Covector Fluids

MOHAMMAD SINA NABIZADEH, University of California, San Diego
STEPHANIE WANG, University of California, San Diego
RAVI RAMAMOORTHI, University of California, San Diego
ALBERT CHERN, University of California, San Diego

The animation of delicate vortical structures of gas and liquids has been of great interest in computer graphics. However, common velocity-based fluid solvers can damp the vortical flow, while vorticity-based fluid solvers suffer from performance drawbacks. We propose a new velocity-based fluid solver derived from a reformulated Euler equation using covectors. Our method generates rich vortex dynamics by an advection process that respects the *Kelvin circulation theorem*. The numerical algorithm requires only a small local adjustment to existing advection-projection methods and can easily leverage recent advances therein. The resulting solver emulates a vortex method without the expensive conversion between vortical variables and velocities. We demonstrate that our method preserves vorticity in both vortex filament dynamics and turbulent flows significantly better than previous methods, while also improving preservation of energy.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Physical simulation**; • **Mathematics of computing** → **Partial differential equations**; *Differential calculus*; *Discretization*.

Additional Key Words and Phrases: Fluid dynamics, differential forms, Lie derivatives, Kelvin circulation theorem

ACM Reference Format:
Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector Fluids. *ACM Trans. Graph.* 41, 4, Article 113 (July 2022), 16 pages. https://doi.org/10.1145/3528223.3530120

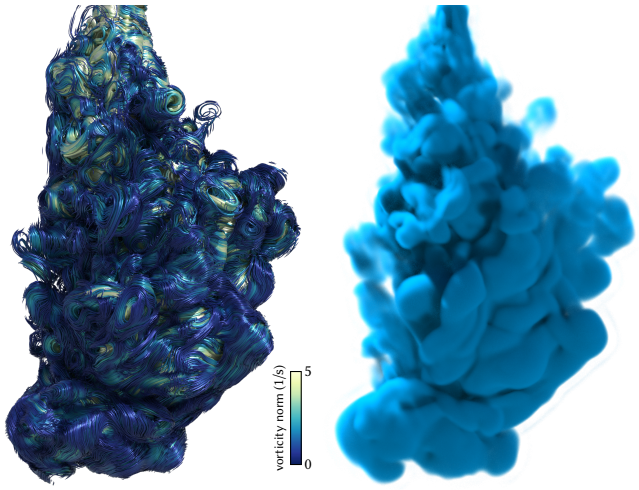


Fig. 1. Ink jet created after high initial velocity injection of ink into the domain under gravity. Intricate vortex nucleation is made possible with our Covector Fluids (CF) method. Left: Vorticity field visualized. Right: Ink jet visualized using a high density field.

1 INTRODUCTION

A realistic animation of smoke or ink with intricate vortex phenomena (Fig. 1 and Fig. 2) requires simulating an incompressible fluid with low viscosity. The *incompressible Euler equation*, a.k.a. the *inviscid Navier–Stokes equation*, governs the time-evolution of the velocity vector field $\mathbf{u}\colon M \subset \mathbb{R}^n \to \mathbb{R}^n$ ($n = 2$ or 3) of such a fluid with domain M:

$$\frac{\partial}{\partial t}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p, \quad \nabla\cdot\mathbf{u} = 0. \tag{1}$$

Here, $p\colon M \to \mathbb{R}$ is the kinematic pressure of the fluid.

The majority of fluid solvers are based on *splitting* (1) into an advection step and a projection step:

1: Set the flow velocity $\mathbf{v} \leftarrow \mathbf{u}$; ▷ freeze flow velocity
2: Transport $\mathbf{u}$ by $(\frac{\partial}{\partial t} + \mathbf{v}\cdot\nabla)\mathbf{u} = \mathbf{0}$ for Δt time; ▷ advection
3: $\mathbf{u} \leftarrow \mathbf{u} - \Delta t\nabla p$ so that $\nabla\cdot\mathbf{u} = 0$ afterwards. ▷ projection

Authors' addresses: Mohammad Sina Nabizadeh, University of California, San Diego, mnabizad@ucsd.edu; Stephanie Wang, University of California, San Diego, stw006@eng.ucsd.edu; Ravi Ramamoorthi, University of California, San Diego, ravir@cs.ucsd.edu; Albert Chern, University of California, San Diego, alchern@eng.ucsd.edu.

0730-0301/2022/7-ART113
https://doi.org/10.1145/3528223.3530120

In practice, the advection step (Step 2) is achieved by generating an inverse flow map $\Psi\colon M \to M$ with the flow velocity $\mathbf{v}$ for a time span of Δt and looking up the velocity field (Fig. 3a)

$$\mathbf{u}(\mathbf{x}) \leftarrow \mathbf{u}(\Psi(\mathbf{x})), \quad \mathbf{x} \in M. \tag{2}$$

The update of the variable $\mathbf{u}$ keeps the velocity invariant on each particle moving with the flow during the advection step.

A well-known problem of the splitting method is that it tends to lose vorticity. As well illustrated by [Zhang et al. 2015] (see also Fig. 3b), the transportation (2) rearranges the rotation component of $\mathbf{u}$ to the divergent component, which is then removed by the projection step. Hence, fluid solvers that aim at reproducing energetic vortices must include some intervention in (2) that restores the missing vorticity. Here, we name a few examples. The *vorticity confinement method* modifies the flow velocity $\mathbf{v}$ with a centripetal force to maintain vortex concentration [Fedkiw et al. 2001]. The *energy preserving method* [Mullen et al. 2009], or the more recent *advection-reflection method* [Zehnder et al. 2018], adds twice the pressure at the halfway point of the advection rather than the end of it. More direct methods restore the vorticity by simulating the *vorticity equation* [Elcott et al. 2007; Zhang et al. 2015]. However, the conversion between the vorticity and velocity variables requires an expensive global integration.

In this paper, we propose a simple modification of (2):

$$\mathbf{u}(\mathbf{x}) \leftarrow (d\Psi(\mathbf{x}))^\intercal\,\mathbf{u}(\Psi(\mathbf{x})), \quad \mathbf{x} \in M, \tag{3}$$

(a) MC+R [Zehnder et al. 2018] (b) BiMocq [Qu et al. 2019] (c) CF (Covector Fluids)+MCM (Ours)

Fig. 2. A bunny meteor falling. Smoke is generated from the surface of a bunny obstacle against a laminar flow with no other external force. Our method is capable of shedding many more vortices from the surface of the obstacle. This results in a more detailed and heavier smoke cloud trailing the bunny.

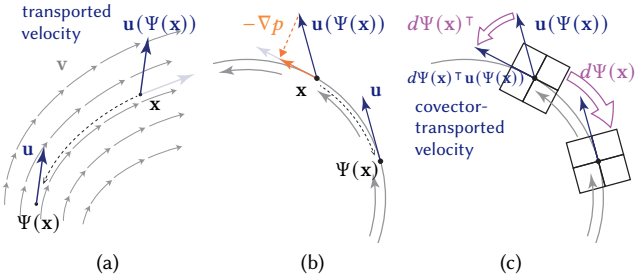


Fig. 3. (a) In a standard fluid solver, the velocity $\mathbf{u}$ is transported by the flow $\mathbf{v}$ using an inverse flow map Ψ. (b) This process can turn a rotation motion into a divergent one which is subsequently damped by the pressure projection. (c) Our advection method maintains the vorticity by a multiplication with the transposed Jacobian of the inverse flow map.

where $(d\Psi)^\intercal$ is the transposed Jacobian of the inverse flow map. See Fig. 3c. We show that (3) effectively *simulates the correct vorticity equation even in the absence of pressure* (Section 4.4). Our method splits the incompressible Euler equation while respecting vorticity conservation, and requires no further vorticity restoration or rescheduling of the addition of pressure. The modification (3) only involves a *local* calculation, as opposed to other vortex methods that require a *global* integration from vorticity to velocity. Fig. 1, Fig. 2, and Fig. 4 showcase our method producing rich vortex structures.

The modified velocity update (3) transports the velocity field $\mathbf{u}$ so that its line integrals along any curve C carried by the fluid are preserved:

$$\int_C \underbrace{(d\Psi(\mathbf{x}))^\intercal \mathbf{u}(\Psi(\mathbf{x}))}_{\text{updated } \mathbf{u} \text{ by (3)}} \cdot d\mathbf{l} = \int_C \mathbf{u}(\Psi(\mathbf{x})) \cdot d\Psi(\mathbf{x}) d\mathbf{l} = \int_{\Psi(C)} \mathbf{u} \cdot d\mathbf{l}. \quad (4)$$

In particular, this transportation ensures that the circulation $\oint_C \mathbf{u} \cdot d\mathbf{l}$ around every closed loop is conserved, which is the integral form of the vorticity equation.

Equation (3) is a *Lie advection* of $\mathbf{u}$ *treated as a covector field*. This transportation differs from the direct componentwise advection (2) as they depict different conservation laws. The traditional componentwise advection (2) follows the *conservation of linear momentum*, whereas the covector Lie advection follows *Kelvin's conservation of circulation* [Thomson 1868; Frisch and Villone 2014]. For simulating vortex-dominant phenomena, computations based on the conservation of circulation [Elcott et al. 2007] are more advantageous.

The idea of using Lie advections in fluid simulations dates back to the work of *impulse methods* [Oseledets 1989; Buttke 1993; Cortez 1995] and the development of *discrete exterior calculus* [Grinspun et al. 2006; Elcott et al. 2007; McKenzie 2007]. However, these methods have mainly been studied either as smoothed particle methods [Cortez 1995] or vortex methods [Elcott et al. 2007], which require computational setups different from the common grid-based velocity solver. Hence, there has been a lack of integration or discussion of Lie advections in the mainstream fluid simulation paradigm. Our new treatment (3) makes access to Lie advection exceedingly simple in standard solvers. With this unification, we have access both to accurate vortex dynamics from Lie advections and the detail-preserving quality of the recent methods (Fig. 2).

Contributions. The proposed approach brings new machinery to fluid simulations:

- *Velocity-based vortex method:* Because our method preserves circulation, it emulates a *vortex method* (Section 4.4). However, the computation of this new vortex method stays entirely at the velocity level and within the advection-projection computational paradigm. This alleviates the known tradeoffs in vortex methods, such as the cost of velocity reconstruction from vorticity and the stability problem of vortex stretching. Compared to previous velocity-based methods, our method preserves more details in a vortex-dominant flow (Fig. 2).
- *Energy-preserving vortex method:* Energy preservation has been challenging for previous vortex methods. We demonstrate that while our method is equivalent to a vortex method, it preserves energy comparable to the advection-reflection method [Zehnder et al. 2018] (Fig. 8 and Fig. 9).
- *New Lie advection scheme:* We introduce a simple Lie advection integrator (Alg. 4) for covector fields using the technique of

Table 1. Method acronyms used throughout the paper.

Method	Reference	Acronym
Stable fluids	[Stam 1999]	SF
Back-and-forth error compensation and correction	[Dupont and Liu 2003]	BFECC
Stable and circulation preserving fluids	[Elcott et al. 2007]	SCPF
MacCormack	[Selle et al. 2008]	MC
Integrated vorticity of convective kinematics	[Zhang et al. 2015]	IVOCK
Reflection	[Zehnder et al. 2018]	R
Method of characteristic mapping	[Sato et al. 2018]	MCM
Bi-directional mapping of convective quantities	[Qu et al. 2019]	BiMocq
Covector fluids	Our method	CF

back-and-forth error correction and compensation (BFECC). This is needed to reduce the numerical dissipation from which semi-Lagrangian methods suffer [Elcott et al. 2007] (Fig. 5).

- *Method of characteristic mapping with covectors:* We demonstrate that the covector fields can be pulled back using a Lagrangian marker (Section 4.6, Section 5.3). We therefore combine our method with the BiMocq scheme [Qu et al. 2019] into a *method of characteristic mapping* that preserves more spatial details by reducing the amount of interpolation (Alg. 5).

2 RELATED WORK

Our method stays close to the common computational framework for physics-based smoke simulation in computer graphics introduced by [Foster and Metaxas 1997] and [Stam 1999] (*cf.* the algorithm below (1)). Foster and Metaxas [1997] employ the staggered *Marker-and-Cell* (MAC) grid [Harlow and Welch 1965] and the *projection method* [Chorin 1968; Temam 1969] from computational fluid dynamics (CFD). Stam's *Stable Fluids* [1999] uses the *semi-Lagrangian scheme* for the advection step, whose stability makes fluid simulation highly practical for graphics applications. Prior to graphics, the semi-Lagrangian scheme was primarily developed in meteorology [Sawyer 1963; Staniforth and Côté 1991] and the finite element flow analysis community [Douglas and Russell 1982].

A caveat in the vanilla Stable Fluids method (SF) is that it comes with *numerical viscosity*. Such dissipation makes simulating inviscid fluid phenomena such as vortex dynamics challenging, driving active research interest in solving this problem. See Table 1 for a list of relevant methods.

Non-dissipative advections. The lower order of accuracy and exceeding amount of interpolation in the semi-Lagrangian method cause significant numerical dissipation. This problem can be reduced by methods of higher-order accuracy based on (W)ENO interpolations [Losasso et al. 2006]. Kim *et al.* [2005] and Selle *et al.* [2008] apply a simple back-and-forth error compensation and correction (BFECC) [Dupont and Liu 2003] or a modified MacCormack method (MC) to bootstrap a 1st order semi-Lagrangian advection to a 2nd order method. Qu *et al.* [2019] employ the *dual mesh characteristic* (DMC) method [Cho et al. 2018], also known as a *non-interpolating semi-Lagrangian method* [Rančić and Sindjić 1989]. The method of characteristic mapping (MCM) [Tessendorf and Pelfrey 2011; Sato et al. 2018; Qu et al. 2019] keeps track of a full Eulerian-to-Lagrangian map to evaluate the transported quantities with fewer interpolations. Other full Lagrangian methods use particles (FLIP, APIC, PolyPIC) to carry out the advection with a general goal of reducing the interpolation cost when transferring fields between particles and the grid [Zhu and Bridson 2005; Jiang et al. 2015; Fu et al. 2017]. *Kinetic models* or the *lattice Boltzmann methods* (LBM) [Li et al. 2018, 2020; Lyu et al. 2021] represent the statistics of moving particles on a grid, which are more immune to interpolations as distributions in different velocity directions do not automatically blend like in a macroscopic model.

However, most previous work on non-dissipative schemes only tackles the traditional advection equation for scalar or componentwise vector fields. Except for [McKenzie 2007; Mullen et al. 2011] featuring WENO interpolations, few authors have explored non-dissipative methods for *Lie advection equations* for covectors and other differential forms. Our work introduces a simple non-dissipative scheme for Lie advections by using the BFECC technique.

Reduction of splitting error. Besides the dissipation from the advection solver, the source of vorticity loss comes from the splitting error between the advection and projection steps (Fig. 3b). To counteract the loss of vorticity, Fedkiw *et al.* [2001] employs a *vorticity confinement* [Steinhoff and Underhill 1994] which adds an artificial centripetal force to confine the vorticity. The *energy preserving fluid* by Mullen *et al.* [2009] reschedules the addition of pressure to the halfway point of the advection rather than the end of advection. However, the energy preserving fluid is an implicit method that requires an expensive nonlinear Newton solve at every time step. A more efficient variant of this energy-preserving scheme is the *advection-reflection method* [Zehnder et al. 2018], which can be viewed as an application of *Strang's splitting* [Strang 1968; LeVeque 2002, Sect. 17.4]. The reflection method is also included in the BiMocq method [Qu et al. 2019].

Our method removes the splitting error by modifying the advection into a covector advection. The covector advection commutes with the pressure projection (Section 4.5).

Vortex methods. The collection of vortex methods is a major line of work that aims at simulating vortical phenomena. In a vortex method, one constructs a representation of the fluid vorticity and advances it with the vorticity equation. Such representations include vortex particles [Selle et al. 2005; Park and Kim 2005; Zhang and Bridson 2014; Angelidis 2017], filaments [Cottet et al. 2000; Angelidis and Neyret 2005; Weißmann and Pinkall 2009, 2010; Padilla et al. 2019], segments [Chorin 1990; Xiong et al. 2021], sheets [Brochu et al. 2012; Pfaff et al. 2012], volumes [Elcott et al. 2007; Zhang et al. 2015], spectral elements [De Witt et al. 2012; Liu et al. 2015; Cui et al. 2018], and Clebsch level sets [Chern et al. 2016, 2017; Yang et al. 2021]. While the vorticity equation is a straightforward scalar transport equation in 2D [Yaeger et al. 1986; Chiba et al. 1994; Azencot et al. 2014], the 3D counterpart has a *vortex stretching* term which can cause numerical instability. This stretching instability is most severe in 3D vortex particle methods, which therefore require artificial clamping or diffusion sacrificing energy conservation. The stretching problem is reduced with filaments, segments and sheets, but at a cost of sophisticated vortex reconnection or re-meshing. The Lie advection based vortex volume [Elcott et al. 2007] can maintain a conditional stability, but suffers from loss of energy due to the semi-Lagrangian scheme. Another caveat of vortex methods is the demand for a stream function solver, a Biot–Savart integrator, or a

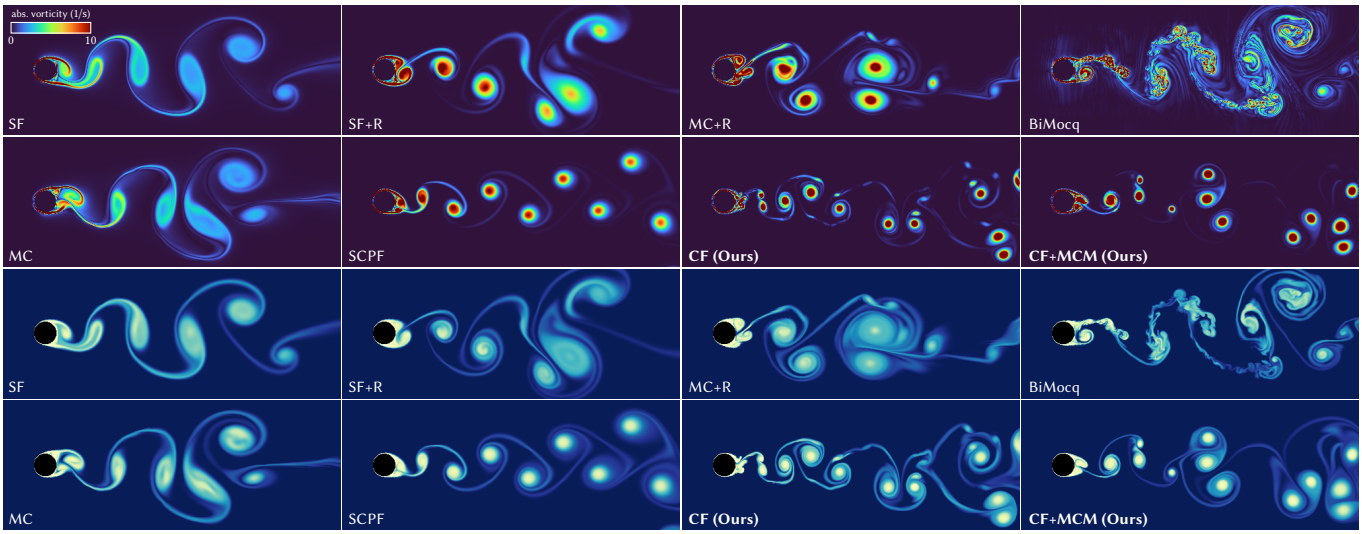


Fig. 4. Von Kármán vortex street simulated for different methods. Our method results in increasingly more vortex nucleation close to the obstacle compared to other methods (see Table 1 for method acronyms). The additional vortex nucleation allows our method to achieve more vortical structures throughout the simulation (see video 4:41).

velocity-to-Clebsch variable converter [Chern et al. 2017]. These global integrators are more expensive than the pressure solver in the velocity-based methods.

Our method simulates the vorticity equation only using the velocity variables (Section 4.4) and hence bypasses the cost of variable conversions. Under a mild CFL condition similar to [Elcott et al. 2007], we do not observe vortex stretching instabilities, and therefore do not require additional treatments that may sacrifice conservation of energy.

Covector formulations. Over the years, covector formulations have been re-introduced many times. Covectors and differential forms were commonly used in the early studies of the incompressible Euler equations [Lagrange 1788]. Such formalism has led to several discoveries of conservation laws in the Lagrangian coordinates [Frisch and Villone 2014] for several quantities including vorticity [Cauchy 1815; Hankel 1861], velocity covector fields [Weber 1868], and their line integrals along closed and open curves [Thomson 1868]. Unfortunately, covectors began to attract less attention during the development of vector calculus in the later half of the 19th century. The covector formulation resurfaced in Geometric Hydrodynamics [Arnold 1966; Marsden and Weinstein 1983; Arnold and Khesin 1998] due to its natural role in a fluid's *particle relabeling symmetry*. Oseledets [1989] also introduces this formulation in the context of Hamiltonian formulation of fluid dynamics [Clebsch 1859; Holm et al. 1983; Morrison 1998; Pavlov et al. 2011; Mumford and Michor 2012; Chern 2017].

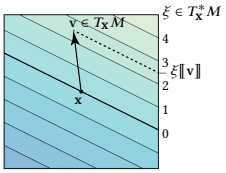
Several researchers adopted Oseledets' covector formulation into the *impulse methods* [Buttke 1993; Cortez 1995; E and Liu 1997; Russo and Smereka 1999; Feng et al. 2022]. The impulse methods were approached using smoothed particles [Buttke 1993] and finite differencing [E and Liu 1997]. However, there is limited research on extending these impulse methods into semi-Lagrangian schemes or methods of characteristics. For example, the recent attempts of impulse method [Feng et al. 2022] split the "stretching" term in the Lie advection away from the semi-Lagrangian advection, failing to be a true characteristic mapping. More recently, the impulse method has evolved into the *gauge method*, whose focus has shifted to the commutativity between the impulse advection and the addition of pressure (Section 4.5) for interfacial treatments [Saye 2016; Yang et al. 2021]. Beyond these lines of work, relatively few papers have explored covector fluids computationally. To our knowledge, we are the first to introduce a simple and general treatment (3) to obtain covector fluid simulation that can be integrated into mainstream fluid solvers based on semi-Lagrangian schemes and methods of characteristic mappings.

3 PRELIMINARIES

In this section, we cover the necessary background to work with covectors. Readers can find similar coverage for covectors in [Hirani 2003; Grinspun et al. 2006; Needham 2021]. Those readers who are familiar with exterior calculus may skip ahead to Section 4.

3.1 Covectors

We use M to denote our fluid domain: it can be an open and connected region in $\mathbb{R}^n$ with $n = 2, 3$.[1] At each point $\mathbf{x} \in M$, the *tangent space* $T_\mathbf{x}M$ is the space of tangent vectors based at $\mathbf{x}$. For $M \subset \mathbb{R}^n$, $T_\mathbf{x}M \cong \mathbb{R}^n$ using the Cartesian coordinates. The dual space of $T_\mathbf{x}M$, the *cotangent space* $T^*_\mathbf{x}M$



[1] Although we focus on Euclidean spaces $\mathbb{R}^n$ in this paper, the theory covered in Section 3.1 applies to general Riemannian manifolds.

$= (T_{\mathbf{x}}M \xrightarrow{\text{linear}} \mathbb{R})$, represents the collection of all linear scalar functions on $T_{\mathbf{x}}M$. Each element of $T^*_{\mathbf{x}}M$ is called a *covector* based at $\mathbf{x}$. The evaluation, a.k.a. the *dual pairing*, of a covector $\xi \in T^*_{\mathbf{x}}M$ and a vector $\mathbf{v} \in T_{\mathbf{x}}M$ is denoted by $\xi[\![\mathbf{v}]\!]$ (see inset). A *covector field* is an assignment of covector $\xi(\mathbf{x})$ at each point $\mathbf{x} \in M$. We let $\mathfrak{X}(M)$ and $\mathfrak{X}^*(M)$ denote the space of all vector fields and covector fields on M respectively.[2] A covector field $\xi \in \mathfrak{X}^*(M)$ is also understood as a *differential 1-form*, which is an object that can be integrated along an oriented curve C as $\int_C \xi[\![d\mathbf{l}]\!]$, or $\int_C \xi$ for short. Here, $d\mathbf{l}$ represents an infinitesimal directed element of curve C.

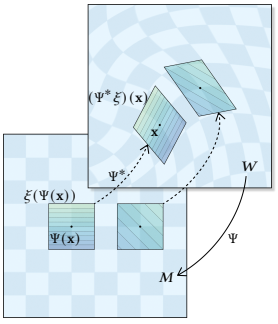
3.2 Musical isomorphisms

When $T_{\mathbf{x}}M$ is equipped with an inner product $\langle\cdot,\cdot\rangle$, each vector $\mathbf{a} \in T_{\mathbf{x}}M$ is uniquely associated with a covector $\mathbf{a}^\flat \in T^*_{\mathbf{x}}M$ so that $\mathbf{a}^\flat[\![\cdot]\!] = \langle \mathbf{a}, \cdot\rangle$. Mirroring what $\flat$ (flat) and $\sharp$ (sharp) entail in the musical context, they are the mathematical inverse of each other: $\flat$ turns a vector $\mathbf{a} \in T_{\mathbf{x}}M$ into a covector $\mathbf{a}^\flat$, so $\sharp$ turns a covector $\xi \in T^*_{\mathbf{x}}M$ into a vector $\xi^\sharp$. In this paper, we will be using the Euclidean $\mathbb{R}^n$ inner product for our musical isomorphisms.

3.3 Differential of a function

The differential df of a function $f\colon M \to \mathbb{R}$ is a covector field defined so that $(df)[\![\mathbf{v}]\!]$ is the directional derivative of f in the direction $\mathbf{v}$. In fact, when there is an inner product structure, we have $(\operatorname{grad} f) = (df)^\sharp$ or that $\langle \operatorname{grad} f, \mathbf{v}\rangle = df[\![\mathbf{v}]\!] = \langle df^\sharp, \mathbf{v}\rangle$ for all directions $\mathbf{v}$. The differentials of functions form an important subclass of covector fields. Note that not all covector fields are the differential of a function.

3.4 Pullback operator



Pullback is the exterior calculus version of *change of variables*. In the context of fluids, this encompasses the communication between the Lagrangian and the Eulerian coordinates. While a change of variables for a function is simply a function composition, a change of variables for a covector field requires the chain rule. Suppose there are two domains M and W and a map $\Psi\colon W \to M$. The pullback of a scalar function $g\colon M \to \mathbb{R}$ by Ψ becomes a function on W given by

$$\Psi^* g\colon W \to \mathbb{R}, \quad (\Psi^* g)(\mathbf{x}) = g(\Psi(\mathbf{x})), \quad \mathbf{x} \in W. \tag{5}$$

Now, the pullback of a covector field $\xi \in \mathfrak{X}^*(M)$ by Ψ is a covector field $\Psi^*\xi \in \mathfrak{X}^*(W)$ defined so that

$$(\Psi^*\xi)(\mathbf{x})[\![\mathbf{v}]\!] = \xi(\Psi(\mathbf{x}))[\![d\Psi(\mathbf{x})\mathbf{v}]\!] \quad \text{for all } \mathbf{v} \in T_{\mathbf{x}}W, \mathbf{x} \in W. \tag{6}$$

Here, $d\Psi(\mathbf{x})\colon T_{\mathbf{x}}W \xrightarrow{\text{linear}} T_{\Psi(\mathbf{x})}M$ is the Jacobian matrix of Ψ at $\mathbf{x}$ (see inset). This definition ensures that for exact differentials

$$\Psi^*(dg) = d(\Psi^* g) \quad \text{for } g\colon M \to \mathbb{R}, \tag{7}$$

and that under the integration sign

$$\int_C \Psi^*\xi = \int_{\Psi(C)} \xi \tag{8}$$

for each oriented curve C in W. The right-hand side of (6) can further be expressed in terms of the adjoint (matrix transpose) $d\Psi^\intercal(\mathbf{x})\colon T^*_{\Psi(\mathbf{x})}M \xrightarrow{\text{linear}} T^*_{\mathbf{x}}W$ as follows:

$$\xi(\Psi(\mathbf{x}))[\![d\Psi(\mathbf{x})\mathbf{v}]\!] = \Big(d\Psi^\intercal(\mathbf{x})\xi(\Psi(\mathbf{x}))\Big)[\![\mathbf{v}]\!]. \tag{9}$$

In sum, the pullback of a covector by Ψ involves a composition and a matrix multiplication by the Jacobian transpose:

$$\boxed{(\Psi^*\xi)(\mathbf{x}) = d\Psi^\intercal(\mathbf{x})\xi(\Psi(\mathbf{x})).} \tag{10}$$

3.5 Lie Derivative

In continuum mechanics, the map between the Lagrangian and Eulerian coordinates evolves over time as the continuum flows. The associated change of variables, *i.e.* pullback, also changes over time as a result. The *Lie derivative* is the generalization of the directional derivative that measures the rate of change of a time-varying pullback field.

Suppose we have a one-parameter family of maps $\Phi_t\colon M \to W$, $t \in \mathbb{R}$. At each instance t, the time-derivative $\dot\Phi_t$ represents the velocity of the flow. Each scalar field $g\colon W \to \mathbb{R}$ and covector field $\xi \in \mathfrak{X}^*(W)$ renders a one-parameter family of pullback scalar fields $\Phi_t^* g$ and pullback covector fields $\Phi_t^*\xi$ on M. In the more familiar case of scalar functions, the rate of change of the pullback scalar field is equivalent to a directional derivative of g along the vector field $\dot\Phi_t = \frac{\partial \Phi}{\partial t}$ on W:

$$\frac{\partial}{\partial t}(\Phi_t^* g)(\mathbf{x}) = \frac{\partial}{\partial t}(g(\Phi_t(\mathbf{x}))) = \left(\dot\Phi_t \cdot \nabla g\right)(\Phi_t(\mathbf{x})). \tag{11}$$

In terms of pullbacks, $\frac{\partial}{\partial t}(\Phi_t^* g) = \Phi_t^*\left((\dot\Phi_t \cdot \nabla) g\right)$. We define the quantity $(\dot\Phi_t \cdot \nabla) g$ as the *Lie derivative* for the scalar field g.

For a covector field $\xi \in \mathfrak{X}^*(W)$, the *Lie derivative* $\mathscr{L}_{\dot\Phi_t}\xi$ along the vector field $\dot\Phi_t$ is defined in the same manner by the equation

$$\frac{\partial}{\partial t}(\Phi_t^*\xi) = \Phi_t^*\left(\mathscr{L}_{\dot\Phi_t}\xi\right). \tag{12}$$

Integrating both sides on the image $\Phi_t(C) \subset W$ of a curve $C \subset M$,

$$\frac{d}{dt}\int_{\Phi_t(C)} \xi = \int_{\Phi_t(C)} \mathscr{L}_{\dot\Phi}\,\xi. \tag{13}$$

The above equation shows that the Lie derivative expresses the result of differentiating an integral with a varying integration domain $\Phi_t(C)$.

In terms of vector calculus, the Lie derivative of the covector field $\xi = \mathbf{a}^\flat$ along the vector field $\mathbf{v} = \dot\Phi$ is given by[3](Appendix A.1)

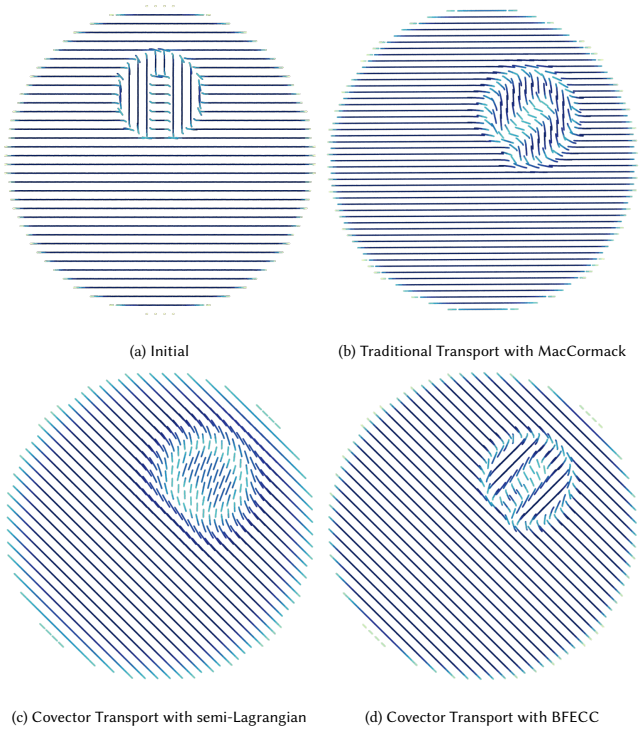
$$(\mathscr{L}_{\mathbf{v}}\,\xi)^\sharp = \mathbf{v}\cdot\nabla\mathbf{a} + (\nabla\mathbf{v})\cdot\mathbf{a}. \tag{14}$$

4 THEORY

We use this section to elucidate the mathematical foundation of the proposed method. The key is to rewrite the incompressible Euler equation in terms of the *velocity covector field* and its Lie derivative. We explain why the covector formulation is more resilient to time splitting. We also demonstrate that a simulation algorithm based on

[2] A common notation is $\mathfrak{X}(M) = \Gamma(TM)$ and $\mathfrak{X}^*(M) = \Omega^1(M)$.

[3] In index notation, $(\mathbf{v}\cdot\nabla\mathbf{a} + (\nabla\mathbf{v})\cdot\mathbf{a})_i = v^j\partial_j a_i + a_j\partial_i v^j$.



(a) Initial (b) Traditional Transport with MacCormack

(c) Covector Transport with semi-Lagrangian (d) Covector Transport with BFECC

Fig. 5. (a) This covector field, visualized as a level set, is initialized differently inside and outside of the Zalesak's disk. (b) Component-wise advection of covector fields fails to correctly transport the field, resulting in a wrong orientation of the covector field. (c) The field is correctly transported using a covector advection scheme. Despite correct transportation of the covector field, the field is heavily dissipated due to sampling error. (d) Using BFECC, the field is accurately transported through time. This results in a field which matches the initial orientation while being rotated with the flow.

the covector formulation is equivalent to a vortex method computed at the velocity level.

4.1 Transportation of covector fields

One fundamental building block of a fluid solver is the *linear advection equation* or the *transport equation*. Traditionally written using the *material derivative* $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla$, the transport equation $\frac{D}{Dt} X = 0$ describes a quantity X being transported along a flow with a given velocity $\mathbf{v}$. However, this notion of material derivative only applies to scalar fields and fails to capture the transportation of covector fields. Here, we discuss a generalized transport equation that incorporates covector fields. This transportation of covector fields will be a fundamental building block of our Covector Fluids method.

Suppose we have a known time-independent vector field $\mathbf{v} \in \mathfrak{X}(M)$. In the following, we use M_t instead of M to provide a visual cue of the space at different time stamps. The vector field $\mathbf{v}$ generates a (forward) flow map $\Phi_t : M_0 \to M_t$ by tracing the vector field:

$$\frac{\partial}{\partial t}\Phi_t = \mathbf{v} \circ \Phi_t, \quad \Phi_0 = \mathrm{id}_M . \tag{15}$$

Let the *inverse flow map* $\Psi_t : M_t \to M_0$ be the inverse function of the flow map, $\Psi_t = \Phi_t^{-1}$. By taking the derivative of $\Psi_t \circ \Phi_t = \mathrm{id}_M$ and (15), the equation satisfied by Ψ_t is

$$\frac{\partial}{\partial t}\Psi_t = -d\Psi_t[\![\mathbf{v}]\!], \quad \Psi_0 = \mathrm{id}_M . \tag{16}$$

The classical transport equation for a scalar field $q_t : M_t \to \mathbb{R}$ is given by

$$\frac{D}{Dt} q_t = \left(\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla\right) q_t = 0, \tag{17}$$

which describes that the value of q is invariant along the flow, $q_t(\mathbf{x}) = q_0(\Psi_t(\mathbf{x}))$ for all $\mathbf{x} \in M_t$. Simply put, for any scalar field $q_t : M_t \to \mathbb{R}$,

$$\boxed{\frac{D}{Dt} q_t = 0 \quad \Leftrightarrow \quad q_t = \Psi_t^* q_0.} \tag{18}$$

The transportation for a covector field $\xi_t \in \mathfrak{X}^*(M_t)$ is also characterized by the flow-invariance in terms of pullback

$$\xi_t = \Psi_t^* \xi_0, \quad \text{or equivalently} \quad \xi_0 = \Phi_t^* \xi_t. \tag{19}$$

By taking $\partial/\partial t$ on both sides of $\xi_0 = \Phi_t^* \xi_t$, we obtain[4]

$$0 = \frac{\partial}{\partial t}\left(\Phi_t^* \xi_t\right) \overset{(12)}{=} \Phi_t^* \left(\frac{\partial}{\partial t}\xi_t\right) + \Phi_t^* \left(\mathscr{L}_{\mathbf{v}}\, \xi_t\right). \tag{20}$$

This relation is similarly summarized as

$$\boxed{\left(\frac{\partial}{\partial t} + \mathscr{L}_{\mathbf{v}}\right) \xi_t = 0 \quad \Leftrightarrow \quad \xi_t = \Psi_t^* \xi_0.} \tag{21}$$

We call the equation $(\frac{\partial}{\partial t} + \mathscr{L}_{\mathbf{v}})\xi_t = 0$ the *transport equation for the covector fields*. We shall refer to $(\frac{\partial}{\partial t} + \mathscr{L}_{\mathbf{v}})$ as the *Lie material derivative*.

The distinction between (18) and (21) is demonstrated in Fig. 5 where a covector field is transported with a rigid rotating flow. Evolving a covector field using traditional advection (18) (Fig. 5b) only transports the field component-wise. Advection with (21) correctly co-rotates the covector field (Fig. 5c).

Geometrically, the covector Lie material derivative measures the rate of change of the line integral of a covector field ξ_t along any curve C pushed by the flow Φ_t,

$$\frac{d}{dt}\int_{\Phi_t(C)} \xi_t = \int_{\Phi_t(C)} \left(\frac{\partial}{\partial t} + \mathscr{L}_{\mathbf{v}}\right) \xi_t. \tag{22}$$

In particular, the covector field satisfies the transport equation if and only if its line integral along the flowing curve $\Phi_t(C)$ stays constant over time.

Remark 1. *In terms of the vector counterpart* $\mathbf{a}_t = \xi_t^\sharp$*, the covector transport equation reads (*cf. (14)*)*

$$\frac{\partial}{\partial t}\mathbf{a}_t + \mathbf{v} \cdot \nabla \mathbf{a}_t + (\nabla \mathbf{v}) \cdot \mathbf{a}_t = 0. \tag{23}$$

This equation is also the basis for the impulse methods *[Buttke 1993; Cortez 1995; Feng et al. 2022]. In the literature,* (23) *is treated as a non-trivial dynamical system for each flowing particle* $\frac{D\mathbf{a}_t}{Dt} = -(\nabla \mathbf{v}) \cdot \mathbf{a}_t$*. According to* (21) *and* (10), (23) *has an explicit integral solution*

$$\mathbf{a}_t(\mathbf{x}) = d\Psi_t^\intercal \mathbf{a}_0(\Psi_t(\mathbf{x})). \tag{24}$$

[4] In (12), we define Lie derivative for time-independent covector field ξ and time-dependent forward flow map Φ_t. Adding $\frac{\partial}{\partial t_1}\Phi_{t_1}^*\xi_{t_2} = \Phi_{t_1}^*(\mathcal{L}_{\mathbf{v}}\xi_{t_2})$ to $\frac{\partial}{\partial t_2}\Phi_{t_1}^*\xi_{t_2} = \Phi_{t_1}^*(\frac{\partial \xi_{t_2}}{\partial t_2})$ with the relation $t_1 = t_2 = t$, we obtain the subsequent equation.

In later sections, we will apply this formula to solve the covector transport equation.

4.2 Euler equation in covector form

Here we derive the incompressible Euler equation in covector form. Adding $(\nabla \mathbf{u}) \cdot \mathbf{u} = \frac{1}{2}\nabla|\mathbf{u}|^2$ to both sides of (1) yields

$$\frac{\partial}{\partial t}\mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + (\nabla \mathbf{u}) \cdot \mathbf{u} = -\nabla\left(p - \tfrac{1}{2}|\mathbf{u}|^2\right). \tag{25}$$

Following Remark 1, the left-hand side is the Lie material derivative of the covector field $\eta = \mathbf{u}^\flat$, and by Section 3.3, the right-hand side is the differential of $\lambda := p - \frac{1}{2}|\mathbf{u}|^2$. We call η the *velocity covector field* and λ the *Lagrangian pressure*. Rewriting (25) with the notations introduced in Section 3, we obtain the incompressible Euler equation for the velocity covector field

$$\frac{\partial}{\partial t}\eta + \mathscr{L}_{\mathbf{u}}\,\eta = -d\lambda, \quad \mathbf{u} = \eta^\sharp, \quad \nabla \cdot \mathbf{u} = 0. \tag{26}$$

Although they are equivalent, (1) and (26) focus on different motion laws. The left-hand side of (1) measures the rate of change of the *linear momentum* $\mathbf{u}$ of every flowing *particle*. On the other hand, the left-hand side of (26) measures the rate of change of the *line integral* $\int_{C_t} \eta = \int_{C_t} \mathbf{u} \cdot d\mathbf{l}$ along every *curve* C_t flowing with the fluid (see (22)). Equation (26) as a whole describes that this rate of change of the line integral matches the difference of the Lagrangian pressure λ at the two ends of the curve[5] [Thomson 1868]. This property is a generalization of the better-known conservation of circulation on flowing closed curves [Chorin and Marsden 1990].

The difference between the two motion laws (1) and (26) becomes apparent in a flow dominated by vorticity. In (1), the pressure force is responsible for the concentration of vorticity. Without the pressure, the inertial motion (conservation of linear momentum) turns into a centrifugal force that makes vortices disintegrate. In (26), the Lagrangian pressure plays *no* role in the persistence of vorticity. The mechanism for the conservation of vorticity is entirely encoded in the left-hand side of the equation. Such a property makes an algorithm based on time-splitting into advection and pressure steps especially appealing (Section 4.3). We explore this property in more detail in Section 4.4 and Section 4.5.

4.3 The Covector Fluids (CF) method

Now we describe our *Covector Fluids* (CF) method. Similar to a classical advection-projection solver (Section 1), CF advances the fluid state $\mathbf{u}$ by splitting (26) into the following three substeps.

1: Estimate a flow velocity, *e.g.* $\mathbf{v} \leftarrow \mathbf{u}$; ▷ freeze flow velocity
2: Solve $(\frac{\partial}{\partial t} + \mathscr{L}_{\mathbf{v}})\mathbf{u}^\flat = 0$ for Δt time; ▷ covector advection
3: $\mathbf{u} \leftarrow \mathbf{u} - \Delta t \nabla\lambda$ so that $\nabla \cdot \mathbf{u} = 0$ afterwards. ▷ projection

These steps are visualized in Fig. 6 along side traditional advection-projection and vortex methods. In particular, the advection step (Step 2) is carried out by (3),

$$\mathbf{u}(\mathbf{x}) \leftarrow (d\Psi(\mathbf{x}))^\intercal \mathbf{u}(\Psi(\mathbf{x})), \quad \mathbf{x} \in M, \tag{27}$$

where Ψ is the inverse flow map generated by velocity $\mathbf{v}$ for a time step of Δt (Remark 1). The minor modification of (27) from the standard methods allows one to map the technology in standard fluid methods to the new method rather seamlessly (Section 5).

[5]If a curve C connects point $\mathbf{a}$ to point $\mathbf{b}$, then $\int_C d\lambda = \lambda(\mathbf{b}) - \lambda(\mathbf{a})$.

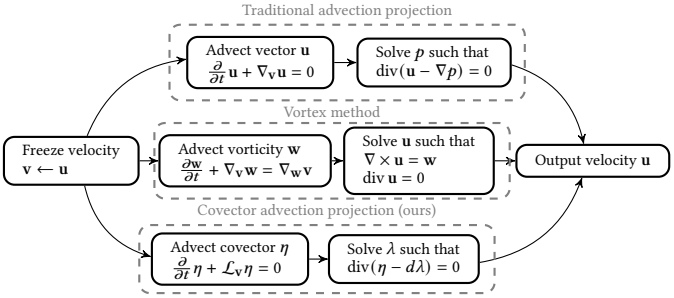


Fig. 6. Vortex methods are better at capturing vorticity, with an expensive velocity reconstruction as a trade-off. Our method emulates the vortex method with comparable costs to traditional advection methods.

In the remaining parts of Section 4, we explain the advantage of the CF advection

$$\left(\frac{\partial}{\partial t} + \mathscr{L}_{\mathbf{v}}\right)\eta = 0, \quad \eta = \mathbf{u}^\flat, \tag{28}$$

or (Remark 1)

$$\frac{\partial}{\partial t}\mathbf{u} + \mathbf{v} \cdot \nabla \mathbf{u} + (\nabla \mathbf{v}) \cdot \mathbf{u} = \mathbf{0}, \tag{29}$$

over the traditional advection $(\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla)\mathbf{u} = \mathbf{0}$.

4.4 Equivalence to a vortex method

The traditional advection-projection method introduces a splitting error that destroys vorticity [Zhang et al. 2015; Zehnder et al. 2018]. This phenomenon arises solely from the advection step, since the projection step only modifies the velocity $\mathbf{u}$ with a pure gradient which leads to no change in the its curl.

Under the classical advection equation $(\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla)\mathbf{u} = \mathbf{0}$, the vorticity $\mathbf{w} = \nabla \times \mathbf{u}$ evolves according to[6] (Appendix A.3)

$$\frac{\partial}{\partial t}\mathbf{w} + \mathbf{v} \cdot \nabla \mathbf{w} - \mathbf{w} \cdot \nabla \mathbf{v} = \langle \nabla \mathbf{u} \times \nabla \mathbf{v} \rangle. \tag{30}$$

This *modified vorticity equation* deviates from the correct vorticity equation by a term $\langle \nabla \mathbf{u} \times \nabla \mathbf{v} \rangle$. By contrast, the evolution of $\mathbf{w} = \nabla \times \mathbf{u}$ that undergoes (29) is (Appendix A.2)

$$\frac{\partial}{\partial t}\mathbf{w} + \mathbf{v} \cdot \nabla \mathbf{w} - \mathbf{w} \cdot \nabla \mathbf{v} = \mathbf{0} \tag{31}$$

which is the correct vorticity equation. By advancing $\mathbf{u}$ via the covector transportation (28) or (29), we implicitly solve the vorticity equation (31), which is the modeling equation for vortex methods.

The ability to solve (31) at the velocity level without using the vorticity variable is significant. Previous vortex methods which solve (31) have to include an expensive integration that converts vorticity back to velocity.

4.5 Commutativity between covector transportation and pressure projection

In a traditional fluid solver, the splitting error between traditional advection and the projection arises because two operations do not commute. Here we show that advection and projection *commute* in CF. This property of CF fundamentally removes the splitting error of these two operations.

[6]In index notation $\langle \nabla \mathbf{u} \times \nabla \mathbf{v} \rangle_i = \epsilon_{ijk}\partial_j u^\ell \partial_k v^\ell$.

Consider the equivalence classes of $\mathfrak{X}^*(M)$ where $[\xi] = [\eta]$ whenever $\xi - \eta = d\varphi$ for some function φ. This is a natural abstraction for our discussion since two covector fields are equivalent if and only if they share the same pressure projection result. The pressure projection can be understood as extracting the unique divergence-free representative in each equivalence class $[\eta] \in \mathfrak{X}^*(M)/\mathrm{im}(d)$.

Now consider two covector fields $\xi_0, \eta_0 \in \mathfrak{X}^*(M)$, divergence-free or not, and transport them by the covector advection equation (28) to obtain ξ_t, η_t respectively. Then

$$[\xi_0] = [\eta_0] \quad \text{if and only if} \quad [\xi_t] = [\eta_t]. \tag{32}$$

To see this assertion, use (21) to express $\xi_t - \eta_t$ as the pullback of $\xi_0 - \eta_0$ by the inverse flow map

$$\xi_t - \eta_t = \Psi_t^*(\xi_0 - \eta_0) \tag{33}$$

and apply (7) so that the pullback of an exact differential is still exact. Therefore, whether one first projects then advects, advects then projects, or inserts a projection (or reflection [Zehnder et al. 2018]) at the halfway point of the advection, one will obtain covector fields all in the same equivalence class and hence the same divergence-free representative.

Note that the same argument does not apply to the traditional advections. The transportation with a simple value look-up (2) generally turns an exact gradient vector field into a non-gradient field.

4.6 Extending CF with a long-time characteristic mapping

The analysis in Section 4.5 implies that we may delay the pressure projection for a longer time (rather than a time step) and just transport the velocity covector field η using a long-time flow map.

Let $\Psi_t : M_t \to M_0$ be the inverse flow map (Eulerian-to-Lagrangian map) which is the Lagrangian marker carried by the history of the solution

$$\frac{\partial}{\partial t}\Psi_t + \mathbf{u}_t \cdot \nabla \Psi_t = \mathbf{0}, \quad \Psi_0 = \mathrm{id}_M . \tag{34}$$

Then the velocity covector field at the current time t is the pressure projection of

$$\eta_t = \Psi_t^* \eta_0. \tag{35}$$

In other words, if we maintain a Lagrangian marker Ψ_t we obtain the fluid state through a single-step look up. This drastically reduces the amount of interpolation in the advection-projection iteration.

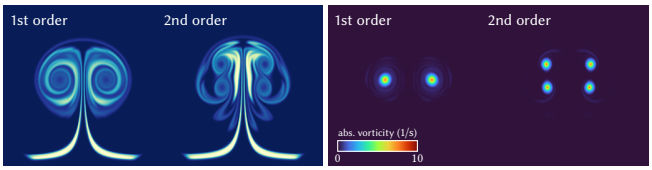
Such a long-time method of characteristic mapping (MCM) is proposed by [Tessendorf and Pelfrey 2011; Sato et al. 2018; Qu et al. 2019]. We call the variant of the CF method based on (34) and (35) *CF+MCM*.

Remark 2. *CF+MCM is subtly different from the traditional MCM. The latter requires an accumulation of the pressure gradient over time whereas the former does not. In the traditional MCM [Qu et al. 2019], one evaluates* $\mathbf{u}_t$ *by* (1) *integrating over time along particle trajectories:*

$$\mathbf{u}_t(\mathbf{x}) = \mathbf{u}_0(\Psi_t(\mathbf{x})) + \int_0^t (\nabla p_\tau)(\Phi_\tau(\Psi_t(\mathbf{x}))) \, d\tau. \tag{36}$$

In CF+MCM, the time integration of (26) *yields*

$$\begin{aligned} \eta_t &= \Psi_t^* \eta_0 + \int_0^t (\Phi_\tau \circ \Psi_t)^* d\lambda_\tau \, d\tau \\ &= \Psi_t^* \eta_0 + d\left(\int_0^t (\Phi_\tau \circ \Psi_t)^* \lambda_\tau \, d\tau \right). \end{aligned} \tag{37}$$



(a) Density. (b) Vorticity.

Fig. 7. 2D leapfrogging demonstrated with our method in its 1st and 2nd order variants. By lowering diffusion with the 2nd order covector advection scheme, the leapfrogging phenomenon is better captured.

which in a comparable notation to (36) *corresponds to*

$$\begin{aligned} \mathbf{u}_t(\mathbf{x}) &= d\Psi_t^{\mathsf{T}} \mathbf{u}_0(\Psi_t(\mathbf{x})) \\ &\quad + \int_0^t d\Psi_t^{\mathsf{T}} \, d\Phi_\tau^{\mathsf{T}} (\nabla(p_\tau - \tfrac{|\mathbf{u}_\tau|^2}{2}))(\Phi_\tau(\Psi_t(\mathbf{x}))) \, d\tau \\ &= d\Psi_t^{\mathsf{T}} \mathbf{u}_0(\Psi_t(\mathbf{x})) + \nabla \left(\int_0^t (p_\tau - \tfrac{|\mathbf{u}_\tau|^2}{2})(\Phi_\tau(\Psi_t(\mathbf{x}))) \, d\tau \right). \end{aligned} \tag{38}$$

As omitted in (35)*, the second terms in* (37) *and* (38) *can be absorbed in a single pressure projection as they are exact differentials. This is possible since d (resp.* ∇*) in* (37) *(resp.* (38)*) can be pulled out of the integral by the commutativity property (*cf. (7)*) between d and pullback operators. By contrast in* (36)*, the* ∇ *in the pressure term cannot be pulled out of the time integral. The accumulated pressure* $\int_0^t (\nabla p_\tau) \circ \Phi_\tau \circ \Psi_t \, d\tau$ *is generally not an exact gradient. Therefore, a traditional MCM must carefully record the accumulated pressure. This procedure is entirely removed in CF+MCM.*

5 ALGORITHM

In this section, we describe the algorithmic details of the Covector Fluids (CF) method as discussed in Section 4.3. Our core advection based on (28) can replace the traditional advection step and enable us to leverage all of the standard solution techniques.

We use the following notations similar to [Zehnder et al. 2018]. Let $\mathscr{A}(q; \mathbf{v}, \Delta t)$ denote an advection solver that solves $(\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla) q = 0$ for a Δt timestep for a generic field q. Let $\mathscr{A}_{\mathrm{covec}}(\mathbf{u}; \mathbf{v}, \Delta t)$ denote a Lie transportation $(\frac{\partial}{\partial t} + \mathscr{L}_{\mathbf{v}})\mathbf{u}^\flat = 0$ of a covector field $\mathbf{u}^\flat$. We detail the algorithm of the advection step in Alg. 3 and Alg. 4. Let $\mathscr{P} : \mathfrak{X}(M) \to \mathfrak{X}(M)$ be the pressure projection operator.

5.1 Base method

Our base method is given by

Algorithm 1 Covector Fluids (1st order)

Input: Initial velocity $\mathbf{u}$; step size Δt;
1: **for each** time step **do**
2: $\quad \mathbf{v} \leftarrow \mathbf{u}$; ▷ freeze flow velocity
3: $\quad \mathbf{u} \leftarrow \mathscr{A}_{\mathrm{covec}}(\mathbf{u}; \mathbf{v}, \Delta t)$; ▷ covector Lie advection
4: $\quad \mathbf{u} \leftarrow \mathscr{P}(\mathbf{u})$; ▷ pressure projection
5: **end for**

A simple modification using the *midpoint method* (a 2nd order Runge–Kutta method) can reduce the truncation error arising from the freezing of the flow velocity (Fig. 7):

Algorithm 2 Covector Fluids (2nd order)

Input: Initial velocity $\mathbf{u}$; step size Δt;
1: **for each** time step **do**
2: $\quad \mathbf{v} \leftarrow \mathscr{P}\left(\mathscr{A}_{\text{covec}}(\mathbf{u}; \mathbf{u}, \frac{\Delta t}{2})\right)$; ▷ estimate flow velocity at $\frac{\Delta t}{2}$
3: $\quad \mathbf{u} \leftarrow \mathscr{P}(\mathscr{A}_{\text{covec}}(\mathbf{u}; \mathbf{v}, \Delta t))$; ▷ full step
4: **end for**

5.2 Covector advection

The core covector advection solver is based on (27). A direct application of (27) yields the following *semi-Lagrangian* (sL) method for the covector advection $\mathscr{A}^{\text{sL}}_{\text{covec}}(\mathbf{u}; \mathbf{v}, \Delta t)$.

Algorithm 3 Subroutine $\mathscr{A}^{\text{sL}}_{\text{covec}}(\mathbf{u}; \mathbf{v}, \Delta t)$

Input: Field to advect $\mathbf{u}$, flow velocity $\mathbf{v}$, time span Δt
1: **for each** point $\mathbf{x} \in M$ **do** ▷ construct inverse flow map
2: $\quad \Psi(\mathbf{x}) \leftarrow$ RK4 backtrace from $\mathbf{x}$ with flow velocity $\mathbf{v}$;
3: **end for**
4: $\mathbf{u} \leftarrow d\Psi^\intercal(\mathbf{u} \circ \Psi)$; ▷ Pullback (Section 5.4.1)
Output: $\mathbf{u}$;

Our covector advection routine $\mathscr{A}^{\text{sL}}_{\text{covec}}$ is simple enough, and with a few function calls, one can derive the *back-and-forth error compensation and correction* (BFECC) [Dupont and Liu 2003; Kim et al. 2005; Selle et al. 2008] for covector advection. Fig. 5 shows that the covector BFECC advection transports velocity correctly in a non-dissipative manner, while the covector semi-Lagrangian advection and the vanilla vector-based BFECC yield a dissipated result.

Algorithm 4 Covector advection solver $\mathscr{A}^{\text{BFECC}}_{\text{covec}}(\mathbf{u}; \mathbf{v}, \Delta t)$

Input: field to advect $\mathbf{u}$, flow velocity $\mathbf{v}$, time span Δt
1: $\mathbf{u}_1 \leftarrow \mathscr{A}^{\text{sL}}_{\text{covec}}(\mathbf{u}; \mathbf{v}, \Delta t)$;
2: $\tilde{\mathbf{u}}_0 \leftarrow \mathscr{A}^{\text{sL}}_{\text{covec}}(\mathbf{u}_1; \mathbf{v}, -\Delta t)$; ▷ back-and-forth advection
3: $\mathbf{e} \leftarrow \tilde{\mathbf{u}}_0 - \mathbf{u}$; ▷ roundtrip error
4: $\mathbf{u} \leftarrow \mathbf{u}_1 - \mathscr{A}^{\text{sL}}_{\text{covec}}(\frac{\mathbf{e}}{2}; \mathbf{v}, \Delta t)$; ▷ error correction
Output: $\mathbf{u}$;

5.3 Characteristic mapping (CF+MCM)

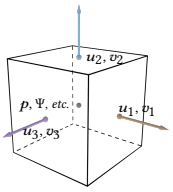
As we described in Section 4.6, the covector advection integrates seamlessly with methods of characteristic mappings (MCM). We combine our method and BiMocq [Qu et al. 2019] (CF+MCM) by delaying the reinitialization of the Lagrangian Marker Ψ_t. This largely reduces the amount of interpolation during subsequent advection steps, by only needing to advect the map Ψ and not the velocity components. The calculation of $d\Psi^\intercal(\mathbf{u} \circ \Psi)$ can utilize the values from this map, where $\mathbf{u}$ is a snapshot of the velocity from the last reinitialization event. This incurs only a minor change to our previous algorithms, as shown in Alg. 5. We choose the same reinitialization criteria as [Qu et al. 2019], where the map is reset either after a certain number of frames or when the map is no longer accurate (i.e. $\Phi \circ \Psi \neq \Psi \circ \Phi$).

Algorithm 5 Covector Fluids (CF+MCM)

Input: Initial velocity $\mathbf{u}$; step size Δt;
1: $\mathbf{u}_0 \leftarrow \mathbf{u}$; backward map $\Psi \leftarrow$ id; forward map $\Phi \leftarrow$ id;
2: **for each** time step **do**
3: $\quad \mathbf{v} \leftarrow \textsc{EstimateVelocity}_{t+\Delta t/2}$; ▷ for midpoint method
4: $\quad \Psi \leftarrow \mathscr{A}(\Psi; \mathbf{v}, \Delta t)$; ▷ advect inverse flow map
5: $\quad \mathbf{u}_1 \leftarrow d\Psi^\intercal(\mathbf{u}_0 \circ \Psi)$; ▷ pullback velocity
6: $\quad \Phi \leftarrow \textsc{SolveODE}(\frac{\partial \Phi}{\partial t} = \mathbf{v} \circ \Phi; \Delta t)$; ▷ march flow map
7: $\quad \tilde{\mathbf{u}}_0 \leftarrow d\Phi^\intercal(\mathbf{u}_1 \circ \Phi)$; ▷ back-and-forth transport
8: $\quad \mathbf{e} \leftarrow \tilde{\mathbf{u}}_0 - \mathbf{u}$; ▷ roundtrip error
9: $\quad \mathbf{u} \leftarrow \mathbf{u}_1 - d\Psi^\intercal(\frac{1}{2}\mathbf{e} \circ \Psi)$; ▷ error correction
10: $\quad \mathbf{u} \leftarrow \mathscr{P}(\mathbf{u})$; ▷ pressure projection
11: $\quad$ **if** reinitialization condition **then**
12: $\quad\quad \Phi, \Psi \leftarrow$ id, ▷ reset flow maps
13: $\quad\quad \mathbf{u}_0 \leftarrow \mathbf{u}$; ▷ reset velocity
14: $\quad$ **end if**
15: **end for**

Line 6–9 is the BFECC treatment, where we use RK4 for line 6. We estimate the flow velocity at line 3 by the result of line 4–10 using flow $\mathbf{v} = \mathbf{u}$.

5.4 Additional details



5.4.1 Staggered Grid. We use the standard MAC grid [Harlow and Welch 1965; Bridson 2015] to store our variables (see inset). In Step 4 of Alg. 3, we evaluate the velocity $\mathbf{u} \circ \Psi$ by tracing back face centers and the Jacobian $d\Psi^\intercal$ on the corresponding entries of the Jacobian on face centers using finite difference from neighboring cell centers. For example, the first component of the equation $\mathbf{u} \leftarrow d\Psi^\intercal(\mathbf{u} \circ \Psi)$ is computed on the face center F_{ij} with $\mathbf{e}_1$ normal located between cell C_i and C_j:

$$u_1\big|_{F_{ij}} \leftarrow \left[\begin{array}{ccc} \frac{\partial \Psi_x}{\partial x} & \frac{\partial \Psi_y}{\partial x} & \frac{\partial \Psi_z}{\partial x} \end{array}\right]\Big|_{F_{ij}} \left[\begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array}\right]\Big|_{\Psi(F_{ij})}, \tag{39}$$

where

$$\frac{\partial \Psi_\alpha}{\partial x}\Big|_{F_{ij}} = \frac{1}{\Delta x}\left(\Psi_\alpha\big|_{C_j} - \Psi_\alpha\big|_{C_i}\right), \quad \alpha \in \{x, y, z\}. \tag{40}$$

5.4.2 BFECC Limiter. The BFECC technique requires an *extrema* (minmod) *limiter* [Selle et al. 2008; Qu et al. 2019] to reduce the oscillatory dispersion. When using Alg. 4, the output value $\mathbf{u}$ is clamped to the minimum and maximum values of $\mathbf{u}_1$ componentwise within the immediate neighboring stencils.

5.4.3 Conditional Stability. Our method is numerically stable given a reasonably small step-size Δt. For the trefoil knot (Fig. 9) and smoke plume (Fig. 14) experiments, we tested various timesteps to find the maximal Δt where the simulation remains stable. The critical CFL number is this largest Δt normalized with the grid size Δx and the maximal fluid speed $U = \max_{\mathbf{x},t} |\mathbf{u}(\mathbf{x}, t)|$

Simulation	C
IVOCK	1.45
SCPF	4.85
CF (Ours)	6.18

$$C = \frac{U \Delta t^{\text{critical}}}{\Delta x}. \tag{41}$$

We use C to compare our conditional stability against various vorticity methods: IVOCK [Zhang et al. 2015] and SCPF [Elcott et al.

2007] (see inset). In our experiments we observe a stability condition similar to SCPF and much stabler than advect-and-stretch methods such as IVOCK.

5.4.4 Buoyancy. For gravity or buoyancy force, we apply the Boussinesq model [Qu et al. 2019] $\mathbf{f} = c\rho\mathbf{g}$ where is c is a constant, $\mathbf{g}$ the gravity, and ρ is an advected scalar field representing density or temperature. We update the velocities by $\mathbf{u} \leftarrow \mathbf{u} + \Delta t\mathbf{f}$ right before the pressure projection.

5.4.5 Boundary Conditions. We employ standard boundary conditions for fluid simulation [Bridson 2015]. To prescribe solid boundaries, we choose a Neumann, commonly referred to as a *no stick*, boundary condition where the normal component of the velocity is equal to the velocity of either the walls (zero) or obstacles ($\mathbf{u}_{\text{solid}}$). To set free surfaces, we select a Dirichlet boundary condition where pressure on the interface is set to zero. This causes a pressure gradient and velocity pointing outwards from the domain, allowing the smoke to leave. The tracebacks during advection are another computation that require boundary treatments. If the traceback reaches outside of the domain, we set the field value using the closest point inside the domain (sometimes referred to as the *streak* boundary).

6 RESULTS

In this section we discuss the numerical experiments conducted to demonstrate our method. We modified the codebase shared by [Qu et al. 2019] for our method as well as for the comparisons. The codebase offers implementations of Stable Fluids, MacCormack, Reflection, and BiMocq methods. On top of adding our Covector Fluids algorithm to the codebase, we also implemented a few additions (*e.g.* 2nd order MC+R [Narain et al. 2019], RK4 traceback, and SCPF [Elcott et al. 2007]) for a thorough comparison with relevant methods. We include our modified codebase in the supplementary material.

We run the 2nd order version of CF (Alg. 2) with BFECC (Alg. 4) in all of our experiments except in Fig. 7, where we compare 2nd order advection against 1st order. To keep the comparisons fair, we also run the MC+R method in its 2nd order variant [Narain et al. 2019]. For experiments of BiMocq [Qu et al. 2019] and CF+MCM, we use one-level mapping, set reinitialization frequency to every 5 time steps, and adopt Dual Mesh Characteristics (DMC) (3D BiMocq) and RK4 (2D BiMocq and all CF+MCM) for mapping advection.

Performance Summary. We performed our 2D experiments with CPU-parallelism on a laptop with 2.3GHz 8-Core Intel Core i9 processor and 16GB of memory. For 3D experiments, the advection part is GPU-parallelized using a CUDA implementation while the remaining computations stay on the CPU. The 3D results were computed on a desktop machine with 3.6GHz 8-Core Intel Core i7-9700K processor, NVIDIA GeForce RTX 2080, and 16GB of memory. We summarize the performance details of our method in Table 2. Overall, we observe a 15% increase in computational time of a full time step compared to existing methods. This computational cost is mainly due to tracing back Ψ in addition to (u_1, u_2, u_3) compared to traditional methods (*e.g.* MC). When compared against BiMocq [Qu et al. 2019], we utilize the existing tracebacks of Ψ for our $d\Psi^\intercal$ computation and the increase in our computational cost is mainly due to the 2nd order velocity estimate. This is a relatively small computational overhead for our improvement in capturing vortex dynamics.

Table 2. Performance and statistics.

Figure	Domain size	Grid resolution	Δt	Comp. time/step CF	Comp. time/step CF+MCM
Ink jet (Fig. 1)	$5\times10\times5$ m^3	$128\times256\times128$	1/48 s	12.2 s	9.3 s
Bunny meteor (Fig. 2)	$10\times5\times5$ m^3	$256\times128\times128$	1/96 s	12.7 s	10.8 s
Von Kármán vortex street (Fig. 4)	$2\pi\times\pi$ m^2	512×256	1/10 s	1.3 s	1.3 s
Covector transport (Fig. 5)	1×1 m^2	200×200	2 s	21 ms	15 ms
Leapfrogging pairs (Fig. 7)	$2\pi\times2\pi$ m^2	256×256	1/40 s	0.3 s	0.3 s
Taylor vortices (Fig. 8)	$2\pi\times2\pi$ m^2	256×256	1/40 s	0.2 s	0.2 s
Trefoil knot (Fig. 9)	$10\times5\times5$ m^3	$256\times128\times128$	1/48 s	8.0 s	4.8 s
Leapfrogging rings (Fig. 10)	$10\times5\times5$ m^3	$256\times128\times128$	1/48 s	6.4 s	4.8 s
Ink drop (Fig. 11)	0.2×0.2 m^2	512×512	1/100 s	1.4 s	1.2 s
SIGGRAPH ink drop (Fig. 12)	0.2×0.25 m^2	420×520	1/100 s	1.2 s	1.0 s
Pyroclastic cloud (Fig. 13)	$5\times10\times5$ m^3	$128\times256\times128$	1/192 s	11.9 s	10.7 s
Smoke plume (Fig. 14)	$5\times10\times5$ m^3	$128\times256\times128$	1/192 s	11.3 s	9.7 s
Ground truth comp. (Fig. 15)	$2\pi\times\pi$ m^2	1024×512	1/40 s	2.5 s	2.1 s
Moving obstacle (Fig. 16)	$10\times5\times5$ m^3	$256\times128\times128$	1/48 s	9.2 s	5.1 s
Delta wing (Fig. 17)	$5\times5\times5$ m^3	$128\times128\times128$	1/96 s	6.7 s	5.9 s

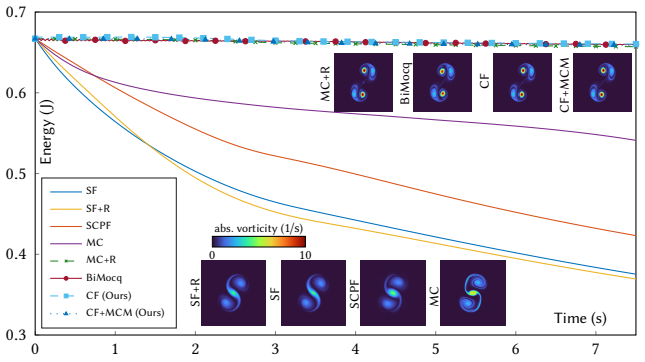
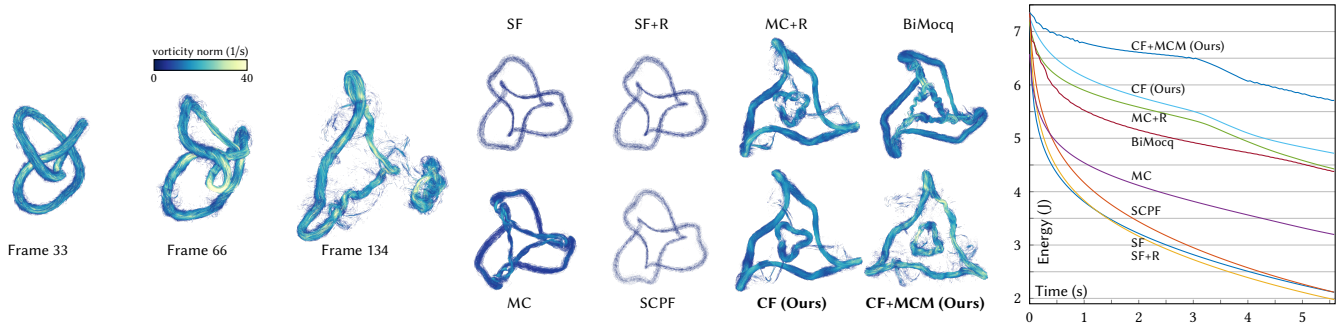


Fig. 8. Evolution of Taylor vortices simulated with different methods. Using our method, energy is conserved as well as energy preserving methods such as MC+R. Note that error correction schemes, such as BFECC/MC, are crucial to energy preservation.

6.1 Validation

We evaluate our method in both 2D (Fig. 8) and 3D (Fig. 9 and Fig. 10) experiments.

6.1.1 2D Taylor Vortices (Fig. 8). Following the setup from [McKenzie 2007; Qu et al. 2019], two shielded gaussian vortices, commonly referred to as *Taylor vortices*, are placed 0.81 meters apart. Concretely, the vorticity distribution for each vortex is given by $\omega(\mathbf{x}) = U/a(2 - r^2/a^2)\exp(0.5(1 - r^2/a^2))$, where r is the distance from $\mathbf{x}$ to the vortex center, $a = 0.3$ m is the core size, and $U = 1$ m/s is the maximum tangential velocity. We plot the energy loss of our methods, compared to previous ones, in Fig. 8. There are two main sources to this energy loss: splitting the Euler equation into an advection step and a projection step, and the sampling error during the advection step. Previous methods (*e.g.* SF+R or SCPF) that target only the splitting error fail to preserve the energy because they still suffer from sampling error. Our method uses covector advection and



(a) Evolution of trefoil knot by our CF+MCM method. (b) Trefoil knot simulated using different methods. (c) Energy plot.

Fig. 9. Trefoil Knot evolution simulated under different methods. Our methods capture the vortex seperation of the knot as shown in physical experiments [Kleckner and Irvine 2013] (a), while best maintaining vortex strength (b). This experiment further shows that our methods improve energy preservation compared to recent methods such as MC+R and BiMocq (c).

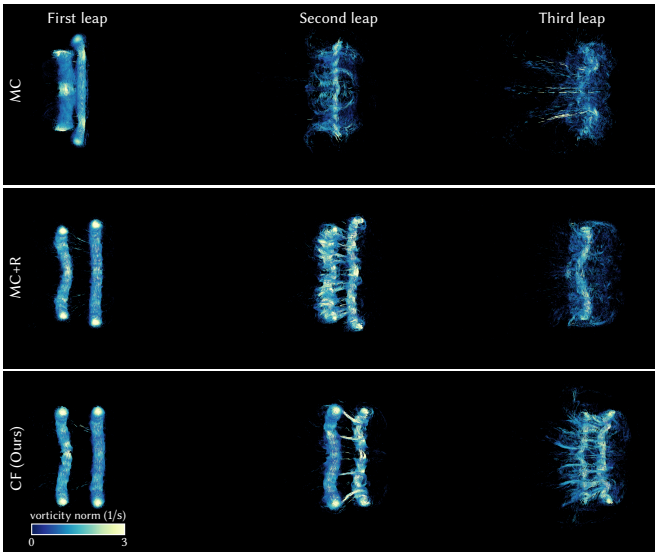


Fig. 10. A pair of leapfrogging vortex rings in 3D. Here, a cross-section of the vorticity of the rings after each leapfrog is shown. Using our method, we are able to preserve the vortex rings after three leapfrogs, while other methods merge during earlier leaps and heavily lose their vorticity.

BFECC to remove both the splitting and the sampling error, and is thus able to effectively preserve energy and match the results from previous methods like MC+R or BiMocq.

6.1.2 3D Trefoil Knot (Fig. 9). We use the parametric equation for the *trefoil knot*

$$\begin{bmatrix} x(\tau) \\ y(\tau) \\ z(\tau) \end{bmatrix} = \begin{bmatrix} \sin(\tau)+2\sin(2\tau) \\ \cos(\tau)-2\cos(2\tau) \\ -\sin(3\tau) \end{bmatrix}, \quad \tau \in [0, 2\pi),$$

to set up a knotted vortex with circulation 1 m^2/s. According to physical experiments [Kleckner and Irvine 2013], the trefoil knot should evolve and reconnect into one smaller and one larger ring. As shown in Fig. 9a, our method captures this vortical behavior. Our method also maintains stronger vorticity and shows better energy preservation compared to other methods (Fig. 9b, Fig. 9c).

6.1.3 3D Vortex Leapfrogging (Fig. 10). We initialize the experiments with two concentric rings of radii 1.2 m and 2.0 m. Both rings have a circulation of 0.56 m^2/s. In inviscid fluids, the two vortex rings should influence each other and leapfrog indefinitely. We use the duration of leapfrogging to compare different fluid simulators. As shown in Fig. 10, the vortex rings in our method remain separate beyond the third leap, while in the other methods the rings merge after one or two leaps.

6.2 Buoyancy and gravity

Fluids with spatially varying density are one of the main sources of intricate vortical phenomena. Examples include rising hot smoke and sinking heavy ink. The change in density results in a differential gravitational acceleration, which in turn creates a vortex sheet that eventually rolls up into mutually interacting vortices. Such a sequence of events is commonly referred to as the *Rayleigh–Taylor instability*. We set up computational experiments with the presence of gravitational acceleration and variation in density. We compare the qualitative results of this vortex dominated phenomenon using different fluid solvers.

6.2.1 2D Ink Drop (Fig. 11 and Fig. 12). We design a numerical experiment of a heavy ink drop sinking in a surrounding fluid in a 2-dimensional space. The ink drop is initialized as a disk with diameter 8 cm. The densities of the two phases are set so that the effective differential acceleration is 0.85 m/s^2. In an ideal continuous setting with little viscosity, the interface will nucleate vortices across multiple scales leading to a fractal with many details. As shown in Fig. 11, different methods achieve different levels of detail despite the same computational resolution. Methods with diffusive advections (SF, SF+R, SCPF) or splitting errors (SF, MC) can only reproduce vortices at a larger scale but fail to capture the small scale details. The non-diffusive reflection-based solvers (MC+R, BiMocq) are able to bring more small scale vortices. Our covector-based solvers outperform previous methods by achieving the most detailed vortical structures. In Fig. 12 we show that our method can apply to ink of a general shape to obtain attractive animations (see video 4:08).

6.2.2 3D Ink Jet (Fig. 1). In nature, when ink dye is shot into water at a high speed, the density evolves to form intricate vortical patterns.

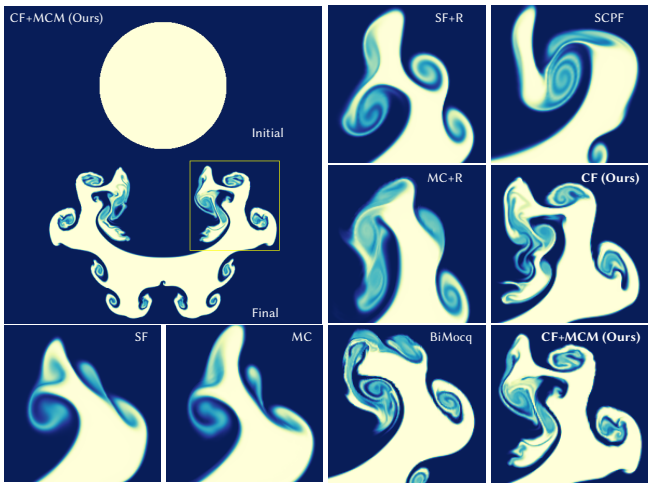


Fig. 11. Ink drop from an initially stationary sphere blob under the influence of gravity. Our method is able to achieve more detailed vortical structures compared to the state-of-the-art methods.

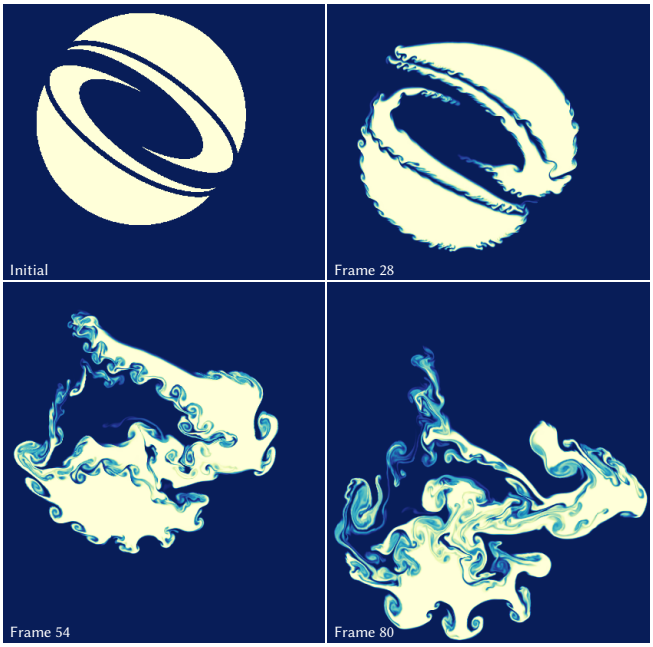


Fig. 12. Ink drop from an initially stationary blob in the shape of the SIGGRAPH logo under the influence of gravity. Our method is able to achieve highly detailed vortical structures.

We construct a similar setup for this experiment. Heavy ink is injected into the domain at a speed of 2 $^{m}/_{s}$. The density field is further accelerated under a gravity of 0.5 $^{m}/_{s^2}$. As presented in Fig. 1, our method faithfully recreates the vortical phenomenon. With the vorticity field visualized, we are able to see the detailed vortical structures evolving with the ink.

Fig. 13. Pyroclastic cloud forming out of hot smoke rising under buoyancy using our Covector Fluids (CF) method. Our method is capable of generating dense vortical structures, allowing for energetic and intricate smoke densities.

6.2.3 3D Smoke (Fig. 13 and Fig. 14). The evolution of smoke, where hot, buoyant smoke is released from an outlet into the air (*e.g.* volcanoes emitting pyroclastic clouds; see Fig. 13), is one fascinating phenomenon in nature that is commonly reproduced in computer animation. In this case, the denser and the more energetic the vortical structures formed, the more pleasant and realistic the plumes look to the eye. We use the amount of vorticity to compare our methods against the state-of-the-art methods.

To set up the experiment in Fig. 14, hot smoke is emitted into the scene from a ball with diameter 0.16 m. Given buoyancy acceleration of 5 $^{m}/_{s^2}$, the plume rapidly rises in the domain with time. As seen in Fig. 14, diffusive methods (SF, SF+R, SCPF) fail to produce noticeable vortical structures throughout the simulation. Previous methods (*e.g.* MC, MC+R, BiMocq) can reduce this diffusion, but our method delivers the most intricate vortical structure and the highest vorticity energy. We demonstrate this advantage of our method with pyroclastic clouds in Fig. 13, with a large circular smoke outlet of 0.46 m radius randomly injecting smoke and causing a buoyancy acceleration of 0.5 $^{m}/_{s^2}$.

6.3 Flow around obstacles

Generating flow around obstacles is essential in animations. With relatively low viscosity, the laminar flow around the obstacle sheds vortices downstream. In 2D (Fig. 4), the shed vortices form the iconic

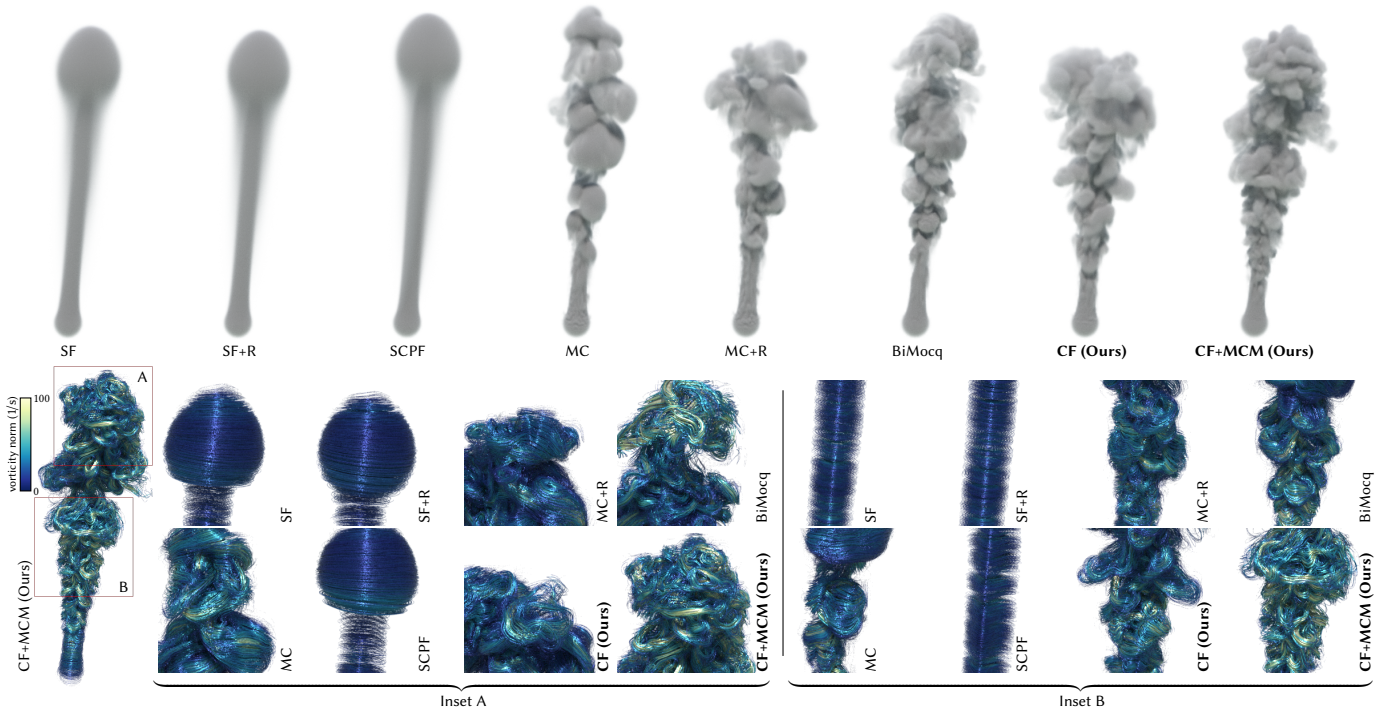


Fig. 14. Smoke plumes rising under buoyancy acceleration simulated with different methods. Top Row: Smoke plumes visualized with heavy density smoke. Bottom Row: Vorticity visualization of the smoke plumes compared in the insets. Note that our method is capable of producing much more interesting and detailed vortical structures throughout the simulation (see video 5:40).

von Kármán vortex street. In 3D, the vortex shedding generates turbulent wakes (Fig. 2) or structured vortices (Fig. 17) depending on the obstacle geometry.

6.3.1 2D von Kármán Vortex Street (Fig. 4 and Fig. 15). In this experiment, we place a disk in a laminar flow with a velocity of 0.5 m/s. The minor asymmetry of the obstacle causes the laminar flow to start shedding vortices and form a well known pattern, commonly known as the *von Kármán vortex street*. As seen in Fig. 4, our method manages to produce more vortices against the surface of the obstacle compared to other methods. This signifies that our method simulates fluid flows at higher Reynolds numbers. Previous methods either suffer from numerical diffusion causing the pattern to match the behavior of lower Reynolds numbers (*e.g.* MC) or produce unwanted noisy results (*e.g.* BiMocq).

To validate this claim, we compare our method against ground truth given by an established method (*e.g.* MC+R), both run with a high resolution, a small timestep, and a controlled Reynolds number. We add small amounts of viscosity that meets the given Reynolds numbers. Fig. 15 shows our method producing results consistent with the ground truth at both low (repeating pattern) and high (broken symmetry) Reynolds numbers [Blevins 1990]. Our results in Fig. 4 match a high Reynolds number von Kármán vortex street.

6.3.2 3D Moving Obstacle (Fig. 16). A known weakness of vortex-based methods is the difficulty of dealing with moving obstacles [Bridson 2015]. Since our method is velocity-based, we employ the standard approach of modifying the no-through boundary condition in the pressure projection step [Bridson 2015]. In this experiment, the speed of the moving obstacle is set to 1 m/s. Our method handles the moving boundary and leaves intricate vortex wakes (Fig. 16).

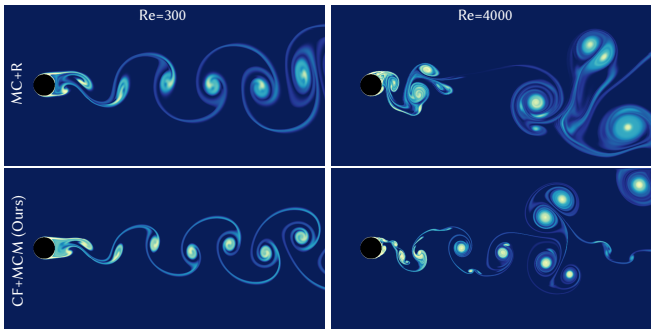


Fig. 15. Ground truth comparisons using the von Kármán vortex street experiment. Here, we show our method (CF+MCM) is consistent with ground truth computed using a established method (MC+R) both at low and high Reynolds numbers. Note that as the Reynolds number increases, symmetry breaks and there is no longer a repeating pattern.

6.3.3 3D Delta Wing (Fig. 17). We setup a delta wing with a 20° angle of attack, 70° sweep angle, and a thickness of 0.125 m [Lyu et al. 2021]. The background flow velocity is set to 1 m/s. Due to

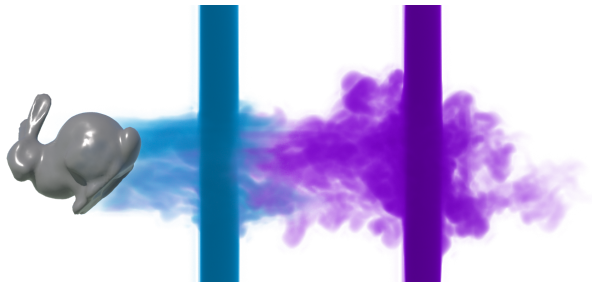


Fig. 16. A moving obstacle in the shape of Stanford bunny passing through smoke walls, generating intricate vortical wakes.

the design of the delta wing, vortices start to nucleate along the sides of the triangle and roll up to create lift. Fig. 17 shows our method qualitatively capturing the vortex dynamics around the delta wing, similar to the physical experiments. Note that an accurate aerodynamics simulation requires a more careful treatment of the solid-fluid interaction and boundary layer modeling, which is omitted in this paper.

6.3.4 3D Bunny Meteor (Fig. 2). This example demonstrates turbulent wakes behind an obstacle with irregular shape, *e.g.* a Stanford bunny. The speed of the background flow is 1 $^{m}/_{s}$. We evaluate the qualitative behavior of different solvers by the amount of vortical structures they generate. Our method outperforms the state-of-the-art in creating more vortices against the obstacle (Fig. 2).

7 CONCLUSION

We introduce a new advection-projection method for simulating incompressible fluids using a Lie advection of the velocity covector field. This method only requires an extra multiplication by the Jacobian transposed of the inverse flow map in the advection step. The simplicity of the modification makes our approach highly compatible with the previous fluid simulation framework. For example, the techniques of BFECC and MCM integrate seamlessly into the new method. Remarkably, the new covector advection emulates a vortex method that is capable of capturing intricate vortex dynamics. Our method also provides better energy preservation compared to previous methods. A number of examples of ink and smoke simulations with obstacles and buoyancy demonstrate that our method is applicable to realistic computer animations.

The conditional stability of our method is a minor drawback, in contrast with the unconditional stability in many of the previous fluid solvers (*e.g.* SF [Stam 1999], MC [Selle et al. 2008], MC+R [Zehnder et al. 2018]). While our method is stable as long as we set Δt based on an empirical CFL number similar to [Elcott et al. 2007], we do not have an analytical stability criterion. The main source of instability is the inclusion of the Jacobian of the flow maps, which can be sensitive to the flow configuration. Nevertheless, our method, which emulates a vorticity solver, is still much more stable than most vortex methods because our method does not require special treatment regarding the vortex stretching.

While our work has only explored Covector Fluids as an advection-projection method on an Eulerian grid, the insights brought by this paper are general. Many other fluid computational and analytical paradigms, previously formulated with (1), can shift to covector-based counterparts with only simple modifications such as (3). For example, studying kinetic models (lattice Boltzmann methods) [Li et al. 2018, 2020] with velocity covectors may be fruitful. It is exciting to see whether such a combination with Covector Fluids will better capture the vorticity aspect of fluids.

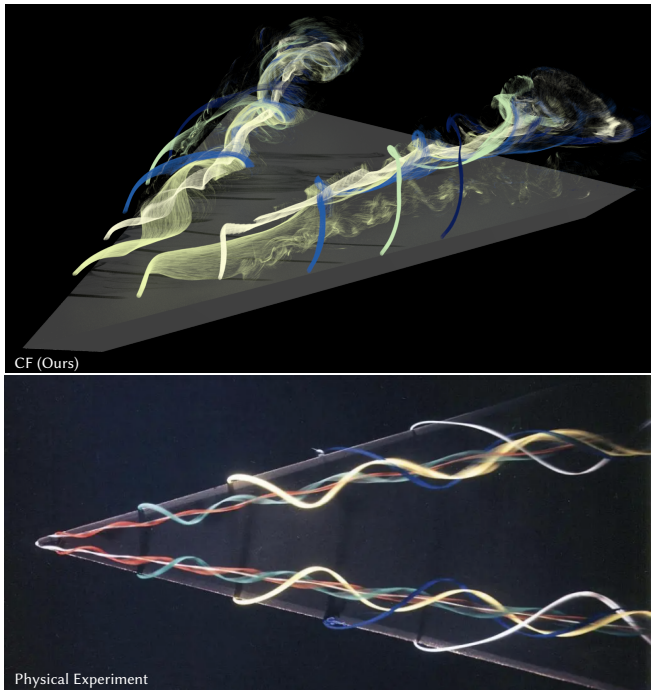


Fig. 17. A delta wing obstacle generating smoke against a laminar flow, with the velocity field visualized by colors. Top: Simulation results using our Covector Fluids (CF) method. Bottom: Physical experiment results by Henri Werlé at the Onera Hydrodynamics Visualization Laboratory [Délery 2011].

ACKNOWLEDGMENTS

This work was funded in part by the Ronald L. Graham Chair and the UC San Diego Center for Visual Computing. Additional support was provided by SideFX Software and Activision Blizzard.

REFERENCES

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.

Vladimir Arnold. 1966. Sur la géométrie différentielle des groupes de Lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits. In *Annales de l'institut Fourier*, Vol. 16. 319–361.

Vladimir I. Arnold and Boris A. Khesin. 1998. *Topological Methods in Hydrodynamics*. Springer.

Omri Azencot, Steffen Weißmann, Maks Ovsjanikov, Max Wardetzky, and Mirela Ben-Chen. 2014. Functional fluids on surfaces. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 237–246.

Robert D Blevins. 1990. Flow-induced vibration. *New York* (1990).

Robert Bridson. 2015. *Fluid simulation for computer graphics* (2 ed.). CRC press.

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*. Citeseer, 87–95.

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.

Augustin-Louis Cauchy. 1815. Théorie de la Propagation des Ondes a la Surface d'un Fluide Pesant d'une Profondeur Indéfinie. In *Oeuvres Complètes d'Augustin Cauchy*. Vol. 1. Imprimerie Royale. Presented to the French Academy in 1815 (published in 1827).

Albert Chern. 2017. *Fluid dynamics with incompressible Schrödinger flow*. Ph.D. Dissertation. California Institute of Technology.

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.

Norishige Chiba, Kazunobu Muraoka, Hiromichi Takahashi, and Mamoru Miura. 1994. Two-dimensional visual simulation of flames, smoke and the spread of fire. *The Journal of Visualization and Computer Animation* 5, 1 (1994), 37–53.

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

Alexandre Joel Chorin. 1968. Numerical solution of the Navier-Stokes equations. *Mathematics of computation* 22, 104 (1968), 745–762.

Alexandre Joel Chorin. 1990. Hairpin removal in vortex interactions. *J. Comput. Phys.* 91, 1 (1990), 1–21.

Alexandre Joel Chorin and Jerrold E Marsden. 1990. *A mathematical introduction to fluid mechanics*. Vol. 168. Springer.

A. Clebsch. 1859. Ueber die Integration der hydrodynamischen Gleichungen. *Journal für die reine und angewandte Mathematik* 56 (1859), 1–10. English translation by D. H. Delphenich, http://www.neo-classical-physics.info/uploads/3/4/3/6/34363841/clebsch_-_clebsch_variables.pdf.

Ricardo Cortez. 1995. Impulse-based methods for fluid flow. https://doi.org/10.2172/87798

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

Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable laplacian eigenfluids. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Tyler De Witt, Christian Lessig, and Eugene Fiume. 2012. Fluid simulation using laplacian eigenfunctions. *ACM Transactions on Graphics (TOG)* 31, 1 (2012), 1–11.

Jean Délery. 2011. Separation in three-dimensional steady flow; Part 3: Topology of Some Remarkable Three-Dimensional Flows. , 17–17 pages. https://www.onera.fr/sites/default/files/ressources_documentaires/cours-exposes-conf/onera-3d-separation-jean-delery-2011-2.pdf

Jim Douglas, Jr and Thomas F Russell. 1982. Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures. *SIAM J. Numer. Anal.* 19, 5 (1982), 871–885.

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

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

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.

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).

Nick Foster and Dimitris Metaxas. 1997. Modeling the motion of a hot, turbulent gas. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 181–188.

Uriel Frisch and Barbara Villone. 2014. Cauchy's almost forgotten Lagrangian formulation of the Euler equation for 3D incompressible flow. *The European Physical Journal H* 39, 3 (2014), 325–351.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–12.

Eitan Grinspun, Mathieu Desbrun, Konrad Polthier, Peter Schröder, and Ari Stern. 2006. Discrete differential geometry: an applied introduction. *ACM Siggraph Course* 7, 1 (2006).

Hermann Hankel. 1861. *Zur allgemeinen Theorie der Bewegung der Flüssigkeiten*. Dieterichsche Univ.-Buchdruckerei, Göttingen. For an English translation see [Villone and Rampf 2017].

Francis H Harlow and J Eddie Welch. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids* 8, 12 (1965), 2182–2189.

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

Darryl D Holm, Boris A Kupershmidt, and C David Levermore. 1983. Canonical maps between Poisson brackets in Eulerian and Lagrangian descriptions of continuum mechanics. *Physics Letters A* 98, 8-9 (1983), 389–395.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jaroslaw R Rossignac. 2005. *Flowfixer: Using BFECC for fluid simulation*. Technical Report. Georgia Institute of Technology.

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

JL Lagrange. 1788. *Méchanique Analitique*. A Paris, Chez La Veuve Desaint.

Randall J LeVeque. 2002. *Finite volume methods for hyperbolic problems*. Vol. 31. Cambridge university press.

Wei Li, Kai Bai, and Xiaopei Liu. 2018. Continuous-scale kinetic fluid simulation. *IEEE transactions on visualization and computer graphics* 25, 9 (2018), 2694–2709.

Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and scalable turbulent flow simulation with two-way coupling. *ACM Transactions on Graphics* 39, 4 (2020), Art–No.

Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-reduced variational fluid simulation. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–12.

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.

Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and versatile fluid-solid coupling for turbulent flow simulation. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–18.

Jerrold Marsden and Alan Weinstein. 1983. Coadjoint Orbits, Vortices, and Clebsch Variables for Incompressible Fluids. *Physica D: Nonlinear Phenomena* 7, 1 (1983), 305–323.

Alexander George McKenzie. 2007. *HOLA: a High-Order Lie Advection of Discrete Differential Forms With Applications in Fluid Dynamics*. Master's thesis. California Institute of Technology.

Philip J Morrison. 1998. Hamiltonian description of the ideal fluid. *Reviews of modern physics* 70, 2 (1998), 467.

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-preserving integrators for fluid animation. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–8.

Patrick Mullen, Alexander McKenzie, Dmitry Pavlov, Luke Durant, Yiying Tong, Eva Kanso, Jerrold E Marsden, and Mathieu Desbrun. 2011. Discrete Lie advection of differential forms. *Foundations of Computational Mathematics* 11, 2 (2011), 131–149.

David Mumford and Peter W Michor. 2012. On Euler's equation and 'EPDiff'. *arXiv preprint arXiv:1209.6576* (2012).

Rahul Narain, Jonas Zehnder, and Bernhard Thomaszewski. 2019. A second-order advection-reflection solver. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 2 (2019), 1–14.

Tristan Needham. 2021. *Visual Differential Geometry and Forms: A Mathematical Drama in Five Acts*. Princeton University Press.

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

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.

Dmitry Pavlov, Patrick Mullen, Yiying Tong, Eva Kanso, Jerrold E Marsden, and Mathieu Desbrun. 2011. Structure-preserving discretization of incompressible fluids. *Physica D: Nonlinear Phenomena* 240, 6 (2011), 443–458.

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.

Miodrag Rančić and Gordana Sindjić. 1989. Noninterpolating semi-Lagrangian advection scheme with minimized dissipation and dispersion errors. *Monthly weather review* 117, 8 (1989), 1906–1911.

Giovanni Russo and Peter Smereka. 1999. Impulse formulation of the Euler equations: general properties and numerical methods. *Journal of Fluid Mechanics* 391 (1999),

189–209.
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), 223–230.
John Stanley Sawyer. 1963. A semi-Lagrangian method of solving the vorticity advection equation. *Tellus* 15, 4 (1963), 336–342.
Robert Saye. 2016. Interfacial gauge methods for incompressible fluid dynamics. *Science advances* 2, 6 (2016), e1501869.
Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35, 2 (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.
Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 121–128.
Andrew Staniforth and Jean Côté. 1991. Semi-Lagrangian integration schemes for atmospheric models—A review. *Monthly weather review* 119, 9 (1991), 2206–2223.
John Steinhoff and David Underhill. 1994. Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids* 6, 8 (1994), 2738–2744.
Gilbert Strang. 1968. On the construction and comparison of difference schemes. *SIAM journal on numerical analysis* 5, 3 (1968), 506–517.
Roger Temam. 1969. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II). *Archive for rational mechanics and analysis* 33, 5 (1969), 377–385.
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.*
William Thomson. 1868. On Vortex Motion. *Earth and Environmental Science Transactions of the Royal Society of Edinburgh* 25, 1 (1868), 217–260.
Barbara Villone and Cornelius Rampf. 2017. Hermann Hankel's "On the general theory of motion of fluids". *The European Physical Journal H* 42, 4 (2017), 557–609.
Heinrich Martin Weber. 1868. Ueber eine Transformation der hydrodynamischen Gleichungen. *Journal für die reine und angewandte Mathematik* 68 (1868), 286–292.
Steffen Weißmann and Ulrich Pinkall. 2009. Real-time Interactive Simulation of Smoke Using Discrete Integrable Vortex Filaments. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2009)*, Hartmut Prautzsch, Alfred Schmitt, Jan Bender, and Matthias Teschner (Eds.). The Eurographics Association. https://doi.org/10.2312/PE/vriphys/vriphys09/001-010
Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers*. 1–12.
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), 98:1–98:11.
Larry Yaeger, Craig Upson, and Robert Myers. 1986. Combining physical and visual simulation—creation of the planet jupiter for the film "2010". *Acm Siggraph Computer Graphics* 20, 4 (1986), 85–93.
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.
Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–8.
Xinxin Zhang 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.
Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.

A DERIVATIONS

This appendix derives equations (14), (30) and (31).[7]

[7]The derivations involve *vector-valued differential forms* $\mathbf{A} \in \Omega^k(M;TM) = \Gamma((\wedge^k T^*M) \otimes TM)$. For example, the differential $\mathbf{A} = \nabla\mathbf{v}$ of a vector field $\mathbf{v} \in \mathfrak{X}(M) = \Omega^0(M;TM)$ is a vector-valued 1-form $\mathbf{A} \in \Omega^1(M;TM)$, $\mathbf{A}[\![\mathbf{a}]\!] = \nabla_{\mathbf{a}}\mathbf{v}$ for each vector $\mathbf{a}$. Whenever a vector field or vector-valued form is taken derivative, we use the *covariant derivative* (Levi-Civita connection) $\nabla\colon \Omega^0(M;TM) \to \Omega^1(M;TM)$ and the *exterior covariant derivative* $d^\nabla\colon \Omega^k(M;TM) \to \Omega^{k+1}(M;TM)$. We will use the *interior product* $i_{\mathbf{v}}\colon \Omega^k(M) \to \Omega^{k-1}(M)$ as well as the *identity* vector-valued 1-form $\mathbf{I} \in \Omega^1(M;TM)$, $i_{\mathbf{v}}\mathbf{I} = \mathbf{I}[\![\mathbf{v}]\!] = \mathbf{v}$ for all vector $\mathbf{v}$. Note that for each $\mathbf{v}, \mathbf{w} \in \mathfrak{X}(M)$, $\alpha \in \Omega^k(M)$, $\beta \in \Omega^\ell(M)$ $\mathbf{A} \in \Omega^k(M;TM)$, $\mathbf{B} \in \Omega^\ell(M;TM)$,

- $\mathbf{v}^\flat = \langle \mathbf{v}, \mathbf{I}\rangle$, ($\flat$ in terms of $\mathbf{I}$) (42a)
- $d\langle \mathbf{A} \wedge \mathbf{B}\rangle = \langle d^\nabla \mathbf{A} \wedge \mathbf{B}\rangle + (-1)^k \langle \mathbf{A} \wedge d^\nabla \mathbf{B}\rangle$, (Leibniz rule, metric) (42b)
- $i_{\mathbf{v}}(\alpha \wedge \beta) = (i_{\mathbf{v}}\alpha) \wedge \beta + (-1)^k \alpha \wedge i_{\mathbf{v}}\beta$, (Leibniz rule for $i_{\mathbf{v}}$) (42c)
- $d^\nabla \mathbf{I} = 0$; or $[\mathbf{v}, \mathbf{w}] = \nabla_{\mathbf{v}}\mathbf{w} - \nabla_{\mathbf{w}}\mathbf{v}$, (Torsion-free) (42d)
- $d^\nabla d^\nabla = 0$, (Curvature-free on flat spaces) (42e)
- $\mathscr{L}_{\mathbf{v}}\alpha = di_{\mathbf{v}}\alpha + i_{\mathbf{v}}d\alpha$, (Cartan's formula) (42f)
- $\mathscr{L}_{\mathbf{v}}(i_{\mathbf{w}}\alpha) = i_{[\mathbf{v},\mathbf{w}]}\alpha + i_{\mathbf{w}}(\mathscr{L}_{\mathbf{v}}\alpha)$, (Lie derivative on contraction) (42g)
- $d\,\mathscr{L}_{\mathbf{v}}\alpha = \mathscr{L}_{\mathbf{v}}\,d\alpha$, (Commutativity between d and $\mathscr{L}_{\mathbf{v}}$). (42h)

Here $[\mathbf{v}, \mathbf{w}] \in \mathfrak{X}$ denotes the *Lie bracket* of vector fields $\mathbf{v}, \mathbf{w}$.

A.1 Derivation of (14)

Here we show that $\mathscr{L}_{\mathbf{v}}(\mathbf{a}^\flat) = (\nabla_{\mathbf{v}}\mathbf{a})^\flat + \langle \nabla\mathbf{v}, \mathbf{a}\rangle$:

$$\begin{aligned}\mathscr{L}_{\mathbf{v}}\,\mathbf{a}^\flat &\overset{(42a,f)}{=} (di_{\mathbf{v}} + i_{\mathbf{v}}d)\langle \mathbf{a}, \mathbf{I}\rangle \overset{(42b)}{=} d\langle \mathbf{a}, \mathbf{v}\rangle + i_{\mathbf{v}}(\langle \nabla\mathbf{a} \wedge \mathbf{I}\rangle + \langle \mathbf{a}, \underbrace{d^\nabla \mathbf{I}}_{=0\,(42d)}\rangle)\\ &\overset{(42b,c)}{=} \langle \nabla\mathbf{a}, \mathbf{v}\rangle + \langle \mathbf{a}, \nabla\mathbf{v}\rangle + \langle \nabla_{\mathbf{v}}\mathbf{a}, \mathbf{I}\rangle - \langle \nabla\mathbf{a}, \mathbf{v}\rangle\\ &\overset{(42a)}{=} \langle \nabla\mathbf{v}, \mathbf{a}\rangle + (\nabla_{\mathbf{v}}\mathbf{a})^\flat.\end{aligned}$$

A.2 Derivation of (31)

Here we show that if $\mathbf{u}$ satisfies (29) under a divergence-free $\mathbf{v}$, then $\frac{\partial}{\partial t}\mathbf{w} + \nabla_{\mathbf{v}}\mathbf{w} - \nabla_{\mathbf{w}}\mathbf{v} = \mathbf{0}$ (eq. (31)) holds for $\mathbf{w} = \operatorname{curl}\mathbf{u}$. By (14), eq. (29) implies that $\eta = \mathbf{u}^\flat$ satisfies $\frac{\partial}{\partial t}\eta + \mathscr{L}_{\mathbf{v}}\eta = 0$ (eq. (28)). Applying d to (28) and using (42h), we obtain $\frac{\partial}{\partial t}\omega + \mathscr{L}_{\mathbf{v}}\omega = 0$ for the *vorticity 2-form* $\omega = d\eta$. In 3D, the relationship between the 2-form $\omega = d\mathbf{u}^\flat$ and the vector field $\mathbf{w} = \operatorname{curl}\mathbf{u}$ is given by $\omega = i_{\mathbf{w}}\mu$ where $\mu \in \Omega^3(M)$ is the volume form. Using this relationship we obtain

$$\begin{aligned}0 = \frac{\partial}{\partial t}\omega + \mathscr{L}_{\mathbf{v}}\omega &= \frac{\partial}{\partial t}(i_{\mathbf{w}}\mu) + \mathscr{L}_{\mathbf{v}}(i_{\mathbf{w}}\mu)\\ &\overset{(42g)}{=} i_{\partial\mathbf{w}/\partial t}\mu + i_{[\mathbf{v},\mathbf{w}]}\mu + i_{\mathbf{w}}\underbrace{\mathscr{L}_{\mathbf{v}}\mu}_{=0} \quad (\mathscr{L}_{\mathbf{v}}\mu = 0 \text{ since } \operatorname{div}\mathbf{v} = 0)\\ &= i_{(\partial\mathbf{w}/\partial t + [\mathbf{v},\mathbf{w}])}\mu.\end{aligned}$$

Therefore, $\frac{\partial}{\partial t}\mathbf{w} + [\mathbf{v}, \mathbf{w}] = \mathbf{0}$. Finally, we arrive at (31) by substituting $[\mathbf{v}, \mathbf{w}] = \nabla_{\mathbf{v}}\mathbf{w} - \nabla_{\mathbf{w}}\mathbf{v}$ (eq. (42d)).

A.3 Derivation of (30)

We show that if $\mathbf{v}$ is divergence-free and $\mathbf{u}$ satisfies $\frac{\partial}{\partial t}\mathbf{u} + \nabla_{\mathbf{v}}\mathbf{u} = \mathbf{0}$, then $\mathbf{w} = \operatorname{curl}\mathbf{u}$ satisfies (30). Differing from Appendix A.2 by the term $\langle \nabla\mathbf{v}, \mathbf{u}\rangle$, the 1-form $\eta = \mathbf{u}^\flat$ satisfies $\frac{\partial}{\partial t}\eta + \mathscr{L}_{\mathbf{v}}\eta = \langle \nabla\mathbf{v}, \mathbf{u}\rangle$. Taking d on both sides of the equation yields

$$\begin{aligned}\frac{\partial}{\partial t}\omega + \mathscr{L}_{\mathbf{v}}\omega = d\langle \nabla\mathbf{v}, \mathbf{u}\rangle &\overset{(42b)}{=} \langle \underbrace{d^\nabla \nabla\mathbf{v}}_{=0\,(42e)}, \mathbf{u}\rangle - \langle \nabla\mathbf{v} \wedge \nabla\mathbf{u}\rangle\\ &= \langle \nabla\mathbf{u} \wedge \nabla\mathbf{v}\rangle.\end{aligned}$$

Converting 2-forms to vector fields in 3D using $\omega = i_{\mathbf{w}}\mu$, we obtain

$$\frac{\partial}{\partial t}\mathbf{w} + \nabla_{\mathbf{v}}\mathbf{w} - \nabla_{\mathbf{w}}\mathbf{v} = \langle \nabla\mathbf{u} \times \nabla\mathbf{v}\rangle.$$