, which is commonly used in impulse-based methods.

4.1 Pullback and Backward March

A merit of impulse lies in its purely geometric evolution, which enables the reconstruction of impulse from the initial conditions and flow maps with the pullback operator [Nabizadeh et al. 2022]. By utilizing (8) and (11), one can derive

$$\boldsymbol{m}(\boldsymbol{x}, t) = \mathcal{T}^T \boldsymbol{u}(\boldsymbol{\psi}(\boldsymbol{x}, t), 0). \tag{17}$$

In the work of Nabizadeh et al. [2022], $\boldsymbol{\psi}$ is advected every time step and $\mathcal{T}$ is obtained by taking the finite difference of $\boldsymbol{\psi}$. Deng et al. [2023] proposed using a velocity buffer to backward march $\boldsymbol{\psi}$ and $\mathcal{T}$ with a *Joint-RK4* scheme, which significantly improves the accuracy and is adopted by us.

Here we explain how to numerically backward march $\boldsymbol{\psi}$. Consider a material point that is located at grid point (i, j, k) at time t^n, and we want to find its position at time 0. We have the previous velocity

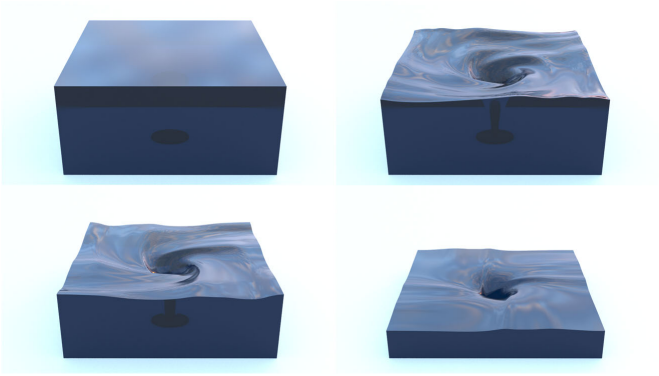


Fig. 6. Sink whirlpool with a hole on the bottom.

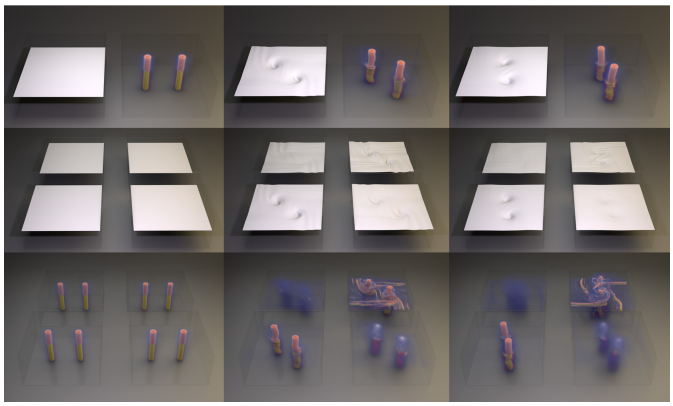


Fig. 7. A pair of vortex tubes crossing liquid-air interface. First row: results of our method. The second row and third row demonstrate the comparisons. Bottom left: our method, top left: standard velocity-based GFM solver, Bottom right: BiMocq^2, Top right: MC+R.

field $\boldsymbol{u}^0, \boldsymbol{u}^1, \cdots, \boldsymbol{u}^{n-1}$ in hand. We can consider the time reverses, the material point starts at the grid point (i, j, k), moves according to the RK4 scheme in $\boldsymbol{u}^{n-1}, \cdots, \boldsymbol{u}^1, \boldsymbol{u}^0$, and eventually marches to its position at time 0.

4.2 Discretized Form for Impulse Projection

We generalize GFM [Fedkiw et al. 1999] from velocity projection to impulse projection. In time step n, we need to project a divergent impulse field $\boldsymbol{m}^n$ to a new divergence-free velocity field $\boldsymbol{u}^n$,

$$\boldsymbol{u}^n = \boldsymbol{m}^n - \frac{1}{\rho}\nabla\alpha^n, \tag{18}$$

Taking the divergence of both sides will yield a Poisson equation:

$$\nabla\cdot\left(\frac{1}{\rho}\nabla\alpha^n\right) = \nabla\cdot\boldsymbol{m}^n. \tag{19}$$

The Poisson equation (19) can be discretized on a MAC grid [Harlow and Welch 1965]. To illustrate, take the 1-D case as an example. We use $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ to denote the center of cell i and cell $i+1$. Suppose $\boldsymbol{x}_i$ is in liquid and $\boldsymbol{x}_{i+1}$ is in air. The interface between the two points is located at $\boldsymbol{x}_i + \theta\Delta x$. At the interface, the left limit of α^n is α_I^n and the right limit is $\alpha_I^n + [\alpha^n](\boldsymbol{x}_I)$. We use a second-order finite difference method to discretize $\nabla\cdot\left(\frac{1}{\rho}\nabla\alpha^n\right)$ on cell i. First, we discretize the divergence operator,

$$\frac{1}{\Delta x}\cdot\left(\frac{1}{\rho}\nabla\alpha^n\right)_{i+1/2} - \frac{1}{\Delta x}\cdot\left(\frac{1}{\rho}\nabla\alpha^n\right)_{i-1/2}. \tag{20}$$

The discretization for $\left(\frac{1}{\rho}\nabla\alpha^n\right)_{i-\frac{1}{2}}$ is straightforward,

$$\frac{1}{\rho_L}\frac{\alpha_i^n - \alpha_{i-1}^n}{\Delta x}. \tag{21}$$

At the liquid-air interface between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, ρ and α are discontinuous, but $\frac{1}{\rho}\nabla\alpha^n$ is continuous. Following the logic of GFM, $\frac{1}{\rho}\nabla\alpha^n$ is assumed to be the same for liquid and air between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, which gives

$$\left(\frac{1}{\rho}\nabla\alpha^n\right)_{i+1/2} = \frac{1}{\rho_L}\frac{\alpha_I^n - \alpha_i^n}{\theta\Delta x} = \frac{1}{\rho_A}\frac{\alpha_{i+1}^n - \alpha_I^n - [\alpha^n](\boldsymbol{x}_I)}{(1-\theta)\Delta x}. \tag{22}$$

It can be derived that

$$\left(\frac{1}{\rho}\nabla\alpha^n\right)_{i+1/2} = \frac{1}{\hat{\rho}}\frac{\alpha_{i+1}^n - \alpha_i^n - [\alpha^n](\boldsymbol{x}_I)}{\Delta x} \tag{23}$$

where

$$\hat{\rho} = \theta\rho_L + (1-\theta)\rho_A. \tag{24}$$

Then on cell i, equation (19) can be discretized as

$$\frac{1}{\hat{\rho}}\frac{\alpha_{i+1}^n - \alpha_i^n}{\Delta x^2} - \frac{1}{\rho_L}\frac{\alpha_i^n - \alpha_{i-1}^n}{\Delta x^2} = \frac{m_{i+1/2}^n - m_{i-1/2}^n}{\Delta x} + \frac{1}{\hat{\rho}}\frac{[\alpha^n](\boldsymbol{x}_I)}{\Delta x^2}. \tag{25}$$

4.3 Path Integral Projection

Before solving equation (25), the jump term $[\alpha^n]$ has to be calculated. A straightforward method involves maintaining a velocity buffer and a level set buffer and using backward marched inverse flow to obtain path integral of $\sigma\kappa$, $|\boldsymbol{u}|^2$ and G. However, this direct approach is impractical. For accurate calculations, $\boldsymbol{\psi}$ must be on the exact interface during backward marching. In theory, every interfacial point should consistently reside on the interface. However, numerical errors can cause $\boldsymbol{\psi}$ to deviate from the interface. Making things worse, curvature varies rapidly as the evaluation point deviates, resulting in low accuracy in evaluating the accumulation of surface tension.

To address this challenge, we propose a two-step approach named Path Integral Projection. Note that α^n can be separated into two parts,

$$\alpha^n = \mathcal{L}_{t^{n-1}}^{t^n} p + \left(\mathcal{L}_0^{t^{n-1}} p - \frac{1}{2}\rho\mathcal{L}_0^{t^n}|\boldsymbol{u}|^2 + \rho\mathcal{L}_0^{t^n} G\right). \tag{26}$$

$\mathcal{L}_{t^{n-1}}^{t^n} p$ can be discretized as $p^n\Delta t$, which corresponds to what is projected out in a traditional velocity-based solver in time step n. And if we maintain a pressure buffer, $\left(\mathcal{L}_0^{t^{n-1}} p - \frac{1}{2}\rho\mathcal{L}_0^{t^n}|\boldsymbol{u}|^2 + \rho\mathcal{L}_0^{t^n} G\right)$ can be obtained from historical information and can be calculated with our Path Integral scheme (Algorithm 1). Therefore, we can first

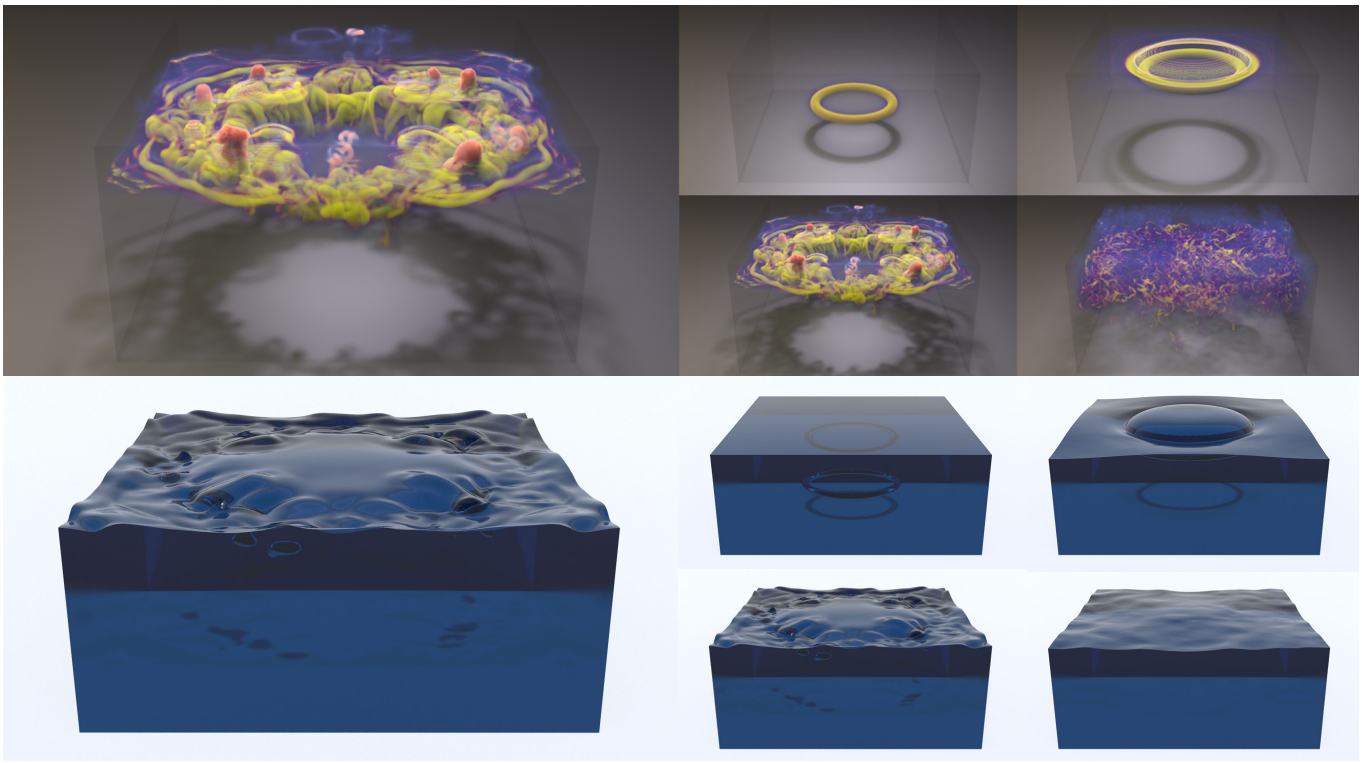


Fig. 8. A rising bubble ring hitting the liquid-air interface. Top: Visualization of vorticity in two phases using different colormaps. Bottom: Interface.

calculate

$$\boldsymbol{u}^* = \boldsymbol{m}^n - \frac{1}{\rho}\nabla\left(\mathcal{L}_0^{t^{n-1}} p - \frac{1}{2}\rho\mathcal{L}_0^{t^n}|\boldsymbol{u}|^2 + \rho\mathcal{L}_0^{t^n} G\right), \tag{27}$$

then perform a velocity projection for $\boldsymbol{u}^*$. $\boldsymbol{u}^*$ corresponds to the advected velocity in a traditional advection-projection velocity-based solver. In this way, we convert the history-dependent jump term into a geometry-dependent jump term. The projection from $\boldsymbol{u}^*$ to $\boldsymbol{u}^n$ is expressed as

$$\begin{cases} \boldsymbol{u}^n = \boldsymbol{u}^* - \Delta t\dfrac{1}{\rho}\nabla p^n. & (28) \\ \Delta t\nabla\cdot\left(\dfrac{1}{\rho}\nabla p^n\right) = \nabla\cdot\boldsymbol{u}^*. & (29) \end{cases}$$

In the 1D-example above, the equation (29) can be discretized on cell i as

$$\frac{\Delta t}{\hat{\rho}}\frac{p_{i+1}^n - p_i^n}{\Delta x^2} - \frac{\Delta t}{\rho_L}\frac{p_i^n - p_{i-1}^n}{\Delta x^2} = \frac{u_{i+1/2}^* - u_{i-1/2}^*}{\Delta x} + \frac{\Delta t}{\hat{\rho}}\frac{\sigma\kappa}{\Delta x^2}. \tag{30}$$

If the distance between a cell and the interface is less than ϵ, we designate the cell as a narrow-band cell. Faces of such cells are marked as narrow-band faces. On these narrow-band faces, instead of calculating $\boldsymbol{u}^*$ from $\boldsymbol{m}^n$, we opt to advect $\boldsymbol{u}^{n-1}$ to obtain $\boldsymbol{u}^*$. This modification is crucial for two reasons. First, it avoids the large velocity gradients near the liquid-air interface, which can lead to numerical instability when computing $\boldsymbol{m}^n$. Second, the path integral used to calculate $\mathcal{L}_0^{t^{n-1}} p$ necessitates the interpolation of p. Interpolation near the interface may inadvertently involve both liquid and air points, leading to invalid results due to the discontinuous nature of p across the liquid-air interface.

5 BIDIRECTIONAL-MARCHED-FLOW-MAP LEVEL SET

The material derivative of the signed distance field φ is 0. Therefore, we can use $\boldsymbol{\psi}$ to pull back φ:

$$\varphi(\boldsymbol{x}, t) = \varphi(\boldsymbol{\psi}(\boldsymbol{x}, t), 0). \tag{31}$$

Previous works have explored the evolution of φ using an inverse flow map [Li et al. 2023] and a bidirectional flow map [Qu et al. 2019]. These methods advect $\boldsymbol{\psi}$ every time step. In contrast to these approaches, our method utilizes a velocity buffer to backward march $\boldsymbol{\psi}$. Our experiments (Fig. 11, 12) demonstrate that backward marching significantly improves the accuracy of tracking $\boldsymbol{\psi}$ and level set in a vortical velocity field.

Advection of a field introduces frequent interpolation, which is equivalent to applying a low-pass filter to the field. If φ is directly advected, then the numeric diffusion is considerable. If we advect $\boldsymbol{\psi}$ every time step and use $\boldsymbol{\psi}$ to pullback φ, then the numeric diffusion for φ is reduced because the frequency of interpolating φ is reduced. However, some amount of numeric diffusion still exists, because advecting $\boldsymbol{\psi}$ can cause numeric diffusion in $\boldsymbol{\psi}$, which influences the pullback mapping.

We use the fourth-order quasi-Newton method proposed by Li et al. [2023] for reinitialization of the level set. The forward flow map, $\boldsymbol{\phi}$, is marched every time step and utilized for back-and-forth

Algorithm 1 Path Integral

Input: position $\boldsymbol{x}$,
time step buffer $< \Delta t^1, \Delta t^2, \cdots, \Delta t^n >$,
velocity buffer $< \boldsymbol{u}^0, \boldsymbol{u}^1, \cdots, \boldsymbol{u}^{n-1} >$,
pressure buffer $< p^1, p^2, \cdots, p^{n-1} >$.
Output: $\left(\mathcal{L}_0^{t^n} G\right)(\boldsymbol{x}, t^n)$,
$\left(\mathcal{L}_0^{t^n} |\boldsymbol{u}|^2\right)(\boldsymbol{x}, t^n)$,
$\left(\mathcal{L}_0^{t^{n-1}} p\right)(\boldsymbol{x}, t^n)$.

ret1, ret2, ret3 $\leftarrow 0, 0, 0$;
for j in **reversed(range**(n)**)** **do**
  ret1 += $-\Delta t^{j+1} \cdot \boldsymbol{g} \cdot \boldsymbol{x}$;
  if $j < n-1$ **then**
    ret3 += $\Delta t^{j+1} \cdot$ **Interpolate**$(p^{j+1}, \boldsymbol{x})$;
  end if
  $\boldsymbol{u}_1 \leftarrow$ **Interpolate**$(\boldsymbol{u}^j, \boldsymbol{x})$;
  $\boldsymbol{x}_1 \leftarrow \boldsymbol{x} - 0.5 \cdot \Delta t^{j+1} \cdot \boldsymbol{u}_1$;
  $\boldsymbol{u}_2 \leftarrow$ **Interpolate**$(\boldsymbol{u}^j, \boldsymbol{x}_1)$;
  $\boldsymbol{x}_2 \leftarrow \boldsymbol{x} - 0.5 \cdot \Delta t^{j+1} \cdot \boldsymbol{u}_2$;
  $\boldsymbol{u}_3 \leftarrow$ **Interpolate**$(\boldsymbol{u}^j, \boldsymbol{x}_2)$;
  $\boldsymbol{x}_3 \leftarrow \boldsymbol{x} - \Delta t^{j+1} \cdot \boldsymbol{u}_3$;
  $\boldsymbol{u}_4 \leftarrow$ **Interpolate**$(\boldsymbol{u}^j, \boldsymbol{x}_3)$;
  $\boldsymbol{x} \leftarrow \boldsymbol{x} - \frac{1}{6} \cdot \Delta t^{j+1} \cdot (\boldsymbol{u}_1 + 2\boldsymbol{u}_2 + 2\boldsymbol{u}_3 + \boldsymbol{u}_4)$;
  ret2 += $\frac{1}{6} \cdot \Delta t^{j+1} \cdot (|\boldsymbol{u}_1|^2 + 2|\boldsymbol{u}_2|^2 + 2|\boldsymbol{u}_3|^2 + |\boldsymbol{u}_4|^2)$;
end for
return ret1, ret2, ret3

Algorithm 2 BMFM-LS

Input:initial level set φ^0,
forward flow map $\boldsymbol{\phi}$,
time step buffer $\Delta T =< \Delta t^1, \Delta t^2, \cdots, \Delta t^n >$,
velocity buffer $\boldsymbol{U} =< \boldsymbol{u}^0, \boldsymbol{u}^1, \cdots, \boldsymbol{u}^{n-1} >$.
Output: φ^n.

$\boldsymbol{\psi} \leftarrow$ **id**;
$\boldsymbol{\psi} \leftarrow$ RK4_Backward_March$(\boldsymbol{\psi}, \boldsymbol{U}, \Delta T)$;
$\boldsymbol{\phi} \leftarrow$ RK4$(\boldsymbol{\phi}, \boldsymbol{u}^{n-1}, \Delta t^n)$;
$\hat{\varphi} \leftarrow \varphi^0(\boldsymbol{\psi})$;
$\hat{\varphi}^0 \leftarrow \hat{\varphi}(\boldsymbol{\phi})$;
$e \leftarrow (\hat{\varphi}^0 - \varphi^0)/2$;
$\varphi^n \leftarrow \hat{\varphi} - e(\boldsymbol{\psi})$;
if reinitialize **then**
  $\boldsymbol{\phi} \leftarrow$ **id**;
  $\varphi^0 \leftarrow$ reinitialize(φ^n);
  Clear $\Delta T, \boldsymbol{U}$.
end if

error correction [Dupont and Liu 2003]. Instead of being advected, both $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are marched through the velocity field, leading us to designate our approach as the Bidirectional-Marched-Flow-Map Level Set (BMFM-LS).

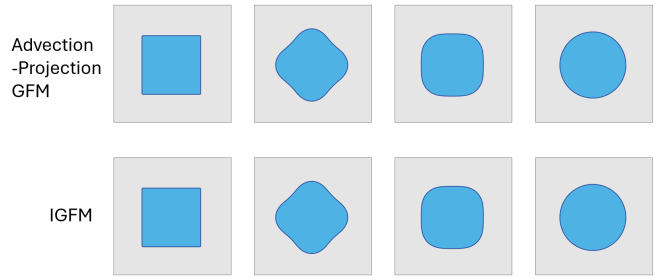


Fig. 9. Comparison experiment: A liquid square oscillating due to surface tension. Top: Advection-Projection GFM. Bottom: IGFM.

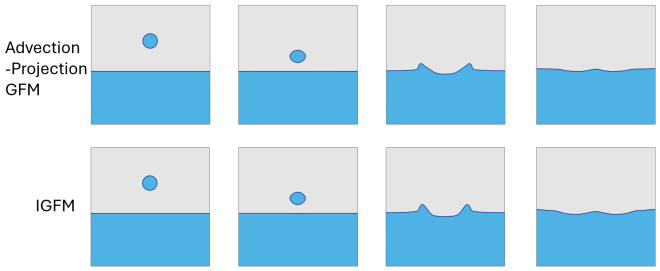


Fig. 10. Comparison experiment: A liquid drop falling into a tank. Top: Advection-Projection GFM. Bottom: IGFM.

6 TIME INTEGRATION

We summarize the time integration scheme of our method in Algorithm 3. Extending to the previous discussion, the second-order *midpoint method* is employed to ensure a symmetric march of bidirectional flow maps.

7 EXPERIMENTS

In this section, we present 6 examples to evaluate our method and 7 examples to demonstrate its application. All examples are performed on a desktop equipped with a NVIDIA RTX 4080. We implemented our algorithm in Taichi [Hu et al. 2019] with a CUDA backend. We use an AMGPCG [Stüben 2001] solver for projection. The narrow-band ϵ ranges from cell size Δx to $3\Delta x$ in our experiments. Unless otherwise specified, the impulse, level set, flow map are reinitialized every 5 steps.

7.1 Evaluation

Oscillating Square. We simulate an oscillating liquid square immersed in air. Driven by surface tension, the square is expected to oscillate. We compare the results of the standard Advection-Projection GFM and IGFM in Fig. 9. IGFM demonstrates consistent oscillation frequency with the standard method and stabilizes into a circular shape.

Droptank. We simulate a liquid drop falling into a tank using the Advection-Projection GFM and IGFM, with both producing congruent results. As shown in Fig. 10 the liquid drop undergoes

Algorithm 3 Time Integration

Input: step number since last reinitialization: n
Forward flow map $\boldsymbol{\phi}$, its jacobian $\mathcal{F}$
Initial level set φ^0, Current level set φ^{n-1}
time step buffer $\Delta T =< \Delta t^1, \Delta t^2, \cdots, \Delta t^n >$
velocity buffer $\boldsymbol{U} =< \boldsymbol{u}^{1/2}, \boldsymbol{u}^{3/2}, \cdots, \boldsymbol{u}^{n-3/2} >$
Initial velocity $\boldsymbol{u}^0$, velocity at the end of last step $\boldsymbol{u}^{n-1}$
pressure buffer $P =< p^1, p^2, \cdots, p^{n-1} >$.

$\varphi^{n-1/2} \leftarrow \text{Advect}(\varphi^{n-1}, \boldsymbol{u}^{n-1}, \Delta t^n/2)$; ▷ midpoint method.
$\boldsymbol{u}^{**} \leftarrow \text{Advect}(\boldsymbol{u}^{n-1}, \boldsymbol{u}^{n-1}, \Delta t^n/2)$;
$\hat{\boldsymbol{u}}^{**} \leftarrow \text{Gravity}(\boldsymbol{u}^{**}, \boldsymbol{g}, \Delta t^n/2)$;
$\boldsymbol{u}^{n-1/2}, p^{n-1/2} \leftarrow \text{Project}(\hat{\boldsymbol{u}}^{**}, \varphi^{n-1/2})$;
Store $\boldsymbol{u}^{n-1/2}$ in $\boldsymbol{U}$;
$\boldsymbol{\psi} \leftarrow \mathbf{id}$, $\boldsymbol{\mathcal{T}} \leftarrow \boldsymbol{I}$; ▷ march bidirectional flow map
$\boldsymbol{\psi}, \boldsymbol{\mathcal{T}} \leftarrow \text{Joint\_RK4\_Backward\_March}(\boldsymbol{\psi}, \boldsymbol{\mathcal{T}}, \boldsymbol{U}, \Delta T)$;
$\boldsymbol{\phi}, \boldsymbol{\mathcal{F}} \leftarrow \text{Joint\_RK4}(\boldsymbol{\phi}, \boldsymbol{\mathcal{F}}, \boldsymbol{u}^{n-1/2}, \Delta t^n)$;
$\hat{\varphi} \leftarrow \varphi^0(\boldsymbol{\psi})$; ▷ BMFM-LS
$\hat{\varphi}^0 \leftarrow \hat{\varphi}(\boldsymbol{\phi})$;
$e \leftarrow (\hat{\varphi}^0 - \varphi^0)/2$
$\varphi^n \leftarrow \hat{\varphi} - e(\boldsymbol{\psi})$;
$\hat{\boldsymbol{m}} \leftarrow \boldsymbol{\mathcal{T}}^T \boldsymbol{u}^0(\boldsymbol{\psi})$; ▷ pullback impulse
$\hat{\boldsymbol{u}}^0 \leftarrow \boldsymbol{\mathcal{F}}^T \hat{\boldsymbol{m}}(\boldsymbol{\phi})$; ▷ error correction
$\boldsymbol{e} \leftarrow (\hat{\boldsymbol{u}}^0 - \boldsymbol{u}^0)/2$;
$\boldsymbol{m}^n \leftarrow \hat{\boldsymbol{m}} - \boldsymbol{\mathcal{T}}^T \boldsymbol{e}(\boldsymbol{\psi})$;
$\boldsymbol{u}^* \leftarrow \text{Advect}(\boldsymbol{u}^{n-1}, \boldsymbol{u}^{n-1/2}, \Delta t^n)$ in narrow band;
$\mathcal{L}_0^{t^n} G, \mathcal{L}_0^{t^n} |\boldsymbol{u}|^2, \mathcal{L}_0^{t^{n-1}} p \leftarrow \text{Path\_Integral}(\boldsymbol{U}, \boldsymbol{P}, \Delta T)$;
$\boldsymbol{u}^* \leftarrow \boldsymbol{m}^n - \frac{1}{\rho}\nabla\left(\mathcal{L}_0^{t^{n-1}} p - \frac{1}{2}\rho \mathcal{L}_0^{t^n} |\boldsymbol{u}|^2 + \rho \mathcal{L}_0^{t^n} G\right)$ out of narrow band;
$\boldsymbol{u}^n, p^n \leftarrow \text{Project}(\boldsymbol{u}^*, \varphi^n)$;
Store p^n in $\boldsymbol{P}$;
$\Delta t^{n+1} \leftarrow \text{CFL}(\boldsymbol{u}^n)$;
Store Δt^{n+1} in ΔT;
if reinitialize **then**
  $\boldsymbol{\phi} \leftarrow \mathbf{id}$, $\boldsymbol{\mathcal{F}} \leftarrow \boldsymbol{I}$;
  $\varphi^0 \leftarrow \text{reinitialize}(\varphi^n)$;$\boldsymbol{u}^0 \leftarrow \boldsymbol{u}^n$;
  Clear $\boldsymbol{U}, \boldsymbol{P}$;
  $\Delta T \leftarrow < \Delta t^{n+1} >$; $n \leftarrow 1$;
end if

deformation during its descent, flattening at the bottom, which typifies the dynamics observed in two-phase flow simulations.

Vortex Ring Reflection. Similar to the reflection of light, a vortex ring exhibits reflective behavior when it obliquely moves towards the liquid-air interface at a moderate angle, a phenomenon consistently observed in real-world experiments[Su and Zhang 2023]. As shown in Fig. 3, our method successfully captures this phenomenon.

Flow Map Test: Single Vortex. We employ various schemes to evolve $\boldsymbol{\psi}$ within a steady vortical velocity field. A single vortex is at the center, and the angular velocity ω varies with radius r:

$$\omega = k/(r+a)^2 \cdot \left[1 - \exp\left(-(r+a)^2/b^2\right)\right] \tag{32}$$

Figure	Resolution	ρ_L/ρ_A	Simulation time/step
Fig. 9	128×128	1000	0.021s
Fig. 10	128×128	1000	0.023s
Fig. 6	$256 \times 128 \times 256$	1000	0.963s
Fig. 3	$384 \times 128 \times 128$	1000	0.697s
Fig. 7	$128 \times 128 \times 128$	1000	0.242s
Fig. 14	$256 \times 192 \times 256$	100	1.153s
Fig. 5	$384 \times 256 \times 256$	100	2.676s
Fig. 4	$256 \times 256 \times 256$	100	1.814s
Fig. 2	$256 \times 64 \times 128$	1000	0.281s
Fig. 8	$256 \times 128 \times 256$	100	0.972s
Fig. 15	$128 \times 64 \times 128$	1000	0.113s

Table 1. Statistics. Simulation time is measured on a Nvidia RTX 4080 GPU and averaged. The CFL number is 1.0 for all examples.

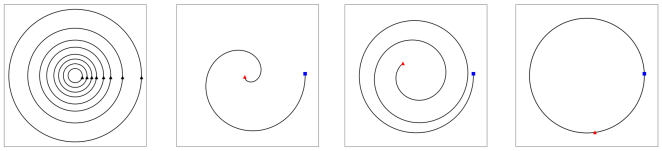


Fig. 11. Inverse flow map $\boldsymbol{\psi}$ is evolved in the steady velocity field of a single vortex centered at (0.5, 0.5). (a) shows the streamlines of the single vortex. (b)(c)(d) record the trails of $\boldsymbol{\psi}(\boldsymbol{x}, \cdot)$ evolved with different schemes. The blue square box denotes $\boldsymbol{x} = (0.617, 0.5)$, and the red triangle denotes $\boldsymbol{\psi}(\boldsymbol{x}, t)$. (b) advection with DMC. (c) advection with TVD-RK3 and bicubic Hermite interpolation. (d) RK4 Backward March.

with $k = 0.005, a = 0.000001, b = 0.02$. The domain is a 1.0-unit square which is discretized using a 128×128 grid. We test the advection of $\boldsymbol{\psi}$ using various numerical methods: advection with DMC [Cho et al. 2018], advection with TVD-RK3 and bicubic Hermite interpolation, and RK4 Backward March, corresponding to BiMocq^2, GARM-LS, and BMFM-LS, respectively. Ideally, $\boldsymbol{x}$ and $\boldsymbol{\psi}(\boldsymbol{x}, t)$ should align the same circular orbit. However, as shown in Fig. 11, when $\boldsymbol{\psi}$ is advected, even with a high-order scheme, interpolation error can cause $\boldsymbol{\psi}(\boldsymbol{x}, t)$ to deviate from the circular orbit where $\boldsymbol{x}$ is located. In contrast, backward marched $\boldsymbol{\psi}(\boldsymbol{x}, t)$ remains perfectly on the circular orbit.

Level Set Test: Vortex Ring. We evolve the level set of a torus within a steady, divergence-free velocity field of a vortex ring, where the major radius and position of the vortex ring match those of the torus. The domain is discretized with a $128 \times 128 \times 128$ grid and the reinitialization interval is 10. Ideally, the volume of the torus should be preserved during evolution. However, as depicted in Fig. 13, significant volume loss occurs when the level set of the torus is evolved using BiMocq^2 and GARM-LS. This is because inaccurate tracking of $\boldsymbol{\psi}$ can lead to an error in computing φ during pullback operations. Thus, precise tracking of the level set of the torus within the velocity field of a vortex ring is crucial for accurately simulating toroidal bubble rings.

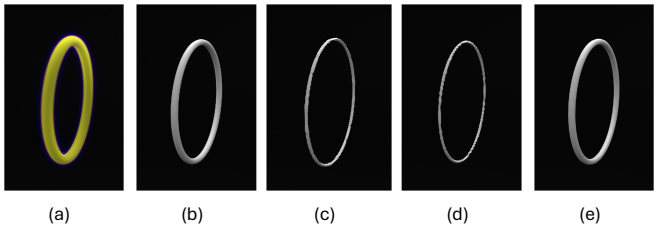


Fig. 12. The level set of a torus is evolved in the steady velocity field of a vortex ring. (a) Vorticity. (b) Initial State. (c) BiMocq^2. (d) GARM-LS. (e) BMFM-LS.

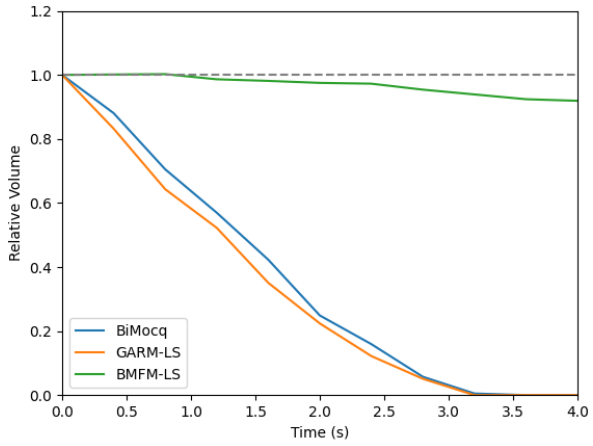


Fig. 13. Relative volume of the level set evolved in the velocity field of a vortex ring. Results of BiMocq^2, GARM-LS and BMFM-LS are plotted.

Vortex Pair. We simulate a pair of vortex tubes crossing the liquid-air interface, creating whirlpools on the interface, and causing mutual spin between the tubes. In the first row of Fig. 7, we visualize the simulation results of our method, coloring the vorticity in the air using the coolwarm map, and the vorticity in the liquid using the plasma map, a convention maintained in subsequent examples. In the second row and third row, we compare our method (bottom left) with the Advection-Projection GFM(top left), Bimocq^2 (bottom right) and MC+R [Selle et al. 2008; Zehnder et al. 2018] (top right). We selected MC+R and Bimocq^2 due to their exceptional abilities to preserve vortical structures in smoke simulation. These methods are velocity-based Eulerian approaches, thus allowing for generalization to two-phase flow simulations by integrating them into an Advection-Projection GFM solver. We opted against Covector Fluids[Nabizadeh et al. 2022] because it does not naturally fit into two-phase flow simulations. In the simulations, vorticity dissipates rapidly using the standard GFM solver. The vortical structure is better preserved by BiMocq^2. Instability near the interface is observed in MC+R. In contrast, our method not only generates a smooth interface but also more effectively preserves the vortical structures. As shown in Table 2, our approach is more expensive than the Advection-Projection GFM. The extra time cost mainly comes from two aspects: (1) In each step, we must march the inverse flow map backward and perform a Lagrangian path integral, and (2) solve an additional Poisson for the midpoint. Table 2 shows that (2) dominates the computational cost. Therefore, the running time of our method is about twice as expensive as the Avection-Projection GFM.

Average Time Cost per Step		
Method	Advection-Projection GFM	IGFM
Time Cost	0.126s	0.242s
Time Breakdown of IGFM		
Projection	Backward March & Path Integral	Level Set Reinit & etc
86.7%	2.4%	10.9%

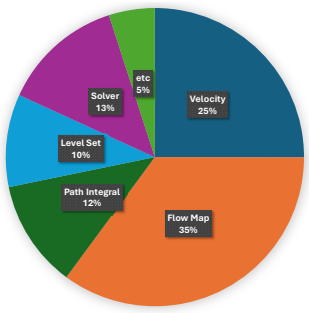
Table 2. Time Comparison & Time Breakdown for Fig. 7.

7.2 Application

Sink Whirlpool. Fig. 6 shows the simulation of a sink whirlpool. Initiated with a rotational velocity field and a hole at the bottom, detailed swirling patterns are observable on the interface.

Rising Bubbles. Fig. 14 shows numerous small bubbles rising within a vortical initial velocity field. The bubbles, randomly generated in the green region, vary in radius. Due to surface tension, smaller bubbles retain a spherical shape, while larger bubbles exhibit noticeable deformation. Notably, the volume of the smaller bubbles is preserved without employing any specific volume control methods, and the vortical structure of the velocity field is maintained throughout the process.

Leapfrogging Bubble Rings. Preservation of vortical structures and accurate tracking of level set enable the simulation of bubble rings. Fig. 5 displays the leapfrogging behavior of two bubble rings. Initially, these identical rings are coaxial. As a result of the induced flow, where the dynamics of one ring influence the other, the two bubble rings repeatedly pass through each other. Here we demonstrate the memory breakdown of our method for this experiment, where the total memory cost is 11.22GB. To pull-back impulse and level set, we need to store the forward/inverse flow maps on mac grid centers and faces as well as their gradients. Also, we need to allocate a buffer for the history of velocities, which leads to extra memory overhead.



Colliding Bubble Rings. Fig. 4 shows the collision between two coaxial bubble rings. The two bubble rings initially possess opposite vorticities. Consequently, the induced flows lead to the rings approaching and eventually colliding with each other. Following

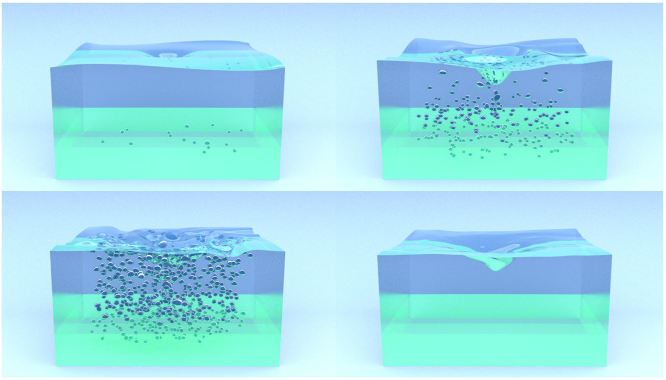


Fig. 14. Rising bubbles in a vortical velocity field. Bubbles are randomly generated in the green region.

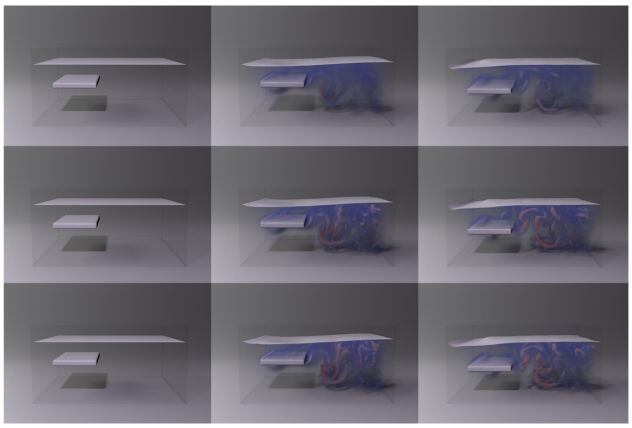


Fig. 15. A heaving board in an inlet. Top row: reinitialize every time step. Middle row: reinitialize every 5 steps. Bottom row: reinitialize every 10 steps.

the collision and subsequent breakup of the rings, small spherical bubbles form as a result of surface tension.

Swinging Fishtail. Fig. 2 illustrates a swinging fishtail in an incoming flow, where half of the fishtail is immersed in air, and the other half is in liquid. The tangential velocity difference on the solid boundary generates rich vortex filaments in both air and liquid.

Rising Bubble Ring. Fig. 8 displays a toroidal bubble ring colliding with the liquid-air interface. As the bubble ring rises and approaches the interface, its major radius grows. Due to surface tension, the bubble ring disintegrates into smaller bubbles that float towards the interface. Upon impact with the interface, the vortex ring generates circular ripples, and the secondary vortex filaments induce a pattern of bumps and dips on the interface.

Heaving Board. Fig. 15 shows a heaving board in an incoming flow. Controlled by a sinuous signal, the board heaves up and down periodically. The interaction between the flow and the board's movement generates surface waves, and vortex shedding can be observed at the trailing edge of the board. As an ablation study, we simulate the process using IGFM with varying reinitialization intervals. The result shows that the simulation with a longer reinitialization interval has less dissipation and preserves vortical structures better.

8 DISCUSSIONS AND FUTURE WORK

In conclusion, we propose a novel impulse ghost fluid method to simulate two-phase flows, capturing complex interactions between vorticity and interfaces. We devise a path integral scheme employing spatiotemporal buffers to tackle the history-dependent jump condition in impulse projection. In addition, we devised a bidirectional flow map scheme to preserve small volumes of the level set interface during its evolution.

Our method stems from the GFM, which features a division of density in pressure projection. Besides GFM, researchers have also explored enforcing the incompressibility for air without solving for the air part. For example, Ando et al. [2015] adopts a stream function formulation. While the projected velocity field may have a small divergence due to numerical tolerance, the stream function solver guarantees that the resulting velocity field is completely divergence-free. Also, the stream function solver avoids the division of density in pressure projection. Previous works using GFM (e.g., MultiFLIP [Boyd and Bridson 2012]) have demonstrated examples with a density ratio of 1000: 1. However, a large density ratio leads to a large spectral condition number of the matrix in pressure projection, slowing down the convergence of the Conjugate Gradient method. We adopt an AMGPCG solver with Galerkin coarsening and multi-color Gauss-Seidel smoother to accelerate the convergence. Using a constant interpolation between different levels, the solver can be implemented in an efficient matrix-free style.

Our approach has several limitations. First, viscosity was not included in our current framework. Although gauge transformation can be defined in a viscous setting, an extra term is introduced to the evolution of impulse, hindering reconstruction of impulse with flow maps and initial condition. Moreover, viscosity will add extra complexities to the jump conditions with gauge variables (e.g., see [Saye 2016]), which requires further investigation on devising effective ghost fluid schemes to handle viscosity on the interface. Second, solid boundaries were treated in a simple way in our simulator. It would be necessary to incorporate more advanced treatments such as the high-order cutting-cell method [Ng et al. 2009] to handle solid boundaries in our ghost fluid framework. Last, our current implementation is not memory efficient. In memory-limited cases, one may adopt the neural representation proposed by Deng et al. [2023] for compression, at the cost of extra training time. How to maintain the buffer in an economical way in terms of its spatial and temporal cost is worth investigating. In our future work, we will further explore in these directions to enhance the current two-phase ghost fluid solver and tackle more challenging interface-vorticity interaction phenomena. We are particularly motivated by simulating complex solid-vortex-interface interaction processes such as fish cross-interface swimming.

ACKNOWLEDGEMENTS

We express our gratitude to the anonymous reviewers for their insightful feedback. We thank Tsinghua University and Shanghai Qi Zhi Institute for their support. We also thank Mengdi Wang, Duowen Chen and Zhiqi Li for their insightful discussion. Georgia Tech authors also acknowledge NSF IIS #2433322, ECCS #2318814, CAREER #2433307, IIS #2106733, OISE #2433313, and CNS #1919647 for funding support. We credit the Houdini education license for video animations.

A EVOLUTION OF IMPULSE

Here we prove equation (12) makes $\boldsymbol{m}$ satisfy equation (11). By the definition of $\mathcal{L}$ and α,

$$\frac{\mathrm{D}\alpha}{\mathrm{D}t} = p - \frac{1}{2}\rho|\boldsymbol{u}|^2 + \rho G.$$

Expand the material derivative of $\nabla\alpha$ as

$$\begin{aligned}\frac{\mathrm{D}(\nabla\alpha)}{\mathrm{D}t} &= \nabla\left(\frac{\mathrm{D}\alpha}{\mathrm{D}t}\right) - (\nabla\boldsymbol{u})^T\nabla\alpha \\ &= \nabla p - \rho(\nabla\boldsymbol{u})^T\boldsymbol{u} - \rho\boldsymbol{g} - (\nabla\boldsymbol{u})^T\nabla\alpha.\end{aligned}$$

By substituting equation (10) into the LHS of equation (11)

$$\begin{aligned}\frac{\mathrm{D}\boldsymbol{m}}{\mathrm{D}t} &= \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} + \frac{1}{\rho}\frac{\mathrm{D}(\nabla\alpha)}{\mathrm{D}t} \\ &= -\frac{1}{\rho}\nabla p + \boldsymbol{g} + \frac{1}{\rho}\nabla p - \boldsymbol{g} - (\nabla\boldsymbol{u})^T\boldsymbol{u} - \frac{1}{\rho}(\nabla\boldsymbol{u})^T\nabla\alpha \\ &= -(\nabla\boldsymbol{u})^T(\boldsymbol{u} + \frac{1}{\rho}\nabla\alpha) = -(\nabla\boldsymbol{u})^T\boldsymbol{m}.\end{aligned}$$

REFERENCES

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2015. A stream function solver for liquid simulations. *ACM Trans. Graph.* 34, 4, Article 53 (2015), 9 pages.

Thomas Bellotti and Maxime Theillard. 2019. A coupled level-set and reference map method for interface representation with applications to two-phase flows simulation. *J. Comput. Phys.* 392 (2019), 266–290.

Landon Boyd and Robert Bridson. 2012. MultiFLIP for energetic two-phase fluid simulation. *ACM Trans. Graph.* 31, 2, Article 16 (2012), 12 pages.

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.

TF Buttke. 1992. Lagrangian numerical methods which preserve the Hamiltonian structure of incompressible fluid flow. (1992).

Tomas F Buttke. 1993. Velicity methods: Lagrangian numerical methods which preserve the Hamiltonian structure of incompressible fluid flow. In *Vortex flows and related numerical methods*. Springer, 39–57.

Thomas F Buttke and Alexandre J Chorin. 1993. Turbulence calculations in magnetization variables. *Applied numerical mathematics* 12, 1-3 (1993), 47–54.

Chung-Ki Cho, Byungjoon Lee, and Seongjai Kim. 2018. Dual-Mesh Characteristics for Particle-Mesh Methods for the Simulation of Convection-Dominated Flows. *SIAM Journal on Scientific Computing* 40, 3 (2018), A1763–A1783.

Ricardo Cortez. 1996. An impulse-based approximation of fluid motion due to boundary forces. *J. Comput. Phys.* 123, 2 (1996), 341–353.

Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023. Fluid Simulation on Neural Flow Maps. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–21.

Todd F. Dupont and Yingjie Liu. 2003. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys.* 190, 1 (2003), 311–324.

Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. 1999. A Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method). *J. Comput. Phys.* 152, 2 (1999), 457–492.

Fan Feng, Jinyuan Liu, Shiying Xiong, Shuqi Yang, Yaorui Zhang, and Bo Zhu. 2023. Impulse Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2023), 3081–3092.

Ryan Goldade, Mridul Aanjaneya, and Christopher Batty. 2020. Constraint bubbles and affine regions: reduced fluid models for efficient immersed bubbles and flexible spatial coarsening. *ACM Trans. Graph.* 39, 4, Article 43 (2020), 15 pages.

Oscar Gonzalez and Andrew M. Stuart. 2008. *A First Course in Continuum Mechanics*. Cambridge University Press.

Toshiya Hachisuka. 2005. Combined Lagrangian-Eulerian approach for accurate advection. In *ACM SIGGRAPH 2005 Posters* (Los Angeles, California) *(SIGGRAPH '05)*. 114–es.

Francis H. Harlow and J. Eddie Welch. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids* 8 (1965), 2182–2189.

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

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Trans. Graph.* 38, 6, Article 201 (2019), 16 pages.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Trans. Graph.* 34, 4, Article 51 (2015), 10 pages.

Byungmoon Kim, Yingjie Liu, Ignacio Llamas, Xiangmin Jiao, and Jarek Rossignac. 2007. Simulation of bubbles in foam with the volume control method. In *ACM SIGGRAPH 2007 Papers*. 98–es.

Wei Li and Mathieu Desbrun. 2023. Fluid-Solid Coupling in Kinetic Two-Phase Flow Simulation. *ACM Trans. Graph.* 42, 4, Article 123 (2023), 14 pages.

Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. 2021. Kinetic-Based Multiphase Flow Simulation. *IEEE Transactions on Visualization and Computer Graphics* 27, 7 (2021), 3318–3334.

Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient kinetic simulation of two-phase flows. *ACM Trans. Graph.* 41, 4, Article 114 (2022), 17 pages.

Xingqiao Li, Xingyu Ni, Bo Zhu, Bin Wang, and Baoquan Chen. 2023. GARM-LS: A Gradient-Augmented Reference-Map Method for Level-Set Fluid Simulation. *ACM Trans. Graph.* 42, 6, Article 192 (2023), 20 pages.

Zhiqi Li, Barnabás Börcsök, Duowen Chen, Yutong Sun, Bo Zhu, and Greg Turk. 2024. Lagrangian Covector Fluid with Free Surface. In *ACM SIGGRAPH 2024 (Conference Track)*.

Olivier Mercier, Xi-Yuan Yin, and Jean-Christophe Nave. 2020. The Characteristic Mapping Method for the Linear Advection of Arbitrary Sets. *SIAM Journal on Scientific Computing* 42, 3 (2020), A1663–A1685.

Viorel Mihalef, Betul Unlusu, Dimitris Metaxas, Mark Sussman, and M Yousuff Hussaini. 2006. Physics based boiling simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 317–324.

Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector fluids. *ACM Trans. Graph.* 41, 4, Article 113 (2022), 16 pages.

Fumiya Narita and Ryoichi Ando. 2022. Tiled Characteristic Maps for Tracking Detailed Liquid Surfaces. *Computer Graphics Forum* (2022).

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.

Valery Iustinovich Oseledets. 1989. On a new way of writing the Navier-Stokes equation. The Hamiltonian formalism. *Russ. Math. Surveys* 44 (1989), 210–211.

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

Saket Patkar, Mridul Aanjaneya, Dmitriy Karpman, and Ronald Fedkiw. 2013. A hybrid Lagrangian-Eulerian formulation for bubble generation and dynamics. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. 105–114.

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Trans. Graph.* 38, 4, Article 128 (2019), 12 pages.

PH Roberts. 1972. A Hamiltonian theory for weakly interacting vortices. *Mathematika* 19, 2 (1972), 169–179.

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

Takahiro Sato, Takeo Igarashi, Christopher Batty, and Ryoichi Ando. 2017. A long-term semi-lagrangian method for accurate velocity advection. In *SIGGRAPH Asia 2017 Technical Briefs*. Article 5, 4 pages.

Robert Saye. 2016. Interfacial gauge methods for incompressible fluid dynamics. *Science advances* 2, 6 (2016), e1501869.

Robert Saye. 2017. Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part I. *J. Comput. Phys.* 344 (2017), 647–682.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An Unconditionally Stable MacCormack Method. *J. Sci. Comput.* 35, 2–3 (2008), 350–371.

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

K. Stüben. 2001. A review of algebraic multigrid. *J. Comput. Appl. Math.* 128, 1 (2001), 281–309.

Zhuang Su and Jun Zhang. 2023. Internal reflection of a vortex ring at an air-water interface. *Bulletin of the American Physical Society* (2023).

D.M. Summers. 2000. A Representation of Bounded Viscous Flow Based on Hodge Decomposition of Wall Impulse. *J. Comput. Phys.* 158, 1 (2000), 28–50.

Mark Sussman and Elbridge Gerry Puckett. 2000. A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows. *J. Comput. Phys.* 162, 2 (2000), 301–337.

E Weinan and Jian-Guo Liu. 1997. Finite difference schemes for incompressible flows in the velocity–impulse density formulation. *J. Comput. Phys.* 130, 1 (1997), 67–76.

E Weinan and Jian-Guo Liu. 2003. Gauge method for viscous incompressible flows. *Communications in Mathematical Sciences* 1, 2 (2003), 317–332.

DC Wiggert and EB Wylie. 1976. Numerical predictions of two-dimensional transient groundwater flow by the method of characteristics. *Water Resources Research* 12, 5 (1976), 971–977.

Han Yan and Bo Ren. 2023. High Density Ratio Multi-Fluid Simulation with Peridynamics. *ACM Trans. Graph.* 42, 6, Article 191 (2023), 14 pages.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Trans. Graph.* 37, 4, Article 85 (2018), 8 pages.

Wen Zheng, Jun-Hai Yong, and Jean-Claude Paul. 2006. Simulation of Bubbles. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. The Eurographics Association, 325–333.

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 Transactions on Graphics (SIGGRAPH 2024)* (2024).

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Trans. Graph.* 24, 3 (2005), 965–972.