Lagrangian Covector Fluid with Free Surface

ZHIQI LI, Georgia Institute of Technology, USA
BARNABÁS BÖRCSÖK, Georgia Institute of Technology, USA
DUOWEN CHEN, Georgia Institute of Technology, USA
YUTONG SUN, Georgia Institute of Technology, USA
BO ZHU, Georgia Institute of Technology, USA
GREG TURK, Georgia Institute of Technology, USA

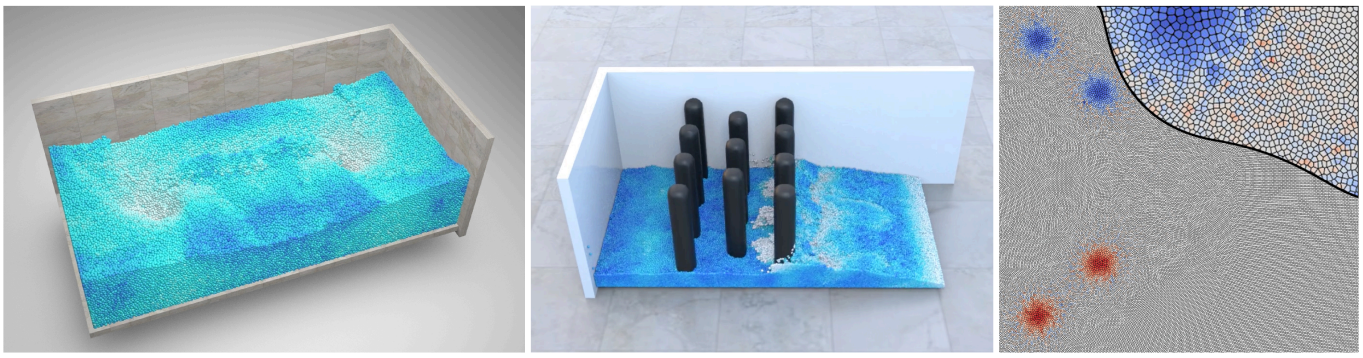


Fig. 1. Examples of our method handling demanding fluid simulation scenarios. In three dimensions, a two-sink setup draining a tank of water *(left)*, and generated waves hitting multiple solid objects *(center)*. Our method achieves state-of-the-art performance amongst purely particle-based methods for simulating dynamic vortical structures, such as two-dimensional leapfrogging *(right)*.

This paper introduces a novel Lagrangian fluid solver based on covector flow maps. We aim to address the challenges of establishing a robust flow-map solver for incompressible fluids under complex boundary conditions. Our key idea is to use particle trajectories to establish precise flow maps and tailor path integrals of physical quantities along these trajectories to reformulate the Poisson problem during the projection step. We devise a decoupling mechanism based on path-integral identities from flow-map theory. This mechanism integrates long-range flow maps for the main fluid body into a short-range projection framework, ensuring a robust treatment of free boundaries. We show that our method can effectively transform a long-range projection problem with integral boundaries into a Poisson problem with standard boundary conditions — specifically, zero Dirichlet on the free surface and zero Neumann on solid boundaries. This transformation significantly enhances robustness and accuracy, extending the applicability of flow-map methods to complex free-surface problems.

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

Additional Key Words and Phrases: Lagrangian method, covector fluid, flow map, path integral, boundary conditions, free surface, Voronoi diagrams

ACM Reference Format:
Zhiqi Li, Barnabás Börcsök, Duowen Chen, Yutong Sun, Bo Zhu, and Greg Turk. 2024. Lagrangian Covector Fluid with Free Surface. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641519.3657514

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07.
https://doi.org/10.1145/3641519.3657514

1 INTRODUCTION

Flow map methods have garnered increasing interest in both computational physics and computer graphics communities in recent years, as evidenced by the emergence of covector/impulse-based [Deng et al. 2023; Nabizadeh et al. 2022] and vorticity-based [Mercier et al. 2020; Yin et al. 2021, 2023] solvers that are known for their exceptional preservation of vortical structures. The key to developing a flow-map method is constructing an efficient and accurate representation that maps any given point from the initial to the current frame (and vice versa if a bi-directional mapping process is necessary). This flow map, or the relationship between the two endpoints of every mapping trajectory sample, was previously realized by advecting the spatial coordinates [Deng et al. 2023; Hachisuka 2005; Sato et al. 2018, 2017; Tessendorf 2015], with improved accuracy later achieved by tracing particles backward over a recorded spatiotemporal velocity field, both of which were implemented in a fixed, Eulerian domain.

One of the main challenges in the impulse/covector fluid models with their flow-map-based implementations is addressing free-surface boundaries. Existing approaches struggle to simulate free-surface fluids due to inherent difficulties with free-surface boundary conditions that are impractical to manage using traditional numerical solvers. In standard free-surface solvers built on velocity space,

fluid incompressibility is enforced by solving a Poisson problem with zero Dirichlet boundary conditions on the free surface (assuming zero air pressure) and appropriate Neumann boundary conditions on solid boundaries. In contrast, the free-surface boundary conditions for a covector flow map model pose greater challenges. This difficulty arises from the complexity of calculating the kinetic energy integral on the free boundary over the entire flow map interval.

In typical liquid simulation scenarios in computer graphics, the fluid surface undergoes extensive geometric and topological transitions over time. Consequently, a fluid particle may appear on the surface at a certain instant and then merge into the fluid body later. This dynamics makes it impractical to track these particles' statuses consistently and determine whether they are on the free surface at any given time over a high-dimensional spatiotemporal space. However, the accuracy of the boundary condition depends on the path integral of *all particles currently on the free surface over the entire flow-map period.* Due to these challenges with traditional numerical solutions, the covector/impulse frameworks and their flow-map implementation are confined to solving fluid without a free surface and can produce smoke animations only.

This paper makes the first step toward addressing the free-surface boundary challenges for covector flow-map methods. We attack the problem from the Lagrangian perspective by treating each flow-map sample as a Lagrangian particle trajectory. Under this Lagrangian view, we proposed a novel mathematical framework based on the path integral identities in flow-map theory to decouple the mapping and projection steps in a conventional covector flow-map algorithm. Our key idea is to leverage these mathematical identities to flexibly control the integral intervals for different physical quantities along each Lagrangian path, and then transform the boundary conditions from the long-range integral (through all flow map time steps) to a short-range integral (within a single time step), and eventually to a standard zero Dirichlet boundary to fit into the existing Poisson solver. By doing so, we can decouple the long-range map for velocity, which serves as the enabling mechanism for the vortical expressiveness of flow-map methods, and the pressure projection step, which is fundamental for simulating incompressible flow, without any model degeneration or artificial numerical blending.

Our proposed Lagrangian covector solver comprises three components: a Lagrangian model for long-range flow maps and path integrals, a reformulated Poisson solver designed for handling complex boundaries, and a Voronoi-based numerical implementation. These components synergistically establish a particle-based framework that facilitates integral-flexible flow maps for the first covector solver capable of handling free surfaces.

We summarize our main contributions as follows:

- We proposed a novel Lagrangian flow map model characterizing the path integral form of covector fluid.
- We proposed a novel mechanism to incorporate long-range flow maps into a short-range projection step and reformulated it into a standard Poisson problem.
- We proposed the first free-surface covector fluid model based on a Voronoi implementation.

2 RELATED WORK

Particle-based fluid simulation. [De Goes et al. 2015] introduced power particles to simulate incompressible fluids. Their geometrically inspired method offers precise pressure solving and an even distribution of particles by describing the fluid motion as a series of well-shaped power diagrams. [Zhai et al. 2018] accelerated the construction of power diagrams on GPUs, adopting ghost particles for fluid-air interactions, while [Lévy 2022] introduced a more precise technique to calculate free-surface interactions, framing it as an optimal transport problem.

Flow-map methods. The method of characteristic mapping (MCM) of Wiggert and Wylie [1976] was first introduced to the graphics community by Tessendorf and Pelfrey [2011]. Due to its superiority in dealing with numerical dissipation through long-range mapping, some methods [Hachisuka 2005; Sato et al. 2018, 2017; Tessendorf 2015] trade off computational cost for better accuracy utilizing virtual particles for tracking the mapping. Subsequently, Nabizadeh et al. [2022] combined this with an impulse fluid model [Cortez 1996] and Qu et al. [2019] developed a bidirectional mapping to prevent dissipation better. Deng et al. [2023] stores intermediate velocity fields using a neural network for storage compression. We track such mapping directly with particles. Compared to previous flow map methods (e.g., [Sato et al. 2017]), our method relies on calculating path integrals on particles directly without any backtrace step, eliminating any extra velocity buffer.

Gauge-based fluid. The concept of the *impulse variable* was initially introduced in [Buttke 1992], reformulating the incompressible Navier-Stokes Equation with a gauge variable and transformation [Buttke 1993; Oseledets 1989; Roberts 1972]. Various gauges have been proposed for applications like boundary treatment, numerical stability, and turbulence simulation [Buttke 1993; Buttke and Chorin 1993; Cortez 1996; Summers 2000; Weinan and Liu 2003]. Saye [2016] and Saye [2017] employed a different gauge for interfacial discontinuity issues. In the graphics community, researchers like Feng et al. [2022], Xiong et al. [2022], Yang et al. [2021], and Nabizadeh et al. [2022] have explored this area, though facing challenges with accurate advection. Flow maps, as shown in [Deng et al. 2023], are effective for advecting such variables, and this method has been adapted to particles in our research.

3 MATHEMATICAL FOUNDATION

Flow Maps. To define the flow map, we start with an initial domain Ω_s at time s and its current domain Ω_r at time r, with $s < r$. A particle at $\mathbf{x}_s \in \Omega_s$ at time s moves to $\mathbf{x}_r \in \Omega_r$ at time r by velocity $\mathbf{u}_t, t \in [s, r]$. The relationship between $\mathbf{x}_s$ and $\mathbf{x}_r$ is defined through two flow map functions, including the forward flow map

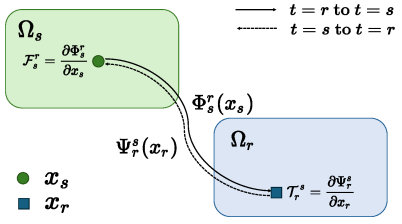


Fig. 2. Forward and backward flow maps defined on a particle.

Notation	Type	Definition
r	scalar	Current time
s	scalar	Initial time
s'	scalar	One time step before the current time
Φ_a^b	vector	Forward flow map from time a to b
Ψ_b^a	vector	Backward flow map from time b to a
$\mathcal{F}_a^b$	matrix	Jacobian of forward flow map Φ_a^b
$\mathcal{T}_b^a$	matrix	Jacobian of backward flow map Ψ_b^a
p_t	scalar	Pressure at time t
λ_t	scalar	Lagrangian pressure calculated at time t
Λ_a^b	scalar	Integration of λ from time a to b
$\mathbf{u}_{a\to b}^M$	vector	**Mapped** velocity at time b by covector flow map from time a
$\mathbf{u}_{a\to b}^A$	vector	**Advected** velocity at time b by particle velocity from time a

Table 1. Summary of important notations used in the paper.

$\mathbf{x}_r = \Phi_s^r(\mathbf{x}_s)$ and the backward flow map $\mathbf{x}_s = \mathbf{\Psi}_r^s(\mathbf{x}_r)$, with Jacobians $\mathcal{F}_s^r = \frac{\partial \Phi_s^r}{\partial \mathbf{x}_s}$ and $\mathcal{T}_r^s = \frac{\partial \Psi_r^s}{\partial \mathbf{x}_r}$. These two flow maps satisfy the equations $\Phi_s^r \circ \Psi_r^s = \mathrm{id}_s$and $\Psi_s^r \circ \Phi_r^s = \mathrm{id}_r$, where id_s and id_r denote the identity transformations in domains Ω_s and Ω_r respectively.

Considering the domain $\Omega_t \in \mathbb{R}^n$ (where $n = 2, 3, \ldots$) at any time t as a linear vector space, the flow maps $\Phi_s^r : \Omega_s \to \Omega_r$ and $\Psi_r^s : \Omega_r \to \Omega_s$ are mappings between these linear spaces. Given a scalar field q_s at time s, a scalar field $\Psi_t^{s*} q_t$ on Ω_t can be induced via the map $(\Psi_t^{s*} q_s)(\mathbf{x}) \triangleq q_s(\Psi_t^s(\mathbf{x}))$ (* means induction, where a map Ψ_t^{s*} between fields is induced by a map Ψ_t^s between domains). This induced scalar field is called the pullback of q_s by the mapping $\Psi_t^s(\mathbf{x})$ (the pullback by $\Phi_s^t(\mathbf{x})$ is similar).

Covector Preliminaries. In the domain Ω_t at any time t, regarded as a linear space, we can define tangent spaces $T_{\mathbf{x}}\Omega_t$ and cotangent spaces $T_{\mathbf{x}}^*\Omega_t$ at any point $\mathbf{x}$ within Ω_t. For $\Omega_t \subset \mathbb{R}^n$, the tangent spaces $T_{\mathbf{x}}\Omega_t = \mathbb{R}^n$ encompass all vectors originating from $\mathbf{x}$, while the cotangent space $T_{\mathbf{x}}^*\Omega_t$ comprises all linear functions (termed cotangent vectors or covectors) defined on $T_{\mathbf{x}}\Omega_t$. Given an inner product in $\Omega_t \subset \mathbb{R}^n$, any tangent vector $\mathbf{v}_{\mathbf{x}} \in T_{\mathbf{x}}\Omega_t$ induces a covector in $T_{\mathbf{x}}^*\Omega_t$, such that $\mathbf{v}_{\mathbf{x}}^\flat(\mathbf{w}) = \langle \mathbf{v}, \mathbf{w} \rangle$ for all $\mathbf{w} \in T_{\mathbf{x}}\Omega_t$. Here, $\flat$ denotes the conversion of a tangent vector to a covector, and $\sharp$ signifies the reverse conversion. A vector field $\mathbf{v}(\mathbf{x})$ (or covector field $\mathbf{v}^\flat(\mathbf{x})$) is formed by selecting one vector $\mathbf{v}_{\mathbf{x}} = \mathbf{v}(\mathbf{x})$ (or one covector $\mathbf{v}_{\mathbf{x}}^\flat = \mathbf{v}^\flat(\mathbf{x})$) from each tangent space $T_{\mathbf{x}}\Omega_t$ (or the cotangent space $T_{\mathbf{x}}^*\Omega_t$) at every point $\mathbf{x}$. The gradient dq of a scalar field q can be considered as a vector field. Refer to [Crane et al. 2013] for details.

Lie Advection. For any covector field $\mathbf{v}_s^\flat$ taken from Ω_s, Ψ_t^s induces a covector field $\Psi_t^{s*}\mathbf{v}_s^\flat$ on Ω_t as $(\Psi_t^{s*}\mathbf{v}_s^\flat)(\mathbf{x}) \triangleq [\mathcal{T}_t^{s\,T}(\mathbf{x})\mathbf{v}_s(\Psi_t^s(\mathbf{x}))]^\flat$, which is referred to as the pullback of the covector field $\mathbf{v}_s^\flat$ by Ψ_t^s. The pullback of $\mathbf{v}_s^\flat$ at any time t satisfies an Lie advection equation:$\left(\frac{\partial}{\partial t} + \mathcal{L}_{\mathbf{u}}\right)\mathbf{v}_t^\flat = 0$, where $\mathcal{L}_{\mathbf{u}}$ is the Lie derivative, and in $\Omega_r \subset \mathbb{R}^n$, it is given by $(\mathcal{L}_{\mathbf{u}}\mathbf{v}_t^\flat)^\sharp = (\mathbf{u}\cdot\nabla)\mathbf{v}_t + (\nabla\mathbf{u})^T \cdot \mathbf{v}_t$. Therefore if there is a vector field that satisfies this advection equation, at any time t, $\mathbf{v}_t^\flat$ can be described through the pullback of the initial $\mathbf{v}_s^\flat$ through pullback.

4 PHYSICAL MODEL

Covector Incompressible Fluid. We model incompressible flow using Euler equations by assuming viscosity zero and density one:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = 0, \\ \nabla \cdot \mathbf{u} = 0, \end{cases} \tag{1}$$

with $\mathbf{u}$ and p specifying the fluid velocity and pressure. The first equation describes the momentum conservation, and the second equation specifies incompressibility. According to the covector model proposed in [Nabizadeh et al. 2022], Equation 1 can be reformulated into its covector form as $\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\mathbf{u}^T \cdot \mathbf{u} + \nabla(p - \frac{1}{2}|\mathbf{u}|^2) = 0$, which can be further written with Lie advection as

$$(\frac{\partial}{\partial t} + \mathcal{L}_{\mathbf{u}})\mathbf{u}^\flat + d(p - \frac{1}{2}|\mathbf{u}|^2) := (\frac{\partial}{\partial t} + \mathcal{L}_{\mathbf{u}})\mathbf{u}^\flat + d\lambda = 0, \tag{2}$$

where $\lambda = p - \frac{1}{2}|\mathbf{u}|^2$ is defined as Lagrangian pressure.

Covector Fluid on Flow Maps. The solution $\mathbf{u}_r$ of Equation 2 at r includes terms obtained by pulling back the velocity field as a covector through the flow map. This process is illustrated by integrating both sides of Equation 2 from time s to r, represented as a covector:

$$\begin{aligned} \mathbf{u}_r^\flat &= \Psi_r^{s*}\mathbf{u}_s^\flat - \int_s^r (\Phi_s^\tau \circ \Psi_r^s)^* d\lambda_\tau d\tau, \\ &= \Psi_r^{s*}\mathbf{u}_s^\flat - d\int_s^r (\Phi_s^\tau \circ \Psi_r^s)^* \lambda_\tau d\tau, \end{aligned} \tag{3}$$

with the second line arising from the commutativity between the differential operator ∇ (more accurately denoted as "d") and the pullback operator (Equation 7 in [Nabizadeh et al. 2022]), and commutativity between ∇ and the integral sign.

Next, we can convert the covector forms and their pullbacks in Equation 3 into their vector forms as:

$$\mathbf{u}(\mathbf{x}, r) = \underbrace{\mathcal{T}_r^{s\,T}\mathbf{u}_s(\Psi_r^s(\mathbf{x}), s)}_{\text{mapping}} - \underbrace{\nabla\int_s^r \lambda((\Phi_s^\tau \circ \Psi_r^s)(\mathbf{x}), \tau)d\tau}_{\text{projection}}, \tag{4}$$

which constitutes two main steps to obtain velocity at $(\mathbf{x}, r)$ using a *mapping-projection* scheme. Initially, the mapping step calculates $\mathcal{T}_r^{s\,T}\mathbf{u}_s(\Psi_r^s(\mathbf{x}), s)$ as the pullback of the velocity field $\mathbf{u}_s$ by the flow map. In the projection step, the gradient of $\int_s^t \lambda((\Phi_s^\tau \circ \Psi_r^s)(\mathbf{x}), \tau)d\tau$ is projected from the mapped velocity to enforce incompressibility.

5 PATH INTEGRAL ON LAGRANGIAN FLOW MAPS

Equation 4 can be written into a path integral form on a Lagrangian trajectory. For a fluid particle q with position $\mathbf{x}_q(t)$ at any time t, the projection term in Equation 4 is the path integral of λ along its trajectory from time s to r, which is denoted as

$$\Lambda_{s,q}^r \triangleq \int_s^r \lambda((\Phi_s^\tau \circ \Psi_r^s)(\mathbf{x}_q(r)), \tau)d\tau. \tag{5}$$

Here the subscript q indicates that this quantity is carried by the particle q. For the mapping part, $\mathbf{u}_s(\Psi_r^s(\mathbf{x}_\mathbf{q}(\mathbf{r})), s)$ is the velocity of particle q at time s, and we simplify the notation as $\mathbf{u}_{s,q}$. Because

$\mathcal{T}_r^s = \frac{\partial \mathbf{x}_s}{\partial \mathbf{x}_r}$ is determined by the positions of all particles in the flow field at time r and s, it can be carried on fluid particles, denoted as $\mathcal{T}_{r,q}^s$. We obtain the reformulated Equation 4 as a path integral:

$$\boxed{\mathbf{u}_{r,q} = \underbrace{\mathcal{T}_{r,q}^{s\ T}\mathbf{u}_{s,q}}_{\text{mapping}} - \underbrace{\nabla\Lambda_{s,q}^r}_{\text{projection}}}. \tag{6}$$

Equation 6 forms the mathematical foundation of our Lagrangian covector fluid model. This equation, in contrast to Equation 4, considers the mapping and projection processes occurring on a particle as it moves along its trajectory in the flow field, which simplifies the formulation by eliminating the need for back-and-forth mappings to identify a Lagrangian point. Thus, we reinterpret the mapping-projection process on a moving particle q as follows: (1) calculate the mapped velocity from the initial time s as $\boldsymbol{u}_{s\to r,q}^M = \mathcal{T}_{r,q}^{s\ T}\mathbf{u}_{s,q}$, and (2) remove its irrotational component by subtracting the gradient of the path integral of the Lagrangian pressure $\nabla\Lambda_{s,q}^r$.

6 INCOMPRESSIBILITY

6.1 Long-Range Mapping, Long-Range Projection

To determine the velocity of a particle q at time r, we initially consider a straightforward method for solving Equation 6. We present the method as a standard *mapping-projection* scheme:

(1) (**Long-Range Mapping**) Calculate the long-range mapped velocity as: $\mathbf{u}_{s\to r,q}^M = \mathcal{T}_{r,q}^{s\ T}\mathbf{u}_{s,q}$;
(2) (**Long-Range Projection**) Solving the following Poisson equation to obtain $\Lambda_{s,q}^r$:

$$\begin{cases} \nabla\cdot\nabla\Lambda_s^r = \nabla\cdot\mathbf{u}_{s\to r}^M, & \mathbf{x}\in\Omega \\ \mathbf{u}_r = \mathbf{u}_b, & \mathbf{x}\in\partial_s\Omega \\ p = 0, & \mathbf{x}\in\partial_f\Omega \end{cases} \tag{7}$$

where $\mathbf{u}_b$ denotes the velocity of the solid boundary, and $\partial_s\Omega$ and $\partial_f\Omega$ denote solid boundary and free surface boundary respectively. Then, we carry out a projection as $\mathbf{u}_{r,q} = \mathbf{u}_{s\to r,q}^M - \nabla\Lambda_{s,q}^r$ to ensure $\mathbf{u}_r$ satisfies $\nabla\cdot\mathbf{u}_r = 0$.

Here, $\mathbf{u}_{s\to r,q}^M$ is long-range mapped from the initial time s and is projected to $\mathbf{u}_r$ by a gradient of long-range path integral of pressure $\Lambda_{s,q}^r$ from time s to t. Hence, the Poisson equation we solve in Equation 7 leads to a long-range projection of the rotational component accumulated from time s to time r. Given these, we refer to this strategy as **L**ong-Range **M**apping **L**ong-Range **P**rojection (**LMLP**).

LMLP has two main issues regarding boundary and performance:

(1) Setting boundary conditions is challenging. Specifically, at the free boundary $\partial_f\Omega$, to enforce the boundary condition $p = 0$, we define $\Lambda_{s,q}^r = \int_s^r p_{\tau,q} - \frac{1}{2}|\mathbf{u}_{\tau,q}|^2 d\tau$. This implies the necessity of establishing a non-zero Dirichlet boundary condition such that $\Lambda_{s,q}^r = \int_s^r -\frac{1}{2}|\mathbf{u}_{\tau,q}|^2 d\tau$. This condition requires a comprehensive path integral of the kinematic energy across a particle's trajectory. Although seemingly feasible from a Lagrangian perspective, it is crucial to recognize that this path integral only constitutes a valid Dirichlet boundary under the specific condition that *the particle remains on the free surface from time s to t*. However, in practical scenarios, this condition is rarely met due to the dynamic topological changes of the surface over time. In the interval from time s to r, particles may transition between the interior and the surface, rendering the calculation of this integral impractical.
(2) The performance issue is notable. The term $\mathbf{u}_{s\to r,q}^M$ encompasses a substantial divergent component $\nabla\Lambda_{s,q}^r$, which includes an integral extending from time s to t. Attempting to eliminate this component through a single projection results in significant computational expenses, primarily due to the increased number of iterations required for solving the Poisson equation.

6.2 Short-Range Mapping, Short-Range Projection

To address the two issues mentioned above, we devise a short-range approach. By setting the flow map's start point to be *only one timestep before r, denoted as s′*, we can calculate the velocity at time r with the following two steps:

(1) (**Short-Range Mapping**) Calculate the mapped velocity: $\mathbf{u}_{s'\to r,q}^M = \mathcal{T}_{r,q}^{s'\ T}\mathbf{u}_{s',q}$
(2) (**Short-Range Projection**) Solve the following Poisson equation to obtain $\hat{\lambda}_q$:

$$\begin{cases} \nabla\cdot\nabla\hat{\lambda} = \nabla\cdot\mathbf{u}_{s'\to r}^M, & \mathbf{x}\in\Omega \\ \mathbf{u}_r = \mathbf{u}_b, & \mathbf{x}\in\partial_s\Omega \\ p = 0 \Rightarrow \hat{\lambda} = -\frac{1}{2}|\mathbf{u}_r^2|\Delta t, & \mathbf{x}\in\partial_f\Omega \end{cases} \tag{8}$$

where $\hat{\lambda}_q = \lambda_{r,q}\Delta t$ is the numerical calculation of one step integration $\Lambda_{s',q}^r$. Then, we conduct the divergence-free projection as $\mathbf{u}_{r,q} = \mathbf{u}_{s'\to r,q}^M - \nabla\hat{\lambda}_q$.

Given both mapping and projection are limited within a short interval from time s' to r, we refer the scheme as **S**hort-Range **M**apping **S**hort-Range **P**rojection (**SMSP**). It is worth noting that SMSP is akin to the Semi-Lagrangian advection schme, as they both involve computing a one-step mapping. The difference lies in the fact that SMSP employs a mapping form based on the Lie advection equation $\mathbf{u}_{s'\to r,q}^M = \mathcal{T}_{r,q}^{s'\ T}\mathbf{u}_{s',q}$, whereas Semi-Lagrangian employs a mapping form based on the ordinary advection equation $\mathbf{u}_{s'\to r,q}^A = \mathbf{u}_{s',q}$.

SMSP ensures robustness by addressing the two issues above: (1) The Neumann boundary can be set as $\Lambda_{s',q}^r = \int_{s'}^r -\frac{1}{2}|\mathbf{u}_{\tau,q}|^2 d\tau$, namely, $\hat{\lambda}_q = -\frac{1}{2}|\mathbf{u}_{s',q}|^2\Delta t$, which is robustly calculable (because of only one timestep). (2) The Poisson solve converges fast thanks to the small divergence on its right-hand side. However, SMSP forgoes the advantages associated with employing a long-range flow map for vorticity conservation. Instead, it resolves to a reformulated Euler equation in covector form, accompanied by a one-step particle advection process. This approach results in the loss of the benefits inherent in utilizing a flow map to preserve vorticies.

6.3 Long-Range Mapping, Short-Range Projection

A natural next step is to combine the merits of LMLP and SMSP. However, this is not mathematically intuitive. As shown in Equation 6, the mapping and projection steps require the same time interval for their path integrals.

To address this issue, we observe the following identity:

$$\boxed{\underbrace{\mathbf{u}^M_{s'\to r,q}}_{\text{SM}} = \underbrace{\mathbf{u}^M_{s\to r,q}}_{\text{LM}} - \nabla\Lambda^{s'}_{s,q}}. \tag{9}$$

This identity can be simply proved as $\mathbf{u}_{r,q} = \mathbf{u}^M_{s\to t,q} - \nabla\Lambda^r_{s,q} = \mathbf{u}^M_{s'\to t,q} - \nabla\Lambda^r_{s',q}$ and $\Lambda^r_{s,q} = \Lambda^{s'}_{s,q} + \Lambda^r_{s',q}$. It establishes a connection between the long-range and short-range mappings, allowing us to express a short-range mapping by adding the gradient of pressure integral $\nabla\Lambda^{s'}_{s,q}$ to its long-range counterpart. This calculation is numerically robust, because for each particle, $\Lambda^{s'}_{s,q}$ is the path integral of the Lagrangian pressure on its trajectory. Substituting Equation 9 into Equation 8, we obtain the following scheme:

(1) (**Long-Range Mapping**) Calculate mapped velocity by the long-range flow map: $\mathbf{u}^M_{s\to r,q} = {\mathcal{T}^s_{r,q}}^T \mathbf{u}_{s,q}$;
(2) Calculate mapped velocity by short-range flow map based on the long-range mapped velocity as: $\mathbf{u}^M_{s'\to r,q} = \mathbf{u}^M_{s\to r,q} - \nabla\Lambda^{s'}_{s,q}$
(3) (**Short-Range Projection**) Solve the following Poisson equation to obtain $\hat{\lambda}_q$:

$$\begin{cases} \nabla\cdot\nabla\hat{\lambda} = \nabla\cdot\mathbf{u}^M_{s'\to r}, & \mathbf{x}\in\Omega \\ \mathbf{u}_r = \mathbf{u}_b, & \mathbf{x}\in\partial_s\Omega \\ p = 0 \Rightarrow \hat{\lambda} = -\frac{1}{2}|\mathbf{u}^2_r|\Delta t, & \mathbf{x}\in\partial_f\Omega \end{cases} \tag{10}$$

where $\hat{\lambda}_q = \lambda_{r,q}\Delta t$ is the numerical calculation of one step integration $\Lambda^r_{s',q}$.
(4) Update the integral: $\Lambda^r_{s,q} = \Lambda^{s'}_{s,q} + \hat{\lambda}_q$ and project $\mathbf{u}_{r,q} = \mathbf{u}^M_{s\to r,q} - \nabla\Lambda^r_{s,q}$ to ensure $\mathbf{u}_r$ satisfies $\nabla\cdot\mathbf{u}_r = 0$.

We name it as **L**ong-Range **M**apping **S**hort-Range **P**rojection(**LMSP**). In LMSP, the $\Lambda^{s'}_s$ in Step (2) is calculated in the previous time step, and this value is updated in Step (4) by $\Lambda^r_{s,q} = \Lambda^{s'}_{s,q} + \hat{\lambda}_q$ to the value in the current time step. This accumulation might lead to a concern that accumulating pressure Λ^r_s could potentially lead to the accumulation of numerical errors, thereby diminishing the long-range map's ability to preserve vorticity when calculating $\mathbf{u}^M_{s\to r,q} - \nabla\Lambda^{s'}_s$ in Step (2). We show this is not an issue. Because $\nabla\Lambda^{s'}_s$ is a gradient field, it only affects the divergence component of $\mathbf{u}^M_{s\to r,q}$. Any numerical error accumulated in $\Lambda^{s'}_s$ goes directly into the rotational part $\mathbf{u}^M_{s\to r,q}$, which will be removed after projection.

We further demonstrate this with the following proposition:

Proposition 6.1. *Poission Equation 10 with initial guess $\hat{\lambda} = 0$ is equivalent to Poission Equation 7 with initial guess $\Lambda^r_s = \Lambda^{s'}_s$*

Proof: Substitute Equation 9 into Equation 10, we obtain $\nabla\cdot\nabla\hat{\lambda} = \nabla\cdot[\nabla\mathbf{u}^M_{s\to r,q} - \nabla\Lambda^{s'}_{s,q}]$, which is equivalent to $\nabla\cdot\nabla(\hat{\lambda} + \Lambda^{s'}_{s,q}) = \nabla\cdot\nabla\mathbf{u}^M_{s\to r,q}$. Use Λ^r_s to substitute $\hat{\lambda} + \Lambda^{s'}_{s,q}$, and we get Equation 7. Also, for the initial guess, when $\hat{\lambda} = 0$, $\Lambda = \Lambda^{s'}_{s,q}$.

7 ADAPTING TO CLASSICAL ADVECTION-PROJECTION

In the final movement, we further adapt the LMSP scheme to a classical advection-projection scheme by solving the Poisson equation with standard Neumann and Dirichlet boundary conditions, namely $p = 0$ on $\partial_f\Omega$ and $\boldsymbol{u} = \boldsymbol{u}_b$ on $\partial_s\Omega$. This modification is motivated by the desire to circumvent any inaccuracies arising from the approximation of $\mathcal{T}^r_s$ for a particle on the surface (due to the lack of sufficient neighboring particles to approximate the Jacobian), which could potentially lead to numerical instabilities during the simulation. We showed an ablation test for this issue in Fig. 11.

We observe the following identity regarding a particle's velocity on a one-step flow map:

$$\boxed{\mathbf{u}^A_{s'\to r,q} = \mathbf{u}^M_{s'\to r,q} + \Delta t\nabla(\frac{1}{2}|\mathbf{u}_{s',q}|^2)} \tag{11}$$

where $\mathbf{u}^A_{s'\to r,q}$ represents the passively advected velocity on the particle's trajectory, namely, to move the particle to a new position and keep its velocity as it is, as seen in all traditional particle-based advection schemes. We show a brief proof in the Appendix A.

Combining Equation 9 and Equation 11, we obtain our final advection-projection scheme as:

(1) (**Long-Range Mapping**) For the interior particles, calculate the long-flow-map mapped velocity: $\mathbf{u}^M_{s\to r,q} = {\mathcal{T}^s_{r,q}}^T \mathbf{u}_{s,q}$.
(2) For an interior particle, calculate its velocity as the advected velocity expressed with long-flow-map mapped velocity (Eq. 11): $\mathbf{u}^A_{s'\to r,q} = \mathbf{u}^M_{s\to r} - \nabla\Lambda^{s'}_s + \nabla(\frac{1}{2}|\mathbf{u}_{s',q}|^2)\Delta t$.
(3) For a boundary particle (refer to Figure 4), calculate its velocity as the advected velocity: $\mathbf{u}^A_{s'\to r,q} = \mathbf{u}_{s',q}$.
(4) (**Classical Projection**) Solve the classical Poisson equation to obtain p_q:

$$\begin{cases} \nabla\cdot\nabla p = \nabla\cdot\mathbf{u}^A_{s'\to r}, & \mathbf{x}\in\Omega \\ \mathbf{u}_r = \mathbf{u}_b, & \mathbf{x}\in\partial_s\Omega \\ p = 0. & \mathbf{x}\in\partial_f\Omega \end{cases} \tag{12}$$

(5) Update the integral: $\Lambda^r_{s,q} = \Lambda^{s'}_{s,q} + \Delta t(p_q - \frac{1}{2}|\mathbf{u}_{s',q}|^2)$ and do projection $\mathbf{u}_{r,q} = \mathbf{u}^M_{s\to r,q} - \nabla\Lambda^r_{s,q}$ for interior part, and $\mathbf{u}_{r,q} = \mathbf{u}^A_{s'\to r,q} - \nabla p_q\Delta t$ for the part near free surface.

We name it as **L**ong-Range **M**apping **Classical P**rojection (**LMCP**). In this scheme, the calculation of the Poisson equation is performed for $u^A_{s'\to t}$ throughout the entire domain, ensuring the mathematical consistency of the final velocity by the covector flow map $u^M_{s\to r,q}$ and the advected velocity $u^A_{s\to r,q}$ after projection. At the same time, for interior particles, the advected velocity $u^A_{s'\to t}$ is calculated by $\mathbf{u}^A_{s'\to r,q} = \mathbf{u}^M_{s\to r,q} - \nabla\Lambda^{s'}_{s,q} + \nabla(\frac{1}{2}|\mathbf{u}_{s',q}|^2)\Delta t$ with long-range mapping for vorticity preserving. Again, because only the gradient $-\nabla\Lambda^{s'}_s + \nabla(\frac{1}{2}|\mathbf{u}_{s'}|^2)\Delta t$ is added to $\mathbf{u}^M_{s\to r}$, similar to subsection 6.3, the process of adding $-\nabla\Lambda^{s'}_s + \nabla(\frac{1}{2}|\mathbf{u}_{s'}|^2)\Delta t$ to $\mathbf{u}^M_{s\to r}$ to obtain $\mathbf{u}^A_{s'\to r}$ keeps the long-range flow map's ability to preserve vorticity.

Summary. We establish a long-range, Lagrangian flow map based on Equation 6 and incorporate it in a classical projection step with zero Neumann boundary by leveraging Equation 9 and 11.

8 TIME INTEGRATION

We adopt the LMCP scheme in our simulation and summarize the time integration scheme of our approach in Algorithm 1. We provide further implementation details in the Appendix B.

Algorithm 1 Lagrangian Covector Fluid

Input: Initial velocity $\mathbf{u}_s$; Initial particle positions $\mathbf{x}_s$; Re-initialization decision strategy $\mathcal{R}$; Near-surface judgment $\mathcal{J}$

set time $s \leftarrow 0, s' \leftarrow 0$; integration $\Lambda^s_{s,i} \leftarrow 0$
for each time step and at time r **do**
  if re-initialization decision strategy $\mathcal{R}$ satisfy **then**
    set $s \leftarrow r, s' \leftarrow r$
    set $\mathbf{u}_{s,i} \leftarrow \mathbf{u}_i$; $\mathbf{x}_{s,i} \leftarrow \mathbf{x}_i$; $\Lambda^s_{s,i} \leftarrow 0$
  for all particle i **do**
    advect particle position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i \Delta t$
  for all particle i **do**
    if $\mathcal{J}(i)$ is True **then**
      compute $\mathbf{u}^A_{s'\to r,i} \leftarrow u_{s',i}$
    else
      compute $\mathcal{T}^s_{r,i} \leftarrow \frac{\partial \mathbf{x}_s}{\partial \mathbf{x}_r}|_{\mathbf{x}_r=\mathbf{x}_i}$
      compute $\mathbf{u}^M_{s\to r,i} \leftarrow \mathcal{T}^s_{r,i}\mathbf{u}_{s,i}$
      compute $\mathbf{u}^A_{s'\to r,i} \leftarrow \mathbf{u}^M_{s\to r,i} - \nabla\Lambda^{s'}_{s,i} + \nabla(\frac{1}{2}|\mathbf{u}_{s',i}|^2)\Delta t$
  solve possion Equation 12 and get $p_{r,i}$
  update $\Lambda^r_{s,i} \leftarrow \Lambda^{s'}_{s,i} + (p_{r,i} - \frac{1}{2}|\mathbf{u}_{r,i}|^2)\Delta t$
  for all particle i **do**
    do velocity projection: $\mathbf{u}_{r,i} = \mathbf{u}^A_{s'\to r,i} - \nabla p_{r,i}$
  set last time: $s' \leftarrow r$

9 NUMERICAL IMPLEMENTATION

Voronoi-based Discretization. We implemented a Voronoi-based particle method for numerical implementation. In each time step, we generate Voronoi diagrams for all moving particles, with each particle corresponding to a Voronoi cell. We represent the i-th particle as q_i, with position $\mathbf{x}_i$. As shown in Fig. 3, the cell corresponding to q_i is denoted $\mathcal{V}_i$, with a volume V_i and centroid $\mathbf{b}_i$. The adjacent facet between two neighboring cells $\mathcal{V}_i$ and $\mathcal{V}_j$ is denoted as $\mathcal{A}_{ij}$. The facet $\mathcal{A}_{ij}$ has the area A_{ij} and the centroid $\mathbf{b}_{ij}$. The distances from $\mathbf{x}_i$ and $\mathbf{x}_j$ to the facet $\mathcal{A}_{ij}$ are respectively denoted as d_{ij} and d_{ji}, while the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is denoted as l_{ij}. According to the properties of Voronoi diagrams, $\mathcal{A}_{ij}$ bisects the line connecting $\mathbf{x}_i$ and $\mathbf{x}_j$ perpendicularly, so $d_{ij} = d_{ji}$ and $d_{ij} + d_{ji} = l_{ij}$ holds.

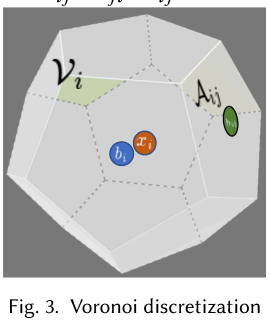


Fig. 3. Voronoi discretization

For each particle q_i, its associated Voronoi cell $\mathcal{V}_i$ is used to define a matrix-form discrete gradient operator G and divergence operator D to calculate the gradient Gp of a scalar quantity q and the divergence $D\mathbf{v}$ of a vector quantity $\mathbf{v}$ carried by particles. The computation of these operators relies on the calculation of the rate of change of the volume $\nabla_{\mathbf{x}_i} V_j$ induced by particle positions. We use the formula $\nabla_{\mathbf{x}_j} V_i = \frac{A_{ij}}{l_{ij}}(\mathbf{x}_j - \mathbf{b}_{ij})$ and $\nabla_{\mathbf{x}_i} V_i = -\sum_{j\in\mathcal{N}_i} \nabla_{\mathbf{x}_i} V_j$ for calculating $\nabla_{\mathbf{x}_i} V_j$ as given in [De Goes et al. 2015].

According to [De Goes et al. 2015; Duque 2023], the matrix-form divergence operator and gradient operator can be defined directly from $\nabla_{\mathbf{x}_i} V_j$ as $D_{ij} = (\nabla_{\mathbf{x}_j} V_i)^T$, $D_{ii} = (\nabla_{\mathbf{x}_i} V_i)^T$ and $G = -D^T$. The divergence of $\mathbf{v}$ and the gradient of p can be computed using the matrix-form divergence and gradient operators defined as follows:

$$[D\mathbf{v}]_i = \sum_{j\in N_i} \frac{A_{ij}}{l_{ij}}[(\mathbf{b}_{ij} - \mathbf{x}_i)\cdot\mathbf{v}_i + (\mathbf{x}_j - \mathbf{b}_{ij})\cdot\mathbf{v}_j],$$
$$[Gp]_i = \sum_{j\in N_i} \frac{A_{ij}}{l_{ij}}(\mathbf{x}_i - \mathbf{b}_{ij})(p_i - p_j), \tag{13}$$

where N_i represents the set composed of the cells adjacent to cell i.

When solving the Poisson equation for velocity projection, the Laplacian operator L is defined as $L = DV^{-1}G$, where V is a diagonal matrix composed of all cell volumes V_i. Because the differential operator matrices satisfy $G = -D^T$, the operator $-L$ is symmetric and positive semi-definite, allowing the discretized Poisson equation $-Lp = -D\mathbf{u}^*$ to be solved using the Conjugate Gradient method, where $\mathbf{u}^*$ represents the velocity before the divergence projection.

Boundary. Similar to [De Goes et al. 2015], we represent the solid boundary using solid particles and sample air particles near the free surface, utilizing the ghost particle method from [Schechter and Bridson 2012] (see Fig. 4). Solid particles and air particles are used for clipping the Voronoi diagrams calculated for fluid particles and setting boundary conditions, as proposed in [De Goes et al. 2015].

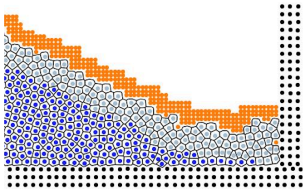


Fig. 4. Particles represent either interior (blue) or boundary (gray-blue) fluid, boundaries (black), or air (orange).

Others. For gravity, we accumulate the integration of gravity $G^r_s = \int_s^r g d\tau$ along the particle trajectory and add it to the mapped velocity before projection:$\mathbf{u}^M_{s\to r} \leftarrow \mathbf{u}^M_{s\to r} + G^r_s$ for interior part and add gravity directly to advected velocity $\mathbf{u}^A_{s'\to r} \leftarrow \mathbf{u}^A_{s'\to r} + g\Delta t$ near the free surface. To make particle distribution more uniform, like in [De Goes et al. 2015], at the end of each time step, we put particles to the centroid of their corresponding Voronoi cell $\mathbf{x}_i \leftarrow \mathbf{b}_i$.

10 RESULTS AND DISCUSSION

Validation. We demonstrate the effectiveness of our method by performing four benchmark experiments against the power particle method (PPM) [De Goes et al. 2015][1], which shares implementation-wise similarities given its geometric data structure. We observe slower energy dissipation rates, less vorticity noise, and better preservation of vortical structures. We color each 2D particle blue (lower/negative values) through gray to red (higher/positive values), based on a linear curve corresponding to its vorticity magnitude.

(1) Leapfrog. We initialize two negative and two positive vortex rings, letting the four vortices move forward by the velocity field generated by their influence on each other. The rate of energy dissipation directly relates to the number of cycles they move forward

[1]For simplicity, we implement a Voronoi diagram instead of the power diagram without affecting the ability to preserve vortex for both our methods and PPM.

before merging. As illustrated in Fig. 6, our method shows a better preservation of the vortical structures than PPM; vortices simulated with our method stay separated at the point where the vortices in the PPM simulation already merged. **(2) Taylor Vortices.** We benchmark our method by simulating two Taylor vortices placed 0.815 apart from each other (Fig. 16). Their velocity field is given by $\omega(\mathbf{x}) = U/a(2-r^2/a^2)exp((1-r^2/a^2)/2)$, where we use $U = 1$ and $a = 0.3$, and r denotes the distance from $\mathbf{x}$ to the vortex center.Using our method, we observe the vortices staying separate, whereas using PPM, they merge at the center. **(3) Taylor-Green Vortices.** Fig. 10 shows a simulation started with a symmetric divergence-free velocity field. The fluid is expected to maintain symmetry along the two axes in 2D while rotating. We observe our method producing less noise in terms of vorticity magnitude carried by the particles as compared to PPM (Fig. 10a-b), with our method presenting a far better energy dissipation curve (Fig. 10c). **(4) 3D Dam Break.** In Fig. 8, we validate our method with the classical 3D benchmark case for verification of solid boundary and free surface flow handling.

Ablation Study. We illustrate (1) the robustness of our LMCP scheme in handling free surface boundary and (2) the subtraction of accumulated pressure gradient significantly accelerating the convergence rate. Fig. 11 illustrates a robust interface achieved using our scheme, in contrast to instabilities that occur without it. Fig. 7 shows that the subtraction of the accumulated pressure gradient can accelerate the convergence of the Poisson equation solver.

Examples. We show additional 2D and 3D examples to demonstrate the robustness and correctness of our method. We use Taichi [Hu et al. 2019] for our implementation, and experiments are run on Tesla V100 GPUs. We use at most 200 000 particles in all experiments to represent fluid, air, and solids. Voronoi diagrams are created using Scipy [Virtanen et al. 2020] and Qhull [Barber et al. 1996]. **Kármán Vortex Street.** Fig. 15 shows alternating vortices forming downstream from a blunt object caused by the unsteady separation of the fluid. **2D Moving and Rotating Board** Fig. 5 shows an object exhibiting flapping motion by traversing the rectangular domain from left to right, and back while generating vortices in its wake. **3D Single & Double Sink** In these examples, we illustrate our method's ability for accurate vorticity perservation combining with free-surface treatment. For single vortex example, we place an initial vorticity field at the center of the tank. A hole is opened as the sink for the tank and water drains out through the hole. Similar settings are adopted for two sink but the sinks have opposite direction of rotation in order to create interesting surface motion. We can observe spiral patterns on the surface in both examples. Results are shown in Fig. 12, Fig. 17 and Fig. 13 **3D Rotating Board** As illustrated in Fig. 9, a board is placed at the center of the scene and set to rotate at a constant speed. We show our method can handle drastic free-surface change and we deal moving solid boundaries in a robust and effective way. Splashes and detailed water surfaces can be observed. **3D Wave Generator.** In this example, we demonstrate the scenario of waves crashing against several pillars. We observe the dynamic water flow behind the pillars and the interaction between the waves. We show particle view for this example in Fig. 14.

CFL. In all of our examples, we set the CFL number to 1 due to constraints imposed by the Voronoi particles. Larger CFL numbers could lead to drastic changes in the neighbors of particles, potentially causing stability issues. Grid-based methods [Nabizadeh et al. 2022] do not encounter these issues.

Table 2. Performance timing. We measure the time each substep takes, and also how much of this was taken up by constructing the Voronoi diagram. Although our simulation method runs on the GPU, the Voronoi diagram calculation takes place on the CPU.

Average Time Cost per Substep		
Scene name	**Particles**	**Total (Voronoi)**
2D Leapfrog	480k	12.5s (8.8s)
2D Taylor Vortices	360k	9.8s (6.4s)
2D Taylor-Green Vortices	156k	5.96s (2.7s)
3D Dam Break	211k	20.9s (18.12s)
2D Kármán Vortex Street	381k	9.56s (6.74s)
2D Moving and Rotating Board	150k	5.92s (2.66s)
3D Sink	180k	29s (25.7s)
3D Rotating Board	150k	5.9s (2.6s)
3D Wave Generator	403k	35s (32.84s)

11 LIMITATIONS AND FUTURE WORK

In summary, this paper presents a novel Lagrangian approach to establishing covector flow maps under complex boundary conditions. The developed decoupling mechanism, rooted in flow-map theory, effectively combines long-range flow maps with short-range (and classical) projections, ensuring robust handling of free boundaries.

A significant limitation of our approach lies in its exclusive treatment of inviscid flows. Addressing viscous flows, as well as other interfacial phenomena, represents a promising direction for future research. Currently, the speed of our fluid simulation code is constrained by the single-threaded Qhull algorithm used for generating Voronoi cells in each frame. We plan to investigate more efficient schemes for solving incompressibility on particles. In our future work, we aim to delve further into flow-map theories within a weakly compressible framework, enhancing meshfree Lagrangian methods such as SPH. Additionally, we are interested in applying our decoupled mapping-projection scheme to other free-surface problems, including levelset-based and particle-grid methods.

ACKNOWLEDGMENTS

We acknowledge NSF IIS #2313075, ECCS #2318814, CAREER #2420319, IIS #2106733, OISE #2153560, and CNS #1919647 for funding support. We credit the Houdini education license for the production of the video animations.

A PROOF OF EQ. 11

Proof: Consider one step advection from time s' to r and we have $\mathbf{x}_{r,q} = \mathbf{x}_{s',q} + \Delta t \mathbf{u}_{s',q}$. Thus the Jacobian $\mathcal{T}_r^{s'} = \frac{\partial \mathbf{x}_{s',q}}{\partial \mathbf{x}_{r,q}}$ can be calculated as $\mathcal{T}_r^{s'} = \frac{\partial \mathbf{x}_{s',q}}{\partial \mathbf{x}_{r,q}} = \frac{\partial (\mathbf{x}_{r,q} - \Delta t \mathbf{u}_{s',q})}{\partial \mathbf{x}_{r,q}} = I - \Delta t \nabla \mathbf{u}_{s',q}$, where I denotes identity matrix. Thus $\mathbf{u}^M_{s' \to r,q} = \mathcal{T}_{s',q}^{s'\ T} \mathbf{u}_{s',q} = \mathbf{u}_{s',q} -$

$\Delta t \nabla \mathbf{u}_{s',q}^T \cdot \mathbf{u}_{s',q}$. In the particle method, $\mathbf{u}_{s'\to r,q}^A = \mathbf{u}_{s',q}$. Due to $\nabla \mathbf{u}^T \cdot \mathbf{u} = \nabla(\frac{1}{2}|\mathbf{u}|^2)$, we have $\mathbf{u}_{s'\to r,q}^A = \mathbf{u}_{s'\to r,q}^M + \nabla(\frac{1}{2}|\mathbf{u}_{s',\mathbf{q}}|^2)\Delta t$. □

B IMPLEMENTATION DETAILS OF ALGO. 1

Reinitialization. We employ a simple reinitialization decision strategy $\mathcal{R}$ triggered every n substeps ($n = 20$ in our implementation).

Boundary particle checking. We employ the following strategy to obtain the boundary particle set $\mathcal{J}$: At the initial time s, set the flag $\mathcal{J}i$ to False; at each step, check if the particle is in the k layers of particles near the free surface at current time r. If it is, we update the flag $\mathcal{J}i$ to True; $J(i)$ returns the value of $\mathcal{J}i$.($k = 3$ in our implementation).

REFERENCES

C.B. Barber, D.P. Dobkin, and H.T. Huhdanpaa. 1996. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)* 22, 4 (Dec 1996), 469–483. http://www.qhull.org

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.

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

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses* (Anaheim, California) *(SIGGRAPH '13)*. Association for Computing Machinery, New York, NY, USA, Article 7, 126 pages. https://doi.org/10.1145/2504435.2504442

Fernando De Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power particles: an incompressible fluid solver based on power diagrams. *ACM Trans. Graph.* 34, 4 (2015), 50–1.

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.

Daniel Duque. 2023. A unified derivation of Voronoi, power, and finite-element Lagrangian computational fluid dynamics. *European Journal of Mechanics-B/Fluids* 98 (2023), 268–278.

Fan Feng, Jinyuan Liu, Shiying Xiong, Shuqi Yang, Yaorui Zhang, and Bo Zhu. 2022. Impulse fluid simulation. *IEEE Transactions on Visualization and Computer Graphics* (2022).

Toshiya Hachisuka. 2005. Combined Lagrangian-Eulerian approach for accurate advection. In *ACM SIGGRAPH 2005 Posters*. 114–es.

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 Transactions on Graphics (TOG)* 38, 6 (2019), 201.

Bruno Lévy. 2022. Partial optimal transport for a constant-volume Lagrangian mesh with free boundaries. *J. Comput. Phys.* 451 (2022), 110838.

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.

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

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

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

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

Takahiro Sato, Christopher Batty, Takeo Igarashi, and Ryoichi Ando. 2018. Spatially adaptive long-term semi-Lagrangian method for accurate velocity advection. *Computational Visual Media* 4, 3 (2018), 6.

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*. 1–4.

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

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

Hagit Schechter and Robert Bridson. 2012. Ghost SPH for animating water. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–8.

DM Summers. 2000. A representation of bounded viscous flow based on Hodge decomposition of wall impulse. *J. Comput. Phys.* 158, 1 (2000), 28–50.

Jerry Tessendorf. 2015. Advection Solver Performance with Long Time Steps, and Strategies for Fast and Accurate Numerical Implementation. (2015).

Jerry Tessendorf and Brandon Pelfrey. 2011. The characteristic map for fast and efficient vfx fluid simulations. In *Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies. Ottawa, Canada.*

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

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.

S. Xiong, Z. Wang, M. Wang, and B. Zhu. 2022. A Clebsch method for free-surface vortical flow simulation. *ACM Trans. Graph.* 41, 4 (2022).

Shuqi Yang, Shiying Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. 2021. Clebsch gauge fluid. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–11.

Xi-Yuan Yin, Olivier Mercier, Badal Yadav, Kai Schneider, and Jean-Christophe Nave. 2021. A Characteristic Mapping method for the two-dimensional incompressible Euler equations. *J. Comput. Phys.* 424 (2021), 109781.

Xi-Yuan Yin, Kai Schneider, and Jean-Christophe Nave. 2023. A Characteristic Mapping Method for the three-dimensional incompressible Euler equations. *J. Comput. Phys.* (2023), 111876.

Xiao Zhai, Fei Hou, Hong Qin, and Aimin Hao. 2018. Fluid simulation with adaptive staggered power particles on gpus. *IEEE Transactions on Visualization and Computer Graphics* 26, 6 (2018), 2234–2246.

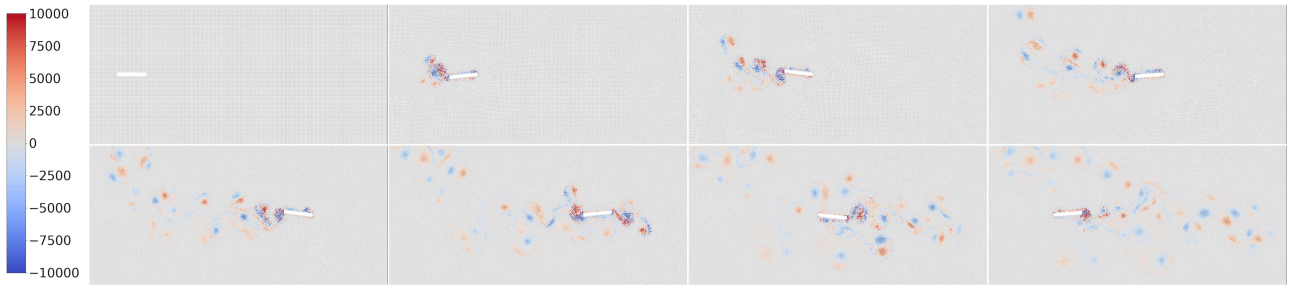


Fig. 5. A moving and rotating board in 2D traverses the domain from left to right, and then back.

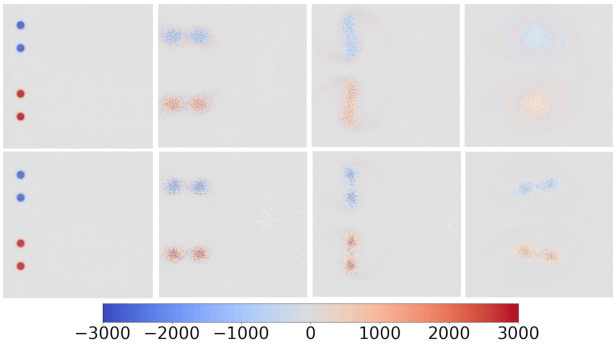


Fig. 6. Leapfrog vortices in 2D. PPM (*top*), our approach (*bottom*).

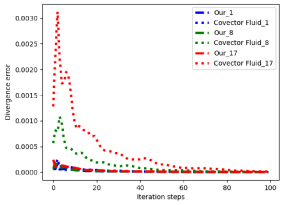


Fig. 7. The performance when using different flow map lengths is shown in the case of both the original Covector Fluid technique, and our method, where previous pressure integration is subtracted before projection. *Blue, green*, and *red* lines depict divergence error-interative step curve for conjugate gradient solver for poisson equation of flow map length 1, 8, and 17, respectively. We observe an improved rate of convergence by our method. This plot corresponds to Leapfrog example and convergence error is calculated by averaging $|\nabla \cdot \mathbf{u}|$ on particles.

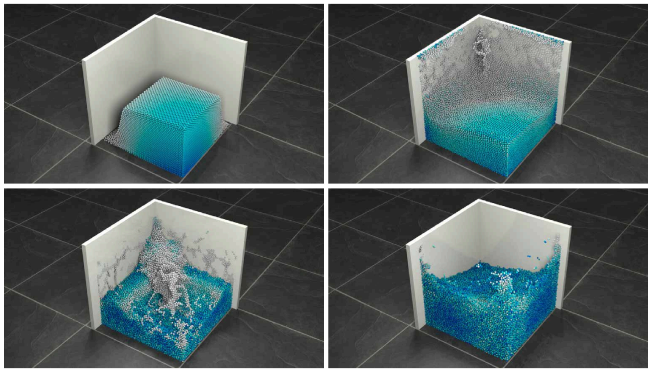


Fig. 8. Dam break (particle visualization).

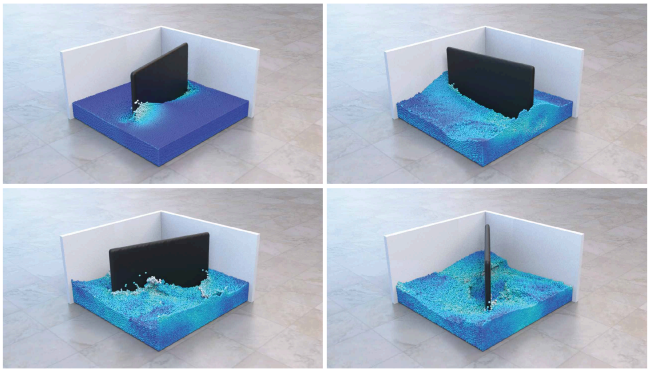
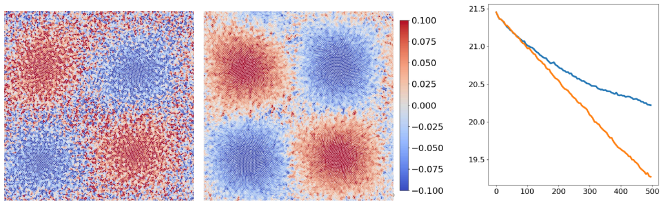
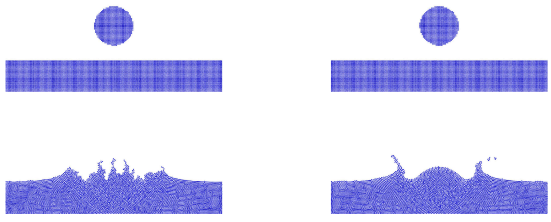


Fig. 9. Rotating board.



(a) PPM (b) Our method

Fig. 10. Qualitative and quantitative evaluation of simulating Taylor-Green vortices. (a) and (b) show the state for each method after 700 time steps. (c) shows the volume-averaged kinematic energy (y-axis) over time steps (x-axis): the energy dissipation for PPM (orange) and our method (blue) over 500 time steps. Here, coloring represents the magnitude and sign of vorticity.



(a) Without (b) Our method

Fig. 11. Ablation study on the effect of our novel free surface treatment. We observe the same ball of fluid particles being dropped into a still body of water.When not using our method (on the left), due to inaccuracies in the $\mathcal{T}_s^r$ approximation, strange shapes appear at the free surface. When using our method (on the right), the shape of the free surface is correct.

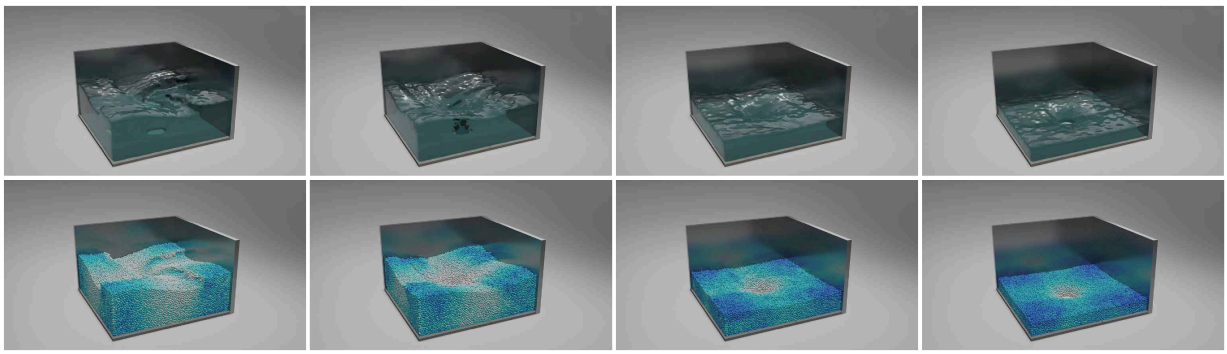


Fig. 12. Single sink. Surface rendering *(top)* and particle visualization *(bottom)*.

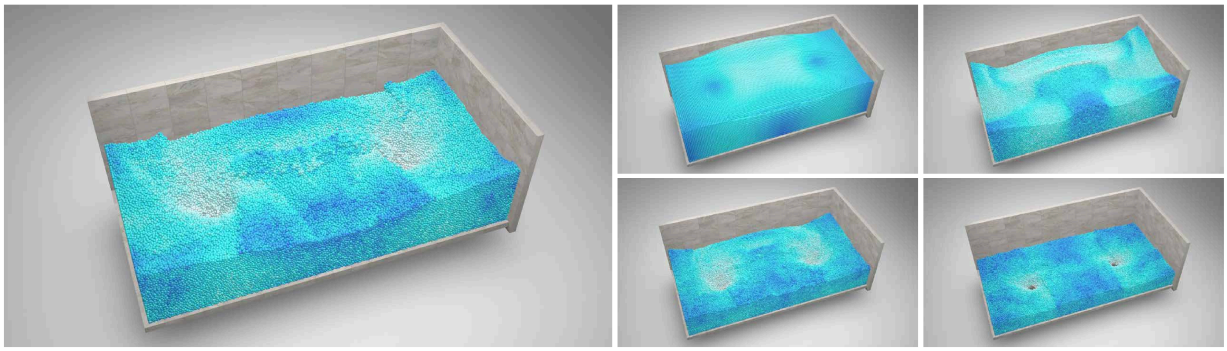


Fig. 13. Double sink (particle visualization).

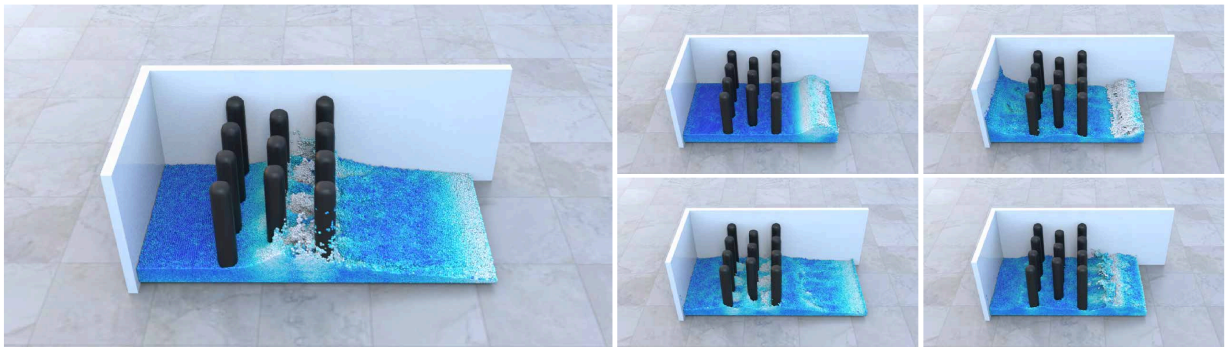


Fig. 14. Waves generated in a water tank with cylindrical obstacles.

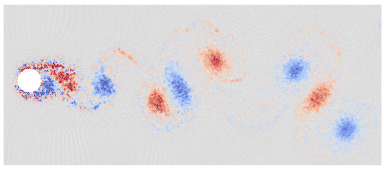


Fig. 15. Kármán vortex street.

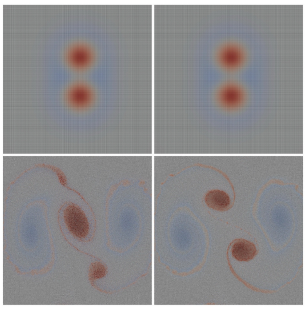


Fig. 16. Taylor vortices. Initial state *(top)* and at 300 steps *(bottom)*. PPM *(left)*, Ours *(right)*.

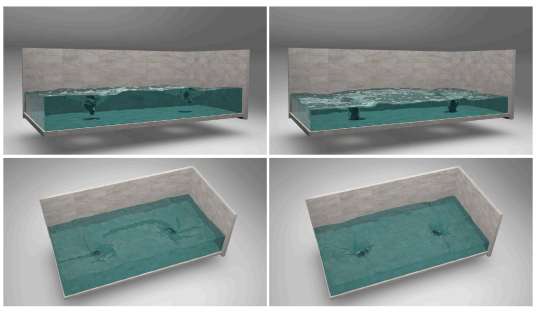


Fig. 17. Double sink (surface rendering).