An Eulerian Vortex Method on Flow Maps

SINAN WANG, Georgia Institute of Technology, USA and The University of Hong Kong, Hong Kong
YITONG DENG, Stanford University, USA
MOLIN DENG, Georgia Institute of Technology, USA
HONG-XING YU, Stanford University, USA
JUNWEI ZHOU, Purdue University, USA and University of Michigan, USA
DUOWEN CHEN, Georgia Institute of Technology, USA
TAKU KOMURA, The University of Hong Kong, Hong Kong
JIAJUN WU, Stanford University, USA
BO ZHU, Georgia Institute of Technology, USA

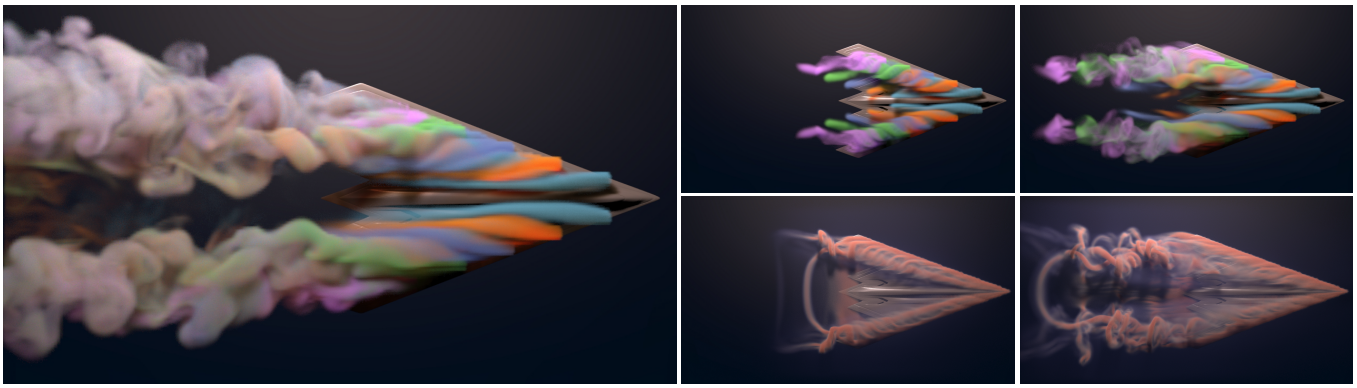

Fig. 1. Turbulent flow generated from a delta wing with zero viscosity and an angle-of-attack of 20°. Two pictures on the right illustrate smoke (top) and vorticity (bottom) snapshots at different time steps. A more stable spiral vortex structure along the two leading edges can be generated when viscosity is considered, as shown in Figure 17 (top).

We present an Eulerian vortex method based on the theory of flow maps to simulate the complex vortical motions of incompressible fluids. Central to our method is the novel incorporation of the flow-map transport equations for *line elements*, which, in combination with a bi-directional marching scheme for flow maps, enables the high-fidelity Eulerian advection of vorticity variables. The fundamental motivation is that, compared to impulse $\boldsymbol{m}$, which has been recently bridged with flow maps to encouraging results, vorticity $\boldsymbol{\omega}$ promises to be preferable for its numerical stability and physical interpretability. To realize the full potential of this novel formulation, we develop a new Poisson solving scheme for vorticity-to-velocity reconstruction that is both efficient and able to accurately handle the coupling near solid boundaries. We demonstrate the efficacy of our approach with a range of vortex simulation examples, including leapfrog vortices, vortex collisions, cavity flow, and the formation of complex vortical structures due to solid-fluid interactions.

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

Additional Key Words and Phrases: Fluid simulation, Vortex method, Flow map, Grid-based methods

ACM Reference Format:
Sinan Wang, Yitong Deng, Molin Deng, Hong-Xing Yu, Junwei Zhou, Duowen Chen, Taku Komura, Jiajun Wu, and Bo Zhu. 2024. An Eulerian Vortex Method on Flow Maps. *ACM Trans. Graph.* 43, 6, Article 268 (December 2024), 14 pages. https://doi.org/10.1145/3687996

Authors' Contact Information: Sinan Wang, swang3081@gatech.edu, Georgia Institute of Technology, USA and The University of Hong Kong, Hong Kong; Yitong Deng, yitongd@stanford.edu, Stanford University, USA; Molin Deng, mdeng47@gatech.edu, Georgia Institute of Technology, USA; Hong-Xing Yu, koven@cs.stanford.edu, Stanford University, USA; Junwei Zhou, zjw330501@gmail.com, Purdue University, USA and University of Michigan, USA; Duowen Chen, dchen322@gatech.edu, Georgia Institute of Technology, USA; Taku Komura, taku@cs.hku.hk, The University of Hong Kong, Hong Kong; Jiajun Wu, jiajunwu.cs@gmail.com, Stanford University, USA; Bo Zhu, bo.zhu@gatech.edu, Georgia Institute of Technology, USA.

ACM 1557-7368/2024/12-ART268
https://doi.org/10.1145/3687996

1 Introduction

Vortex methods have established themselves as a foundational computational method for incompressible flows with turbulence and complex vortical structures. In these methods, the Navier-Stokes equations written in the velocity form $\boldsymbol{u}$ are reformulated into their vorticity form, establishing vorticity

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{u}, \tag{1}$$

as the primary physical quantity for fluid representation. Such a vorticity-based formulation is favored by researchers in both theoretical and experimental fluid dynamics for its unambiguous physical

meaning, distinctive topological structures, and direct correlation with visually appealing vortical details. Over the past decade, extensive research effort across computational physics and computer graphics has been devoted to advancing vorticity-based incompressible flow solvers (see the work by Mimeau and Mortazavi [2021] for a comprehensive survey), exploring a rich set of data structures including particles [Angelidis 2017; Cottet et al. 2000; Park and Kim 2005; Selle et al. 2005; Zhang and Bridson 2014], filaments [Angelidis and Neyret 2005; Ishida et al. 2022; Padilla et al. 2019; Weißmann and Pinkall 2010], sheets [Brochu et al. 2012; Da et al. 2015; Pfaff et al. 2012], as well as their hybridization [Chern et al. 2017, 2016; Xiong et al. 2022; Yang et al. 2021].

Despite these innovative advancements, it stands as a peculiar (yet reasonable) fact that pure Eulerian vortex methods are very rare. Unlike their velocity-based counterparts, which benefit from the parallelizable and large-scale solver-friendly nature of Cartesian grids (e.g., the stable fluid solver [Stam 1999]), vortex methods typically require specific auxiliary data structures, usually particles or other forms of Lagrangian discretizations, to facilitate their temporal evolution. Particularly, owing to the challenges in robustly handling the vorticity stretching term and preserving both the structure and amount of vorticity during its advection, it is difficult to devise a numerical scheme that evolves vorticity on Cartesian grids in a purely Eulerian manner by directly adopting off-the-shelf advection schemes like Semi-Lagrangian [Jameson et al. 1981; Sawyer 1963] or BFECC [Kim et al. 2006; Selle et al. 2008], as in these cases the vorticity representation offers no significant advantages over velocity in capturing the fluid's vortical motion, which renders these efforts less appealing than the conventional approaches, particularly in the computer graphics community.

To this end, we propose a novel Eulerian vortex method tailored for graphical simulation applications. Our framework is based on the mathematical insight that the vorticity-based fluid equations can be elegantly and accurately solved using the transport equations for *line elements* on long-range flow maps, echoing recent advances in the impulse-based fluid methods in computer graphics [Deng et al. 2023; Nabizadeh et al. 2022], which have achieved state-of-the-art simulation accuracy through combining long-range flow maps with the *surface element* impulse. However, despite the encouraging results, a surface element like impulse is inherently limited by its tendency to rapidly and constantly increase in magnitude, which destabilizes the simulation. In comparison, the line element vorticity promises to offer a more advantageous pairing as it is numerically stable and physically interpretable, offering direct connections to the fluid's vortical structures, while enjoying the same low-dissipation advantages as impulse-based methods.

Such an observation motivates the design of our Eulerian vortex method that evolves the vorticity $\boldsymbol{\omega}$ based on long-range flow maps computed on Cartesian grids. First, we employ a bi-directional marching mechanism to evolve accurate flow maps and their Jacobians. Then, we employ an error-compensated advection scheme to compute our novel *line element vorticity advection* based on the computed flow maps. Finally, we devise a novel Poisson solving scheme to reconstruct velocity $\boldsymbol{u}$ from our evolved vorticity $\boldsymbol{\omega}$. For such reconstruction, standard methods [Ando et al. 2015; Elcott et al. 2007; Yin et al. 2023a; Zhang et al. 2015] utilize the vorticity-streamfunction formulation, which requires additional velocity potential solving in order to handle the solid boundary conditions. In comparison, we propose an alternative solution based on the velocity-vorticity formulation, where, instead of the streamfunction, the velocity serves as the unknown of a Poisson system, allowing for straightforward incorporation and direct enforcement of solid boundary conditions at the solver level. As discussed in [Cottet et al. 2000], it is possible that by means of Poincaré identity, we can enforce the solid boundary conditions by directly considering the relationship between the vorticity and the velocity field, thus bypassing the need for the streamfunction and potential. Drawing on this conceptual insight, we implement a novel velocity-vorticity solution using an efficient, matrix-free GPU-based Poisson solver to solve the system. We verify the correctness, versatility, and efficacy of our approach with a diverse set of challenging vortex simulation scenarios, including vortex shedding and development from moving solids, leapfrogging vortices, as well as vortex collision and reconnections.

2 Related Work

Grid-based fluid simulation. Since the seminal work of Stam [1999], grid-based solvers have been used to simulate a wide variety of physical phenomena [Fedkiw et al. 2001; Foster and Fedkiw 2001; Nguyen et al. 2002]. In addition to regular Cartesian grids, sparse and adaptive grid structures like Octree [Aanjaneya et al. 2017; Ando and Batty 2020; Losasso et al. 2006, 2004; Rasmussen et al. 2004], SPGrid [Setaluri et al. 2014], Multi-grid [McAdams et al. 2010], RLE grid [Chentanez and Müller 2011; Houston et al. 2006; Irving et al. 2006] and far-field grid [Zhu et al. 2013] have also been introduced to increase the effective resolution and the size of the simulation domain, as well as to reduce numerical dissipation.

Flow map method. Flow map advection, also referred to as the method of characteristic mapping (MCM), was first adopted for fluid simulation by Wiggert and Wylie [1976], and later introduced to the computer graphics community by Tessendorf and Pelfrey [2011]. Virtual particles are typically used to compute flow maps [Hachisuka 2005; Sato et al. 2018, 2017; Tessendorf 2015], while Qu et al. [2019] proposed a Semi-Lagrangian-like scheme to advect flow maps in the Eulerian manner to mitigate the time cost. Such methods significantly reduce numerical dissipation. Later, Nabizadeh et al. [2022] and Deng et al. [2023] combined the flow map with gauge fluid methods using impulse variables and verified the crucial role that the flow map plays in the accurate advection of these variables. Recently, Li et al. [2024] advanced this approach by developing a Lagrangian method that successfully handles the free surface case. We extend this to the vorticity-velocity Navier-Stokes equations, which is another form of gauge applied to the usual velocity-pressure form [Mercier et al. 2020; Yin et al. 2021, 2023b].

Vortex method. Vortex methods rewrite the incompressible Navier-Stokes equation by treating vorticity as a gauge variable for velocity. By explicitly advecting the vorticity, this approach naturally preserves fluid circulation. In order to reduce the numerical dissipation during the advection, diverse vorticity representations have been

proposed, ranging from particles [Angelidis 2017; Cottet et al. 2000; Park and Kim 2005; Selle et al. 2005; Zhang and Bridson 2014], to filaments [Angelidis and Neyret 2005; Ishida et al. 2022; Padilla et al. 2019; Weißmann and Pinkall 2010], segments [Xiong et al. 2021], sheets [Brochu et al. 2012; Da et al. 2015; Pfaff et al. 2012], and Clebsch level sets [Chern et al. 2017, 2016; Xiong et al. 2022; Yang et al. 2021]. For these Lagaragian vorticity representations, the velocity reconstruction from vorticity is often done with the Biot-Savart law, accelerated with the Fast Multipole Method (FMM) [Greengard and Rokhlin 1987]. Despite the enhanced ability to preserve the vortex structure, these vorticity representations are not as convenient to implement as the Eulerian vortex method when handling solid boundaries. Vortex methods with the vorticity-streamfunction formulation have been adopted by several researchers [Ando et al. 2015; Elcott et al. 2007; Yin et al. 2023a; Zhang et al. 2015]. In spite of the ability to handle solid boundaries, these methods require solving a potential component of the velocity. Existing Eulerian vortex methods that use the velocity-vorticity formulation avoid the streamfunction but suffer from poor vorticity preservation and inaccurate solid boundary conditions. For example, Huang and Li [1997] only addressed 2D cases and Liu [2001] set the vorticity boundary condition simply as $\boldsymbol{\omega}_b = 0$. Our method bypasses the use of streamfunction and potential without sacrificing the correctness in solid boundary handling, and pushes forward the state-of-the-art in vortex method simulations through the novel combination with flow map advection.

3 Physical Model

Naming Convention. In this paper, all subscripts are used to indicate axes (like x, y, z) while superscripts are used for cell or face indices (like i, j, k). All equations are presented in their matrix forms. We also present the definitions of main variables in Table 1.

3.1 Flow map

The forward flow map, $\boldsymbol{\phi}(\cdot, t)$, is defined as a function of space and time, mapping the initial position of a fluid particle at time 0 to its position at a subsequent time t. Analogously, the backward flow map, $\boldsymbol{\psi}(\cdot, t)$, is defined as the mapping from the position of a fluid particle at time t back to its original position at time 0.

For the forward flow map $\boldsymbol{\phi}$:

$$\begin{cases} \dfrac{\partial \boldsymbol{\phi}(\boldsymbol{X}, \tau)}{\partial \tau} = \boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{X}, \tau), \tau), \\ \boldsymbol{\phi}(\boldsymbol{X}, 0) = \boldsymbol{X}, \\ \boldsymbol{\phi}(\boldsymbol{X}, t) = \boldsymbol{x}. \end{cases} \tag{2}$$

Its inverse, the backward flow map $\boldsymbol{\psi}$ can then be defined as

$$\begin{cases} \boldsymbol{\psi}(\boldsymbol{x}, t) = \boldsymbol{x}, \\ \boldsymbol{\psi}(\boldsymbol{x}, 0) = \boldsymbol{X}. \end{cases} \tag{3}$$

In these definitions, $\boldsymbol{X}$ denotes the position of a material point at time 0, while $\boldsymbol{x}$ refers to the position of the same material point at time t. The Jacobians of the functions $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are denoted by $\mathcal{F}$ and $\mathcal{T}$ respectively: $\mathcal{F} := \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{X}}$, $\mathcal{T} := \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{x}}$. In particular, $\mathcal{F}$ is the forward Jacobian evaluated at the undeformed position $\boldsymbol{X}$, while $\mathcal{T}$ is the backward Jacobian evaluated at the deformed position $\boldsymbol{x}$.

Table 1. Summary of the main symbols and notations.

Notation	Type	Definition
$\boldsymbol{X}$	vector	material point position at initial state
$\boldsymbol{x}$	vector	material point position at time t
t	scalar	time
τ	scalar	dummy variable
$\boldsymbol{\phi}$	vector	forward map
$\boldsymbol{\psi}$	vector	backward map
$\mathcal{F}$	matrix	forward map gradients
$\mathcal{T}$	matrix	backward map gradients
$\mathcal{V}$	vector	velocity buffer
$\mathcal{S}$	vector	time buffer
$\boldsymbol{u}$	vector	velocity
$\boldsymbol{m}$	vector	impulse
$\boldsymbol{\omega}$	vector	vorticity
$\boldsymbol{s}$	vector	surface element
$\boldsymbol{l}$	vector	line element
n	scalar	number of steps between reinitializations

For a specific flow map defined on a moving particle, the evolution of the Jacobians $\mathcal{F}$ and $\mathcal{T}$ is governed by:

$$\begin{aligned} \frac{D\mathcal{F}}{Dt} &= \nabla \boldsymbol{u}\, \mathcal{F}, \\ \frac{D\mathcal{T}}{Dt} &= -\mathcal{T}\, \nabla \boldsymbol{u}. \end{aligned} \tag{4}$$

Here, $\frac{D(\cdot)}{Dt}$ represents the material derivative, which describes the rate of change of the Jacobians moving with the particle along its flow map's trajectory.

Line Elements and Surface Elements. Next, we define the flow map-based advection equations for line elements and surface elements , and we will show that vorticity is a line element and impulse is a surface element. Geometrically, line and surface elements amount to 2-forms and 1-forms in differential geometry (see [Nabizadeh et al. 2022] for detailed discussion). In an incompressible flow field $\boldsymbol{u}$, if the advection of a vector field $\boldsymbol{l}$ satisfies

$$\frac{D\boldsymbol{l}}{Dt} = (\nabla \boldsymbol{u})\, \boldsymbol{l}, \tag{5}$$

we define $\boldsymbol{l}$ as a *line element*. The advection of a line element can be characterized by the bidirectional flow map

$$\boldsymbol{l}(\boldsymbol{x}, t) = \mathcal{F}\, \boldsymbol{l}(\boldsymbol{\psi}(\boldsymbol{x}), 0), \tag{6}$$

$$\boldsymbol{l}(\boldsymbol{X}, 0) = \mathcal{T}\, \boldsymbol{l}(\boldsymbol{\phi}(\boldsymbol{X}), t). \tag{7}$$

Here, Eqn. (6) defines the forward evolution, and Eqn. (7) defines the time-reversed evolution. In Eqn (6), we have $\boldsymbol{\psi}(\boldsymbol{x})$ and $\boldsymbol{x}$ specifying the start and end point of the flow map path at time 0 and t. Conversely, in Eqn (7), by examining the backward path from time t to time 0, we can construct a mapping with $\boldsymbol{l}(\boldsymbol{X}, 0)$ denoting the line element located in $\boldsymbol{X}$ at time 0 and $\boldsymbol{l}(\boldsymbol{\phi}(\boldsymbol{X}), t)$ specifying its value at time t.

Similarly, if the advection of a vector field $\boldsymbol{s}$ satisfies

$$\frac{D\boldsymbol{s}}{Dt} = -(\nabla \boldsymbol{u})^T \boldsymbol{s}, \tag{8}$$

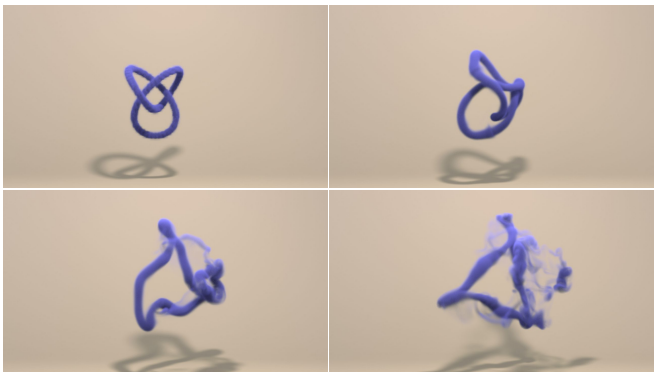

Fig. 2. The development of a vortex trefoil knot over time leads to its division into two vortex rings of varying sizes.

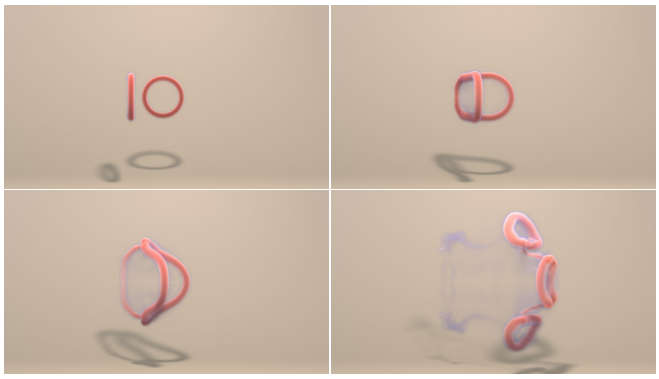

Fig. 3. The transformation of oblique vortex rings starts with two vortices colliding and merging into a single entity. It then undergoes several structural changes, eventually splitting into three distinct vortex rings.

we define $\boldsymbol{s}$ as a *surface element*. The evolution of a surface element can be characterized by its bidirectional flow maps as

$$\boldsymbol{s}(\boldsymbol{x},t) = \mathcal{T}^T \boldsymbol{s}\left(\boldsymbol{\psi}\left(\boldsymbol{x}\right),0\right), \tag{9}$$

$$\boldsymbol{s}(\boldsymbol{X},0) = \mathcal{F}^T \boldsymbol{s}\left(\boldsymbol{\phi}\left(\boldsymbol{X}\right),t\right). \tag{10}$$

When comparing Eqn. (6) and Eqn. (7) with Eqn. (9) and Eqn. (10), it becomes evident that *both line and surface elements can be transported using the same underlying flow map*. The only difference is the transpose on the Jacobians for surface elements. Consequently, if an accurate flow map is constructed, it can be utilized to transport line elements, surface elements, or a combination of both, by leveraging the same geometric mapping between two time instances. This observation motivates the pursuit of the most suitable representation that, when transported using high-fidelity flow maps, could offer the highest level of numerical accuracy and stability.

For a thorough mathematical analysis of the evolution of line and surface elements in a flow field, readers might refer to the works by Wu et al. [2007] and Truesdell [2018], where a comprehensive discussion and valuable insights into these concepts are provided from the differential geometry perspective.

3.2 Inviscid Flow with Vorticity Variables

We start with the inviscid incompressible fluid model described by the Euler equations:

$$\begin{cases} \dfrac{D\boldsymbol{u}}{Dt} = -\dfrac{1}{\rho}\nabla \boldsymbol{p}, \\ \nabla \cdot \boldsymbol{u} = 0, \end{cases} \tag{11}$$

where $\boldsymbol{u}, \rho, \boldsymbol{p}$ represent velocity, density, and pressure respectively. The first equation specifies momentum conservation, and the second equation is the incompressibility condition. Taking the curl of the momentum equation and substituting the incompressibility condition, we obtain the vorticity form of the Euler equation:

$$\frac{D\boldsymbol{\omega}}{Dt} = \left(\nabla \boldsymbol{u}\right)\boldsymbol{\omega}, \tag{12}$$

where $\left(\nabla \boldsymbol{u}\right)\boldsymbol{\omega}$ denotes the vortex-stretching term. The relation between velocity and vorticity is specified by the Poisson equation

$$\nabla \cdot \nabla \boldsymbol{u} = -\nabla \times \boldsymbol{\omega}, \tag{13}$$

which at the domain boundary is subject to

$$\boldsymbol{u} = \boldsymbol{u}_{wall}, \tag{14}$$

and Eqn. (1), where Eqn. (14) is the velocity boundary condition and Eqn. (1) is the compatibility condition of velocity and the vorticity field in the presence of boundaries [Cottet et al. 2000]. We refer readers to [Cottet et al. 2000; Mimeau and Mortazavi 2021] for more details on vorticity-based incompressible fluid models.

3.3 Vorticity on Flow Maps

As evidenced in Eqn. (12), *vorticity is a line element*. Therefore, we can leverage the flow map schemes for line elements as presented in Eqn. (6) and Eqn. (7) for its transport. Specifically, by establishing a bi-directional flow map between time 0 and time t, we have

$$\boldsymbol{\omega}\left(\boldsymbol{x},t\right) = \mathcal{F}\,\boldsymbol{\omega}\left(\boldsymbol{\psi}\left(\boldsymbol{x}\right),0\right), \tag{15}$$

$$\boldsymbol{\omega}\left(\boldsymbol{X},0\right) = \mathcal{T}\,\boldsymbol{\omega}\left(\boldsymbol{\phi}\left(\boldsymbol{X}\right),t\right). \tag{16}$$

Here, in Eqn. (15), $\boldsymbol{\omega}\left(\boldsymbol{x},t\right)$ denotes the vorticity of a material point located in $\boldsymbol{x}$ at time t and $\boldsymbol{\omega}\left(\boldsymbol{\psi}\left(\boldsymbol{x}\right),0\right)$ is the vorticity of the same material point at time 0. Conversely, in Eqn. (16), $\boldsymbol{\omega}\left(\boldsymbol{X},0\right)$ denotes the vorticity of a material point located in $\boldsymbol{X}$ at time 0 and $\boldsymbol{\omega}\left(\boldsymbol{\phi}\left(\boldsymbol{X}\right),t\right)$ is the vorticity of the same material point at time t. In order to establish the forward flow map Jacobian $\mathcal{F}$ that corresponds to the trajectory from position $\boldsymbol{X}$ and time 0 to position $\boldsymbol{x}$ and time t, we have to evolve $\mathcal{F}$ in reverse. The reverse evolution of $\mathcal{F}$ and $\mathcal{T}$ is governed by:

$$\begin{aligned} \frac{D\mathcal{F}}{Dt} &= \mathcal{F}\,\nabla \boldsymbol{u}, \\ \frac{D\mathcal{T}}{Dt} &= -\nabla \boldsymbol{u}\,\mathcal{T}. \end{aligned} \tag{17}$$

For any fluid particle located at position $\boldsymbol{x}$ at time t, the result of the reverse evolution of $\mathcal{F}$, starting from an identity map at time t with position $\boldsymbol{x}$ and moving backward to time 0 with its original position $\boldsymbol{X}$, represents the Jacobian $\mathcal{F}$ of the forward flow map $\boldsymbol{\phi}$ from the original position $\boldsymbol{X}$ of the particle at time 0 to the current position $\boldsymbol{x}$ at time t. More details about the reverse evolution and its mathematical proof can be found in [Zhou et al. 2024].

Fig. 4. The picture depicts an eagle under the behavior of gliding (256 x 256 x 128). Vortices are created at the tail and back of the wings.

Fig. 5. Head-on vortex collision. Two vortex rings are positioned opposite to each other. They merge as one ring and expand upon collision, which ultimately leads to the ring breaking up into a series of smaller, radially arranged secondary vortices.

3.4 Relation to Impulse Variables

To further elucidate the motivation of our design choices, we briefly describe the relationship between vorticity and impulse variables, including an illustrative example to highlight their differences in numerical performance. As indicated in the literature [Cortez 1995], the impulse-form Euler equations write as:

$$\begin{cases} \dfrac{D\boldsymbol{m}}{Dt} = -\left(\nabla \boldsymbol{u}\right)^T \boldsymbol{m}, \\ \boldsymbol{u} = \mathcal{P}\left(\boldsymbol{m}\right), \end{cases} \tag{18}$$

where the first equation describes the evolution of the impulse variable $\boldsymbol{m}$, and the second equation establishes the divergence-free condition for velocity $\boldsymbol{u}$ by the projection operator $\mathcal{P}$. It is obvious that *impulse is a surface element* by comparing Eqn. (8) and (18). Consequently, the forward and backward flow maps for surface elements, as delineated in Eqn. (9) and (10), can be applied to determine its evolution.

3.5 Choosing Vorticity over Impulse

We choose to track vorticity instead of impulse on a bi-directional flow map, positing that line elements are more effective than surface elements when being numerically evolved with the same flow maps discretized on Cartesian grids. The rationale for this choice is twofold: *First*, the strength of surface elements increases significantly and constantly as they evolve in a vortical flow field, which demands high precision in time integration. Specifically, as the impulse value increases, the numerical precision required by the projection solver to compute $\boldsymbol{u}$ from $\boldsymbol{m}$ becomes critical, making the process increasingly susceptible to the accumulation of numerical errors during impulse evolution. This issue, highlighted in several prior studies and particularly emphasized by Summers and Chorin [1996], is recognized as a significant barrier to using impulse-based representations to address vortex dynamics. *Secondly*, the physical interpretation of vorticity directly binds to vortical structures in fluid dynamics. In particular, the distribution of vorticity is closely tied to the evolution of rotational flow motion in a fluid field, a relationship that becomes less clear when using the impulse variable to represent the same fluid motion.

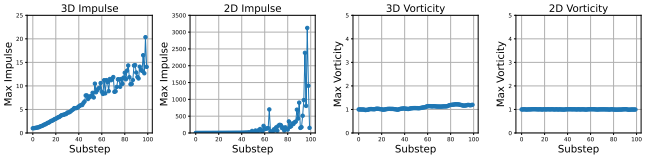

Fig. 6. Motivational Experiment. The increase factor comparison of the maximum norm of impulse and vorticity in leapfrog experiments. For impulse, the maximum increase factors in 100 steps are **20.35** and **3124.27** for 3D and 2D, respectively, while the maximum increase factors for vorticity are **1.22** and **1.01**, indicating large instability induced by using impulse.

We experimentally demonstrate this insight through leapfrog experiments in 2D and 3D. As depicted in Figure 6, without reinitialization, the maximum impulse increases rapidly. In contrast, the maximum vorticity remains stable throughout the simulation.

4 Velocity Reconstruction

As established in Section 3, we see that vorticity $\boldsymbol{\omega}$ is more compatible with flow maps than impulse $\boldsymbol{m}$ due to its numerical stability and physical interpretability. While reconstructing $\boldsymbol{u}$ from $\boldsymbol{m}$ essentially involves a pressure projection step exactly as in conventional Eulerian methods, reconstructing $\boldsymbol{u}$ from $\boldsymbol{\omega}$ requires the solution of a vector Poisson equation as in Eqn. (13). The standard method for such reconstruction is through the streamfunction, decomposing the velocity into its vortical and potential components [Ando et al. 2015; Elcott et al. 2007; Yin et al. 2023a; Zhang et al. 2015]. However, we observe that such a decomposition is not necessary because the sole functionality of the potential component is to enforce the velocity boundary condition Eqn. (14) and the compatibility condition Eqn. (1) near the solid boundary [Cottet et al. 2000]. Instead, inspired by the approach with the Poincaré identity discussed in Section 4.3 of [Cottet et al. 2000], we directly consider the relationship between velocity and vorticity, allowing the vorticity near the solid boundaries to become unknowns to satisfy the boundary conditions.

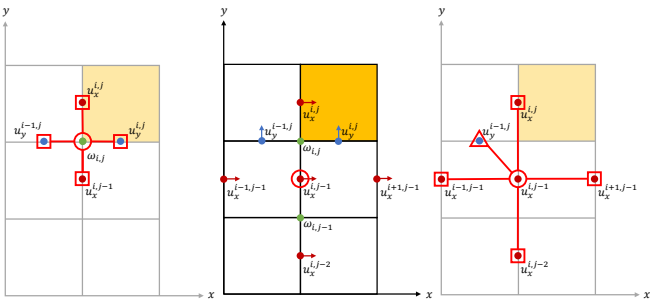

Fig. 7. Solids are represented by yellow cells. The left depicts the vorticity on the boundary, $\omega^{i,j}$ (circles), which is constrained by the four neighboring velocities (squares). The middle depicts the finite difference stencil around velocity component $u_x^{i,j-1}$, where one of the RHS variables $\omega^{i,j}$ is no longer known due to the compatibility condition. The right depicts the four neighboring variables (squares) that play a part in solving $u_x^{i,j-1}$, with the additional u_y (triangles) emerging from the compatibility condition.

That is to say, by explicitly specifying the velocity as the Dirichlet boundary condition and implicitly enforcing the velocity-vorticity compatibility near the solid boundary, we can bypass the use of the streamfunction and the potential while still enforcing the correct boundary conditions. In this section, we propose our revised discrete formulation of the vector Poisson equation, which carefully handles the vorticity-velocity coupling near the boundary without the need for the potential component. Also, we propose a unified GPU-based solver that provides high efficiency without sacrificing physical fidelity.

4.1 Solid Boundary Condition

In this section, we concretely illustrate the challenge in handling the boundary conditions in formulating the Poisson equation, by considering the 2D case as depicted in Figure 7. By discretizing Eqn. (13) using finite difference on a staggered marker-and-cell (MAC) grid as shown in Figure 7, we can write it out as a system of scalar equations on staggered locations. We use the equation at $u_x^{i,j-1}$ as an example:

$$4u_x^{i,j-1} - u_x^{i+1,j-1} - u_x^{i,j-2} - u_x^{i-1,j-1} - u_x^{i,j} = (\omega^{i,j} - \omega^{i,j-1}) \cdot \Delta x, \tag{19}$$

where $u_x^{i,j-1}$ represents the horizontal velocity on the face $(i, j-1)$. The system can be solved if the RHS contains only known variables.

In cases where both $\omega^{i,j}$ and $\omega^{i,j-1}$ are in the fluid's interior, they are indeed known variables that are prescribed by Eqn. (15). However, as illustrated in Figure 7 (left), this ceases to be the case near the boundary, since the velocity boundary condition Eqn. (14) propagates to the vorticity $\boldsymbol{\omega}$ through the compatibility condition Eqn. (1). The compatibility requires that the velocity and the vorticity field need to satisfy Eqn. (1) everywhere in the simulation domain. Specifically, consider Eqn. (19) for variable $u_x^{i,j-1}$ in the middle of Figure 7, which involves a RHS variable $\omega^{i,j}$ that is constrained to its neighboring velocities $u_y^{i-1,j}$, $u_y^{i,j}$, $u_x^{i,j-1}$, $u_x^{i,j}$ by the compatibility condition:

$$\omega^{i,j} = \frac{(u_y^{i,j} - u_y^{i-1,j}) - (u_x^{i,j} - u_x^{i,j-1})}{\Delta x}. \tag{20}$$

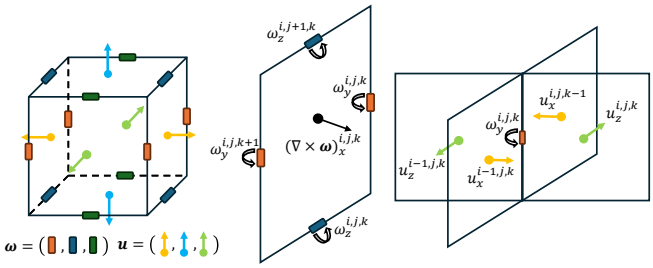

Fig. 8. Illustration of our discretization scheme. The curl operator maps edges to faces (middle) or faces to edges (right).

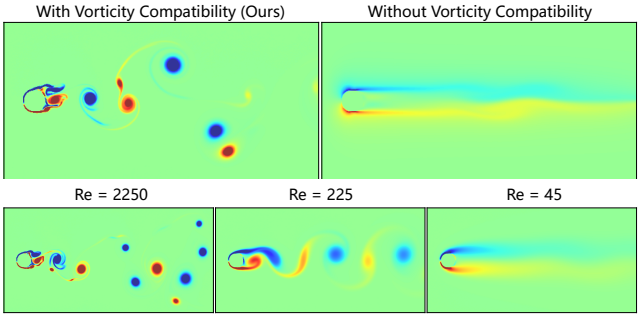

Fig. 9. The top row shows an ablation study for the velocity-vorticity compatibility, which is the comparison of the 150$^{\text{th}}$ frame of the 2D von Kármán vortex street example with a high Reynolds number (Re) = 2250. The bottom row shows that our method accurately models the 2D von Kármán vortex street with varying Re.

Since both $u_y^{i,j}$ and $u_x^{i,j}$ are on solid faces, they are dictated by the wall velocities $u_{y,wall}^{i,j}$ and $u_{x,wall}^{i,j}$. Consequently, $\omega^{i,j}$ becomes an unknown variable dependent on $u_x^{i,j-1}$ and $u_y^{i-1,j}$.

4.2 Revised Poisson Equation

To this end, we need to eliminate the unknown variables on the RHS of Eqn. (19). By substituting $\omega^{i,j}$ on the RHS, we obtain the revised equation for $u_x^{i,j-1}$:

$$\begin{aligned} 3u_x^{i,j-1} - u_x^{i+1,j-1} - u_x^{i,j-2} - u_x^{i-1,j-1} + u_y^{i-1,j} \\ = u_{y,wall}^{i,j} - \omega^{i,j-1} \cdot \Delta x. \end{aligned} \tag{21}$$

It can be observed from the fact that $u_x^{i,j-1}$ is now dependent on $u_y^{i-1,j}$, that such a revised system of equations will be coupled across spatial dimensions, i.e., it can no longer be solved using a generic scalar Poisson solver by treating each dimension separately. For the general case in 3D, u_x has six neighboring velocities, with two on each axis. The y and z-axis neighbors differ from the x-axis neighbors due to an intervening vorticity variable. If this variable is not within the fluid (e.g., on a solid boundary), we adjust the LHS by subtracting one from the coefficient of u_x, removing the corresponding velocity, and adding or subtracting unknowns from other axes within the fluid depending on their relative positions, as shown by an analogous 2D example in Figure 7. Then, we adjust the RHS by removing such intervening vorticities and moving the known velocities on the walls to the RHS. More details and algorithms are discussed in the supplementary materials.

4.3 Coupled Poisson Solver

To solve this revised equation system efficiently and accurately, we develop a GPU-based matrix-free Multi-Grid Preconditioned Conjugate Gradient (MGPCG) solver, whose implementation details are given in the supplementary material. Our solver is a geometric multi-grid solver implemented in Taichi [Hu et al. 2019], with solid boundary conditions being enforced at coarser levels with coarsened solids. The restriction, prolongation, and smoothing are implemented following [McAdams et al. 2010]. On one hand, as seen in Figure 9 (top), our method is able to significantly improve the simulation realism with its physically correct treatment of vorticity-velocity coupling near the boundary. On the other hand, as shown in Table 2, by leveraging the matrix-free nature of our solver, we are able to conveniently, without incurring a hindering memory overhead, harness the parallel computation of modern GPUs to process multiple dimensions concurrently.

5 Numerical Algorithm

Leveraging our novel vorticity-to-velocity method as its core, we build a highly accurate simulation method. As shown in Algorithm 1, our simulation time step is completed in the following stages:

(1) Compute forward and backward flow maps (steps 2-7);
(2) Compute vorticity advection using flow maps (step 8);
(3) Compute velocity from vorticity (step 9).

5.1 Discretization

As shown in Figure 8, $\boldsymbol{u}$ and $\boldsymbol{\omega}$ are stored on faces and edges, in line with their physical semantics as line and surface elements. Differential operators such as curl and divergence can be defined based on Stokes's Theorem. We refer to [Hirani 2003] for details.

Algorithm 1 Time Integration

Initialize: $\boldsymbol{u}, \boldsymbol{w}$ to initial velocity and vorticity;

1: **while** simulating **do**
2: **Reinitialize** every n steps;
3: Compute Δt with $\boldsymbol{u}$ and CFL number;
4: $\boldsymbol{u}_{\text{mid}} \leftarrow \textbf{MidPoint}(\boldsymbol{u}, \boldsymbol{\omega}, \Delta t)$;
5: Store $\boldsymbol{u}_{\text{mid}}$ in the buffer $\mathcal{V}$ and Δt in $\mathcal{S}$;
6: $\boldsymbol{\psi}, \mathcal{F} \leftarrow \textbf{Backtrace}(\mathcal{V}, \mathcal{S})$;
7: $\boldsymbol{\phi}, \mathcal{T} \leftarrow \textbf{March}(\boldsymbol{u}_{\text{mid}}, \Delta t, \boldsymbol{\phi}, \mathcal{T})$;
8: $\hat{\boldsymbol{\omega}} \leftarrow \textbf{BFECC}(\boldsymbol{\omega}_{\text{init}}, \boldsymbol{\psi}, \mathcal{F}, \boldsymbol{\phi}, \mathcal{T})$;
9: $\boldsymbol{u} \leftarrow \Delta^{-1}(-\nabla \times \hat{\boldsymbol{\omega}})$.
10: **end while**

5.2 Time Integration

Bi-directional March. For each step, we first construct flow map quantities using all previously stored mid-point velocity fields. In this process, $\boldsymbol{\phi}$ and $\mathcal{T}$ are evolved forward by the Runge-Kutta method. We adopt the bi-directional march from [Deng et al. 2023]. Consequently, $\boldsymbol{\psi}$ and $\mathcal{F}$ are evolved backward in the same manner as $\boldsymbol{\phi}$ and $\mathcal{T}$, with time reversed. Note that, unlike in [Deng et al. 2023] where $\boldsymbol{\psi}$ and $\mathcal{T}$ are evolved together, we now need to evolve $\mathcal{F}$ in reverse with $\boldsymbol{\psi}$, due to the difference in Eqn. (6) and Eqn. (9). The detailed algorithm is outlined in Algorithm 2 and Algorithm 3.

Algorithm 2 Vorticity Flow Map Computation for Step j

Input: $j, \mathcal{V}, \mathcal{S}, \boldsymbol{\phi}, \mathcal{T}$.
Output: $\boldsymbol{\psi}, \mathcal{F}, \boldsymbol{\phi}, \mathcal{T}$.

Estimate midpoint velocity $\boldsymbol{u}_{\text{mid}}$ according to Algorithm 4;
$\boldsymbol{\psi} \leftarrow \textbf{id}_\Omega, \mathcal{F} \leftarrow \boldsymbol{I}$;
for l in j ... 0 **do**
  Fetch $\boldsymbol{u}_{\text{mid},l}$ and Δt_l from the buffer $\mathcal{V}$ and $\mathcal{S}$;
  March $\boldsymbol{\psi}, \mathcal{F}$ with Algorithm 3, using $\boldsymbol{u}_{\text{mid},l}$ and $-\Delta t_l$;
end for
March $\boldsymbol{\phi}, \mathcal{T}$ with Algorithm 3, using $\boldsymbol{u}_{\text{mid}}$ and Δt_j.

Algorithm 3 Vorticity Joint RK4 for $\boldsymbol{\psi}$ and $\mathcal{F}$

Input: $\boldsymbol{u}, \boldsymbol{\psi}, \mathcal{F}, \Delta t$.
Output: $\boldsymbol{\psi}_{\text{next}}, \mathcal{F}_{\text{next}}$.

$(\boldsymbol{u}_1, \boldsymbol{\nabla u}|_1) \leftarrow \textbf{Interpolate}(\boldsymbol{u}, \boldsymbol{\psi})$;
$\frac{\partial \mathcal{F}}{\partial t}|_1 \leftarrow \mathcal{F}\,\boldsymbol{\nabla u}|_1$;
$\boldsymbol{\psi}_1 \leftarrow \boldsymbol{\psi} + 0.5 \cdot \Delta t \cdot \boldsymbol{u}_1$;
$\mathcal{F}_1 \leftarrow \mathcal{F} - 0.5 \cdot \Delta t \cdot \frac{\partial \mathcal{F}}{\partial t}|_1$;
$(\boldsymbol{u}_2, \boldsymbol{\nabla u}|_2) \leftarrow \textbf{Interpolate}(\boldsymbol{u}, \boldsymbol{\psi}_1)$;
$\frac{\partial \mathcal{F}}{\partial t}|_2 \leftarrow \mathcal{F}_1 \boldsymbol{\nabla u}|_2$;
$\boldsymbol{\psi}_2 \leftarrow \boldsymbol{\psi} + 0.5 \cdot \Delta t \cdot \boldsymbol{u}_2$;
$\mathcal{F}_2 \leftarrow \mathcal{F} - 0.5 \cdot \Delta t \cdot \frac{\partial \mathcal{F}}{\partial t}|_2$;
$(\boldsymbol{u}_3, \boldsymbol{\nabla u}|_3) \leftarrow \textbf{Interpolate}(\boldsymbol{u}, \boldsymbol{\psi}_2)$;
$\frac{\partial \mathcal{F}}{\partial t}|_3 \leftarrow \mathcal{F}_2 \boldsymbol{\nabla u}|_3$;
$\boldsymbol{\psi}_3 \leftarrow \boldsymbol{\psi} + \Delta t \cdot \boldsymbol{u}_3$;
$\mathcal{F}_3 \leftarrow \mathcal{F} - \Delta t \cdot \frac{\partial \mathcal{F}}{\partial t}|_3$;
$(\boldsymbol{u}_4, \boldsymbol{\nabla u}|_4) \leftarrow \textbf{Interpolate}(\boldsymbol{u}, \boldsymbol{\psi}_3)$;
$\frac{\partial \mathcal{F}}{\partial t}|_4 \leftarrow \mathcal{F}_3 \boldsymbol{\nabla u}|_4$;
$\boldsymbol{\psi}_{\text{next}} \leftarrow \boldsymbol{\psi} + \Delta t \cdot \frac{1}{6} \cdot (\boldsymbol{u}_1 + 2 \cdot \boldsymbol{u}_2 + 2 \cdot \boldsymbol{u}_3 + \boldsymbol{u}_4)$;
$\mathcal{F}_{\text{next}} \leftarrow \mathcal{F} - \Delta t \cdot \frac{1}{6} \cdot (\frac{\partial \mathcal{F}}{\partial t}|_1 + 2 \cdot \frac{\partial \mathcal{F}}{\partial t}|_2 + 2 \cdot \frac{\partial \mathcal{F}}{\partial t}|_3 + \frac{\partial \mathcal{F}}{\partial t}|_4)$.

Algorithm 4 Vorticity Midpoint Method

Input: $\boldsymbol{u}, \boldsymbol{\omega}, \Delta t$.
Output: $\boldsymbol{u}_{\text{mid}}$.

Reset $\boldsymbol{\psi}, \mathcal{F}$ to identity;
March $\boldsymbol{\psi}, \mathcal{F}$ with Algorithm 3, using $\boldsymbol{u}$ and $-0.5\Delta t$;
$\boldsymbol{\omega}_{\text{mid}} \leftarrow \mathcal{F}\boldsymbol{\omega}(\boldsymbol{\psi})$;
$\boldsymbol{u}_{\text{mid}} \leftarrow \textbf{Poisson}(\boldsymbol{\omega}_{\text{mid}})$.

Algorithm 5 Error-compensated Vorticity Pullback

Input: $\boldsymbol{\omega}_{init}, \boldsymbol{\psi}, \mathcal{T}, \boldsymbol{\phi}, \mathcal{F}$.
Output: $\boldsymbol{u}$.

$\bar{\boldsymbol{\omega}} \leftarrow \mathcal{F}\boldsymbol{\omega}_{init}(\boldsymbol{\psi})$;
$\bar{\boldsymbol{\omega}}_{init} \leftarrow \mathcal{T}\bar{\boldsymbol{\omega}}(\boldsymbol{\phi})$;
$\boldsymbol{e} \leftarrow 0.5 \cdot (\bar{\boldsymbol{\omega}}_{init} - \boldsymbol{\omega}_{init})$;
$\bar{\boldsymbol{e}} \leftarrow \mathcal{F}\boldsymbol{e}(\boldsymbol{\psi})$;
$\hat{\boldsymbol{\omega}} \leftarrow \bar{\boldsymbol{\omega}} - \bar{\boldsymbol{e}}$;
$\boldsymbol{\omega} \leftarrow \textbf{Clamp}(\hat{\boldsymbol{\omega}})$;
$\boldsymbol{u} \leftarrow \textbf{Poisson}(\boldsymbol{\omega})$.

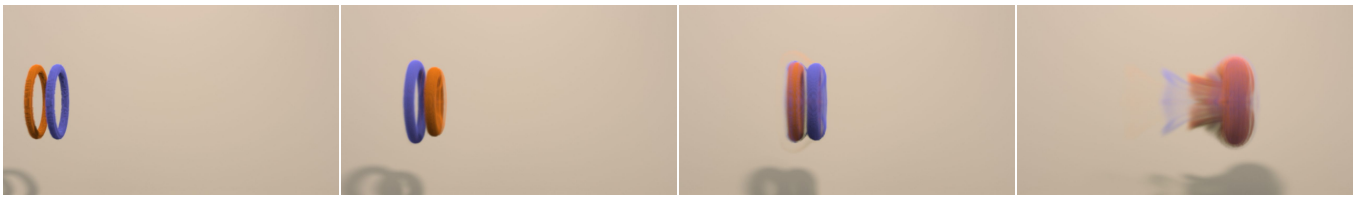

Fig. 10. Leapfrog vortices in 3D. The two rings remain separated after the 4th leap.

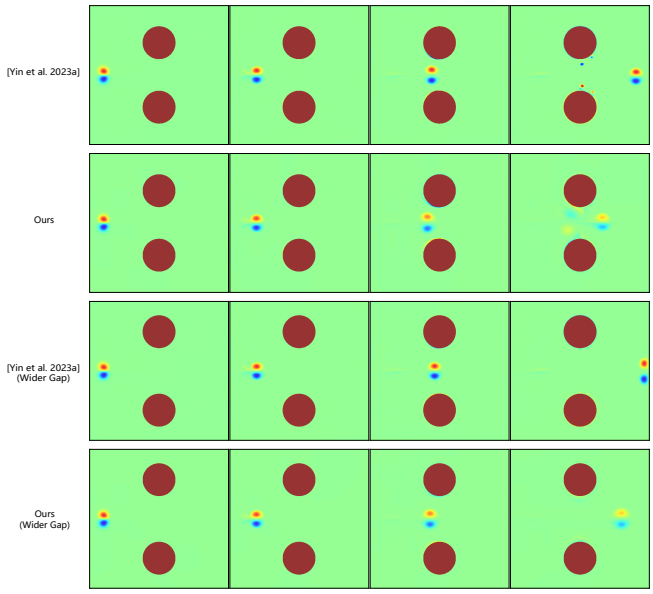

Fig. 11. A vortex pair successfully passes through a gap between two disks, demonstrating that by implicitly modeling the harmonic component, our result is consistent with Fluid Cohomology [Yin et al. 2023a], where the harmonic component is explicitly modeled.

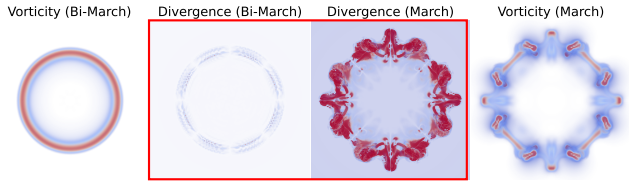

Fig. 12. **Vorticity Divergence Test**. Here we show an ablation study for the bi-directional marching applied for the Eulerian vortex method, which shows the comparison of the 50th frame of the head-on vortex collision for the divergence of vorticity, which is theoretically **zero** as it is the curl of velocity. Without a bi-directional march (right), the highest divergence is **45538.9922**. With bi-directional march (left), the highest divergence significantly reduces to **559.9824**, a reduction by nearly **80** times. A clear contrast can be seen between the two images highlighted with red boxes.

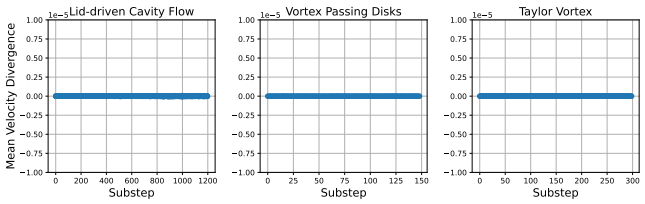

Fig. 13. **Velocity Divergence Test**. Using three examples, we verify that our reconstructed velocity field is divergence free.

Midpoint Method. For time integration, We adopt the midpoint method, following Nabizadeh et al. [2022]. To obtain $\boldsymbol{\psi}$ and $\mathcal{F}$ for advecting the vorticity, we first set $\boldsymbol{\psi}$ and $\mathcal{F}$ to its original position and identity map respectively. Then $\boldsymbol{\psi}$ and $\mathcal{F}$ are evolved backward together by a half step. Finally, we can utilize Eqn. (15) to compute the midpoint vorticity. The detailed algorithm is outlined in Algorithm 4.

Vorticity Advection using Flow Maps. Leveraging the highly accurate bi-directional flow map, we adopt BFECC [Kim et al. 2005] to perform advection in both directions. We begin by using the backward flow map $\boldsymbol{\psi}$ to advect the initial vorticity field to the current time step, represented as $\boldsymbol{\omega}(\boldsymbol{x}, t) = \mathcal{F}\,\boldsymbol{\omega}(\boldsymbol{\psi}(\boldsymbol{x}), 0)$. Next, the forward flow map $\boldsymbol{\phi}$ is applied to advect $\boldsymbol{\omega}(\boldsymbol{x}, t)$ back to the initial time step, given by $\bar{\boldsymbol{\omega}}(\boldsymbol{X}, 0) = \mathcal{T}\,\boldsymbol{\omega}(\boldsymbol{\phi}(\boldsymbol{X}), t)$. Then we compare the error between $\bar{\boldsymbol{\omega}}(\boldsymbol{X}, 0)$ and $\boldsymbol{\omega}(\boldsymbol{X}, 0)$ for computing the BFECC error. The detailed algorithm is outlined in Algorithm 5.

5.3 Implementational Details

Reinitialization. During the simulation, the flow map quantities may undergo significant distortion, necessitating reinitialization every n steps, where n is as a parameter which may be tuned for different simulation scenarios. At every reinitialization step, the initial vorticity $\boldsymbol{\omega}_0$ is set to the curl of the current velocity. The forward map $\boldsymbol{\phi}$ and the backward Jacobian $\mathcal{T}$ are set to the grid positions and identity map.

External forces and viscosity. We adopt the accumulation buffer from [Qu et al. 2019] for external forces and viscosity. In each step, we compute the current vorticity change and add this change back to the buffer at its undeformed position with forward map $\boldsymbol{\phi}$.

6 Evaluation

In this section, we will first verify the correctness of our revised Poisson scheme and boundary treatment; we will then provide the performance gains of our coupled approach and the performance measurements; next, we will compare our method to standard 2D and 3D benchmarks; finally, we will exhibit our simulation results with complex 3D scenarios with moving solid objects. All experimental settings are included in the supplementary materials. Compared

to existing vortex methods, our approach demonstrates significantly improved capabilities in capturing physical phenomena and preserving vortical details. All performance analysis are conducted on a laptop equipped with a 13th Gen Intel(R) Core(TM) i9-13900HX processor (2.20 GHz), 32 GB of RAM, and an NVIDIA GeForce RTX 4080 Laptop GPU. We use Taichi [Hu et al. 2019] version 1.6.0 as our parallel programming tool.

6.1 Solid Boundary Experiments and Divergence Test

2D von Kármán vortex street. Figure 9 (bottom) shows fluid inflow at 0.16 m/s interacting with a disk of 0.141 m diameter, resulting in three vortex shedding patterns under varying viscosities. Our method accurately replicates diverse shedding behaviors across different Reynolds numbers (Re), in line with physical experiments [Blevins 1990]. Simulations exhibit turbulent mixing at high Re (2250), a periodic vortex street at moderate Re (225), and a laminar wake without vortices at low Re (45). This aligns well with the observed behavior across Re = 60–5000 [Wu 2003].

Velocity-vorticity compatibility. Figure 9 (top) shows a comparison of the 150th frame of the 2D von Kármán vortex street example with a high Re (2250). The right shows the boundary treatment that only enforces the velocity boundary condition. Without the velocity-vorticity compatibility, the system fails to generate vortices.

Harmonic component. Figure 11 shows the vorticity visualization of a vortex pair passing through two disks in 2D, along with a comparison with Fluid Cohomology [Yin et al. 2023a], which explicitly models the harmonic component. As described in [Yin et al. 2023a], the expected behavior for the vortex pair of a vortex method that correctly handles the harmonic component is to pass through the two disks. Ours can pass through the two disks as expected, further verifying the correctness of our method, though with a bit more dissipation compared to [Yin et al. 2023a].

Lid-driven cavity flow. Figure 14 shows the streamline of the 2D Lid-driven cavity flow where the velocity of the top boundary is fixed at 1m/s horizontally to the right, while the other three boundaries are static solid walls. We tested our method with Re = 5000, it generates three vortices at the left top, left bottom and right bottom corner, which is consistent with the ground truth shown in [Ghia et al. 1982].

Velocity Divergence. Figure 13 shows the mean divergence of three examples. Even though we do not use the vorticity-streamfunction formulation, the divergence remains near zero.

Fig. 14. Cavity flow

Vorticity Divergence. We demonstrate the effectiveness of bi-directional marching when applied to our method. A significant challenge associated with a traditional Eulerian vortex method is the accumulation of numerical errors over time, leading to a vorticity field that is not divergence-free. As illustrated in Figure 12, adopting the bi-directional march leads to a nearly **80**× decrease in error.

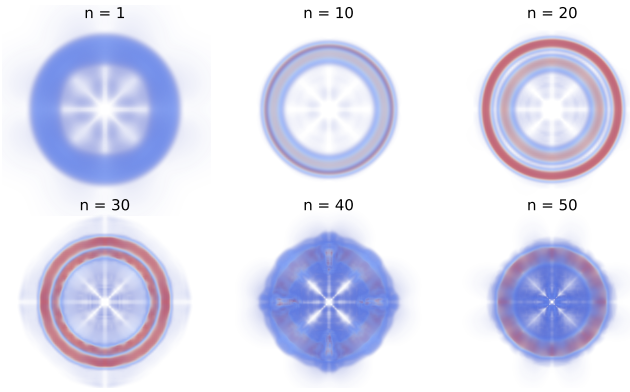

Fig. 15. Reinitialization sensitivity experiment. The 150th frame of the 3D leapfrog with different reinitialization step n is shown.

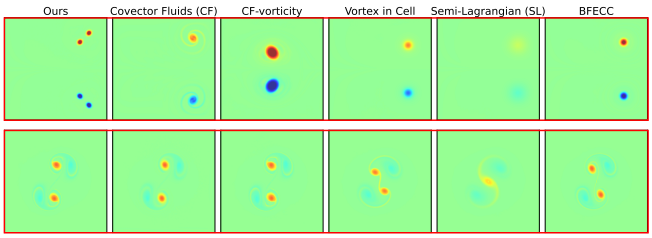

Fig. 16. Comparisons of 2D leapfrog vortices (top) and Taylor vortex (bottom). The columns from left to right depict ours, Covector Fluids (CF) [Nabizadeh et al. 2022], the vorticity version of CF (traditional flow map advection), Vortex-in-Cell, Semi-Lagrangian (SL) advection, Back and Forth Error Compensation and Correction (BFECC) respectively (SL and BFECC only refer to the advection techniques used, so these are still vortex methods. A normal velocity-pressure BFECC comparison is shown in Figure 20).

Fig. 17. For the top row, we compare delta-wing simulated under different viscosity settings. The left one shows a more turbulent appearance compared to others, while the right one creates stable spiral vortex structures along the two leading edges. For the bottom row, we show a static stingray countering an incoming flow. A more chaotic and complex structure of vortices can be observed as fluid viscosity reduces.

6.2 Performance Analysis and Reinitialization Sensitivity

Performance gains of our coupled approach. The performance gains of our coupled approach are provided in Table 2. Both the coupled approach and the separate solve runs in parallel. The coupled system is obtained by stacking the separate system along the x-axis, modified with the compatibility condition. As shown in Table 2, for oblique vortex collision and trefoil knot, the coupled approach

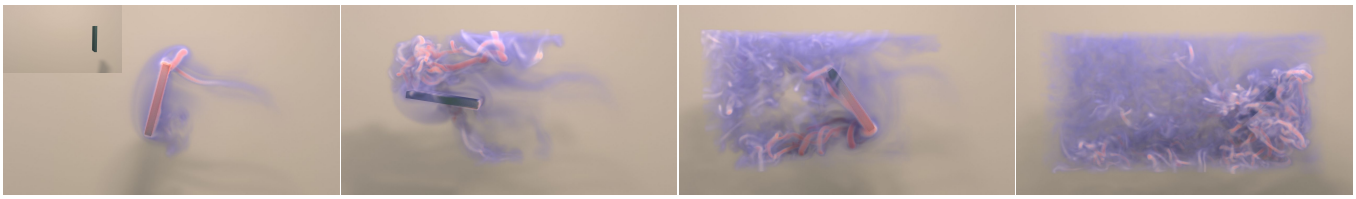

Fig. 18. In this scenario, a paddle moves on a preset path inside a box, which results in vortex shedding and creates turbulent flow, particularly around edges and corners. Numerous vortex filaments and tubes are generated during this process.

Table 2. Computational cost comparison between the coupled and separate solutions. The GPU memory (the second column) is set through Taichi [Hu et al. 2019]. The minimum and the maximum number of iterations of convergence observed during testing are provided. The time column represents the wall-clock time of solving a complete three-dimensional Poisson equation. Note that the 3D Leapfrog experiment could not be executed with 3 GB of GPU memory due to the higher resolution demands.

Experimental Settings		**Coupled Solve**			**Separate Solve**		
Examples	GPU Mem. Provided (GB)	Min Conv. Iters	Max Conv. Iters	Time (sec/solve)	Min Conv. Iters	Max Conv. Iters	Time (sec/solve)
Obique	3	**10**	**10**	**0.15**	15	114	2.40
Obique	6	**10**	**10**	**0.14**	15	114	2.63
Obique	9	**10**	**10**	**0.14**	15	114	1.46
Trefoil	3	**10**	**11**	**0.16**	14	123	1.64
Trefoil	6	**10**	**11**	**0.16**	14	123	2.22
Trefoil	9	**10**	**11**	**0.16**	14	123	1.65
Leapfrog	3	-	-	-	-	-	-
Leapfrog	6	11	11	0.34	**9**	**10**	**0.29**
Leapfrog	9	11	11	0.33	**9**	**10**	**0.28**

Table 3. Average simulation time per frame and peak GPU memory usage. Each frame typically contains 3-10 steps, with each step involving 2 Poisson solves. Our examples usually consist of 300-600 frames. The CFL number are set to be 1.0 for 2D examples and 0.5 for 3D examples.

Name	Time (sec / frame)	GPU Mem. (GB)
2D Leapfrog	1.69	1.54
3D Leapfrog	2.59	5.93
3D Trefoil	1.07	3.60
3D Oblique	0.95	3.56

converges faster and requires less time per Poisson solve, achieving speedups of **10-20**×. For leapfrog, the performance is similar.

Simulation time and memory. In Table 3, we present the performance measurements in terms of the average wall-clock time per frame and the maximum GPU memory for our 2D and 3D examples.

Reinitialization sensitivity. The vorticity field at the 150^{th} frame of the 3D leapfrogging vortices experiment with different reinitialization steps, n, is shown in Figure 15. The numerical dissipation is significant for very small n, such as $n = 1$, leading to poor preservation of vortex structures. Conversely, for much larger n, like $n = 50$, while the ability to preserve vortices improves, the distortion of flow map quantities can introduce artifacts and cause the simulation to become unstable, resulting in a higher number of iterations needed for convergence. As shown in the top left image of Figure 15, the two vortex rings merged at an early stage due to the large numerical dissipation. In contrast, artifacts appear when the reinitialization step is too large, causing distortion in the flow map quantities.

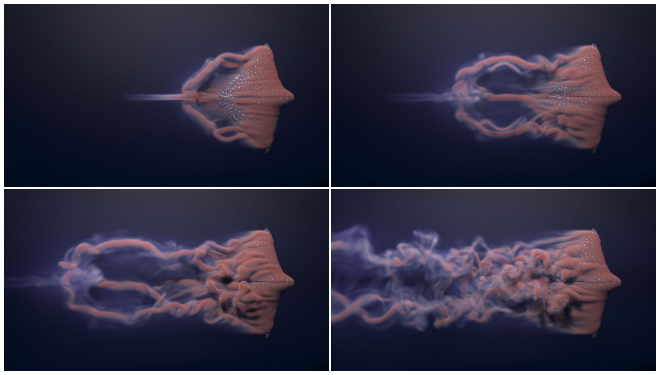

Fig. 19. A stingray passively glides through water at a constant speed with zero viscosity, as simulated using our Eulerian vortex method. The figure illustrates the turbulent vorticity field generated at different time steps.

Fig. 20. Comparison of 3D leapfrog vortices against benchmarks.

6.3 2D Examples

2D leapfrog vortices in a box. In Figure 16 (top), two vortex pairs released from the left engage in leapfrog-like rotation, moving rightward. Nearing the right wall, they split and return along the top and bottom walls. Colliding with the left wall, they reform and repeat this cycle. Ours maintains vortex separation and structure for over 2000 frames, surpassing others that last less than 400 frames.

2D Taylor vortex. We adopt the setup outlined in [McKenzie 2007] and [Qu et al. 2019], where two vortices are positioned adjacently. Setting the distance at 0.81, our method correctly reproduces the vortex separation phenomenon as shown in Figure 16 (bottom).

6.4 3D Examples

Head-on vortex collision. In Figure 5, two opposing vortex rings collide and expand. Smaller vortices form along the periphery, mirroring the experimental phenomena in [Lim and Nickels 1992].

Trefoil knot. Using the initialization file from [Nabizadeh et al. 2022], Figure 2 shows the knot breaking up and reforming into a larger and a smaller ring, which is in line with physical experiment findings in [Kleckner and Irvine 2013].

3D leapfrog vortices. Here, we present the 3D version of the 2D leapfrogging vortices, following the same setup as described in [Deng et al. 2023]. Initially, two vortex rings are placed separately, moving in the same direction. We compare our results with those of BFECC [Kim et al. 2006], CF [Nabizadeh et al. 2022], CF + BiMocq [Nabizadeh et al. 2022; Qu et al. 2019], NFM [Deng et al. 2023], and the streamfunction-based solver [Ando et al. 2015], as shown in Figure 20. The variation using the streamfunction solver differs in the velocity reconstruction step, where the streamfunction solver from [Ando et al. 2015] is applied. To be more specific, the streamfunction is first solved from the vorticity using a Poisson equation, and then the curl of the streamfunction is taken to obtain the velocity. After the fourth leap, both our method and its streamfunction version, as well as NFM [Deng et al. 2023], maintain the separation of the two rings, whereas in the other methods, the rings merge before the third leap. The smoke visualization is presented in Figure 10.

Oblique vortex collision. Two perpendicular vortex rings collide, connect, and deform, with the collision point bulging outward. Later, lateral vortices detach, forming a central ring as shown in Figure 3.

Moving paddle. In Figure 18, a square paddle in a box moves in a kinematic path, inducing velocity difference around its edges and corners, creating a set of vortex filaments and tubes.

Delta wing. A delta wing featuring a swept angle of 75° is put under an airflow with an angle-of-attack to be 20°. The phenomenon of "vortex lift" [Anderson 2010] is observed, i.e., spiral vortex sheets and cores appear along the leading edges, as shown in Figure 1 and 17 (top), which agrees with experimental results in [Délery 2001].

Gliding stingray with viscosity. Figure 19 illustrates a stingray passively gliding, generating vortices. Figure 17 (bottom) displays the stingray with inflow under different viscosity conditions.

Gliding eagle. The eagle is positioned at a 15° angle-of-attack, and generates a rich bundle of vortices from solid-fluid interaction as depicted in the vorticity visualization in Figure 4.

7 Conclusion & Limitation

This paper presents a novel Eulerian vortex method that combines vortex evolution on flow maps with a carefully devised velocity reconstruction scheme to enable the high-quality simulation of intricate vortical structures and turbulent flows. Our key insight lies in that, while good results with impulse have been achieved, vorticity promises to work better with flow maps, as it is more numerically stable and physically interpretable. The efficacy of our method is validated by a number of challenging simulation scenarios.

Our method is subject to a few limitations. First, our method is not able to handle compressible flows. Second, the current system does not support free-surface cases due to nontrivial free-surface boundary treatments required. Extending it to free-surface cases would be a possible and attractive future research direction. Third, while our solver effectively simulates solid-fluid interactions using voxelized boundary models, our system currently does not support boundaries or obstacles with cut cell geometries. Enabling more advanced solid-fluid coupling schemes would be an exciting future direction. Moreover, our revised Poisson system is symmetric (refer to our solver details in the supplementary material to see that) yet lacks proof for positive definiteness, but the solver converges in all our experiments, with the number of iterations of convergence ranging from 8 to 12 for all examples without solids, and remaining below 50 when complex solid boundaries are present. Unfortunately, we are still unable to empirically state that all Poisson systems are positive definite, especially given complex solid boundary conditions. Besides, we tried to model the harmonic component within our solver implicitly, but the problem is not fully solved, as can be seen from the difference between our result and the ground truth in Figure 11. Resolving the issue of the harmonic component in our method so that it can exhibit behavior identical to that achieved by explicitly modeling the harmonic component will be an interesting direction for future research. Finally, while our approach mitigates the issue of constant impulse increase, which allows for possibly sustaining arbitrarily long flow maps without instability, numerical distortions in the flow map evolution currently necessitate reinitialization, which poses a future challenge to further extend the flow map length.

Acknowledgments

We sincerely thank the anonymous reviewers for their valuable feedback. We extend our appreciation to Taiyuan Zhang for his assistance with remote access to computer systems, and to Mengdi Wang for his insightful discussions. Georgia Tech authors acknowledge NSF IIS #2433322, ECCS #2318814, CAREER #2433307, IIS #2106733, OISE #2433313, and CNS #1919647 for funding support. The work is also in part supported by Google. We credit the Houdini education license for video animations.

References

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

John D Anderson. 2010. Aircraft design and performance by Anderson.

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

Ryoichi Ando, Nils Thuerey, , and Chris Wojtan. 2015. A Stream Function Solver for Liquid Simulations. *Transactions on Graphics (SIGGRAPH)* (August 2015), 8.

Alexis Angelidis. 2017. Multi-scale vorticle fluids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Alexis Angelidis and Fabrice Neyret. 2005. Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 87–96.

RD Blevins. 1990. Flow-induced vibration. (1990).

Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 87–95.

Nuttapong Chentanez and Matthias Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. In *ACM Siggraph 2011 Papers*. 1–10.

Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2017. Inside fluids: Clebsch maps for visualization and processing. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.

Albert Chern, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. 2016. Schrödinger's smoke. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–13.

Ricardo Cortez. 1995. *Impulse-based particle methods for fluid flow*. University of California, Berkeley.

Georges-Henri Cottet, Petros D Koumoutsakos, et al. 2000. *Vortex methods: theory and practice*. Vol. 8. Cambridge university press Cambridge.

Fang Da, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2015. Double bubbles sans toil and trouble: Discrete circulation-preserving vortex sheets for soap films and foams. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.

Jean M Délery. 2001. Robert Legendre and Henri Werlé: toward the elucidation of three-dimensional separation. *Annual review of fluid mechanics* 33, 1 (2001), 129–154.

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.

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG)* 26, 1 (2007), 4–es.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 15–22.

Nick Foster and Ronald Fedkiw. 2001. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 23–30.

UKNG Ghia, Kirti N Ghia, and CT Shin. 1982. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of computational physics* 48, 3 (1982), 387–411.

Leslie Greengard and Vladimir Rokhlin. 1987. A fast algorithm for particle simulations. *Journal of computational physics* 73, 2 (1987), 325–348.

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

Anil Nirmal Hirani. 2003. *Discrete exterior calculus*. California Institute of Technology.

Ben Houston, Michael B Nielsen, Christopher Batty, Ola Nilsson, and Ken Museth. 2006. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Transactions on Graphics (TOG)* 25, 1 (2006), 151–175.

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), 1–16.

Huaxiong Huang and Ming Li. 1997. Finite-difference approximation for the velocity-vorticity formulation on staggered and non-staggered grids. *Computers & fluids* 26, 1 (1997), 59–82.

Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *ACM SIGGRAPH 2006 Papers*. 805–811.

Sadashige Ishida, Chris Wojtan, and Albert Chern. 2022. Hidden degrees of freedom in implicit vortex filaments. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14.

Antony Jameson, Wolfgang Schmidt, and Eli Turkel. 1981. Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In *14th fluid and plasma dynamics conference*. 1259.

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. 2005. FlowFixer: Using BFECC for Fluid Simulation.. In *NPH*. 51–56.

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. 2006. Advections with significantly reduced dissipation and diffusion. *IEEE transactions on visualization and computer graphics* 13, 1 (2006), 135–144.

Dustin Kleckner and William TM Irvine. 2013. Creation and dynamics of knotted vortices. *Nature physics* 9, 4 (2013), 253–258.

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 Papers*. 1–10.

TT Lim and TB Nickels. 1992. Instability and reconnection in the head-on collision of two vortex rings. *Nature* 357, 6375 (1992), 225–227.

Chung Ho Liu. 2001. Numerical solution of three-dimensional Navier–Stokes equations by a velocity–vorticity method. *International journal for numerical methods in fluids* 35, 5 (2001), 533–557.

Frank Losasso, Ronald Fedkiw, and Stanley Osher. 2006. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids* 35, 10 (2006), 995–1010.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *Acm siggraph 2004 papers*. 457–462.

Aleka McAdams, Eftychios Sifakis, and Joseph Teran. 2010. A Parallel Multigrid Poisson Solver for Fluids Simulation on Large Grids.. In *Symposium on Computer Animation*, Vol. 65. 74.

Alexander George McKenzie. 2007. *HOLA: a high-order Lie advection of discrete differential forms with applications in Fluid Dynamics*. Ph. D. Dissertation. California Institute of Technology.

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.

Chloé Mimeau and Iraj Mortazavi. 2021. A review of vortex methods and their applications: From creation to recent advances. *Fluids* 6, 2 (2021), 68.

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

Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. 2002. Physically based modeling and animation of fire. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 721–728.

Marcel Padilla, Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2019. On bubble rings and ink chandeliers. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.

Sang Il Park and Myoung Jun Kim. 2005. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 261–270.

Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–8.

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.

Nick Rasmussen, Douglas Enright, Duc Nguyen, Sebastian Marino, Nigel Sumner, Willi Geiger, Samir Hoon, and Ronald Fedkiw. 2004. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 193–202.

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.

John Stanley Sawyer. 1963. A semi-Lagrangian method of solving the vorticity advection equation. *Tellus* 15, 4 (1963), 336–342.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35 (2008), 350–371.

Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH 2005 Papers*. 910–914.

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

J Stam. 1999. Stable fluids. *Proc. of SIGGRAPH 1999: ACM SIGGRAPH 1999* (1999), 121–128.

David M Summers and Alexandre J Chorin. 1996. Hybrid vortex/magnet methods for flow over a solid boundary. In *ESAIM: Proceedings*, Vol. 1. EDP Sciences, 65–76.

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.*

Clifford Truesdell. 2018. *The kinematics of vorticity*. Courier Dover Publications.

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers*. 1–12.

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.
Jie-Zhi Wu, Hui-Yang Ma, and M-D Zhou. 2007. *Vorticity and vortex dynamics*. Springer Science & Business Media.
Zuo-Bing Wu. 2003. Streamline topology and dilute particle dynamics in a Karman vortex street flow. *International Journal of Bifurcation and Chaos* 13, 05 (2003), 1275–1286.
Shiying Xiong, Rui Tao, Yaorui Zhang, Fan Feng, and Bo Zhu. 2021. Incompressible flow simulation on vortex segment clouds. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–12.
Shiying Xiong, Zhecheng Wang, Mengdi Wang, and Bo Zhu. 2022. A clebsch method for free-surface vortical flow simulation. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.
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.
Hang Yin, Mohammad Sina Nabizadeh, Baichuan Wu, Stephanie Wang, and Albert Chern. 2023a. Fluid Cohomology. *ACM Trans. Graph.* 42, 4, Article 126 (jul 2023), 25 pages. https://doi.org/10.1145/3592402
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. 2023b. A Characteristic Mapping Method for the three-dimensional incompressible Euler equations. *J. Comput. Phys.* (2023), 111876.
Xinxin Zhang and Robert Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–11.
Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–8.
Junwei Zhou, Duowen Chen, Molin Deng, Yitong Deng, Yuchen Sun, Sinan Wang, Shiying Xiong, and Bo Zhu. 2024. Eulerian-Lagrangian Fluid Simulation on Particle Flow Maps. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–20.
Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. 2013. A new grid structure for domain extension. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.