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.

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

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). © 2022 Copyright held by the owner/author(s).

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 \( \mathbf{u} : M \subset \mathbb{R}^n \rightarrow \mathbb{R}^n \) (\( n = 2 \) or \( 3 \)) of such a fluid with domain \( M \):

\[
\begin{aligned}
\frac{\partial}{\partial t} \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla p, \\
\nabla \cdot \mathbf{u} &= 0. 
\end{aligned}
\]

Here, \( p : M \rightarrow \mathbb{R} \) 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 \( \frac{\partial}{\partial t} \mathbf{u} + \mathbf{v} \cdot \nabla \mathbf{u} = 0 \) for \( \Delta t \) time; \( \triangleright \) advection
3. \( \mathbf{u} \leftarrow \mathbf{u} - \Delta t \nabla p \) so that \( \nabla \cdot \mathbf{u} = 0 \) afterwards. \( \triangleright \) projection

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

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

\[
\mathbf{u}(x) \leftarrow \mathbf{u}(\Psi(x)), \quad x \in M. 
\]

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

\[
\mathbf{u}(x) \leftarrow (d\Psi(x))^\top \mathbf{u}(\Psi(x)), \quad x \in M, 
\]
where $(d\Psi)^\top$ 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 $u$ so that its line integrals along any curve $C$ carried by the fluid are preserved:

$$
\int_C (d\Psi(x))^\top u(\Psi(x)) \cdot dl = \int_C u(\Psi(x)) \cdot d\Psi(x) dl = \int_{\Psi(C)} u \cdot dl. \quad (4)
$$

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

Equation (3) is a Lie advection of $u$ 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

![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.](image1)

![Fig. 3. (a) In a standard fluid solver, the velocity $u$ is transported by the flow $v$ using an inverse flow map $\Psi$. (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.](image2)
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


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 Sindjic 1989]. The method of characteristic mapping (MCM) [Tesendorf 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; Mullens 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 advectons 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 Mullens 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 Pindak 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

## Table 1. Method acronyms used throughout the paper.

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable fluids</td>
<td>[Stam 1999]</td>
<td>SF</td>
</tr>
<tr>
<td>Stable and circulation preserving fluids</td>
<td>[Elcott et al. 2007]</td>
<td>SCFF</td>
</tr>
<tr>
<td>MacCormack</td>
<td>[Selle et al. 2008]</td>
<td>MC</td>
</tr>
<tr>
<td>Integrated vorticity of convective kinematics</td>
<td>[Zhang et al. 2015]</td>
<td>IVOCK</td>
</tr>
<tr>
<td>Reflection</td>
<td>[Zehnder et al. 2018]</td>
<td>R</td>
</tr>
<tr>
<td>Method of characteristic mapping</td>
<td>[Sato et al. 2018]</td>
<td>MCM</td>
</tr>
<tr>
<td>Bi-directional mapping of convective quantities</td>
<td>[Qu et al. 2019]</td>
<td>BiMocq</td>
</tr>
<tr>
<td>Covector fluids</td>
<td>Our method</td>
<td>CF</td>
</tr>
</tbody>
</table>
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. Oseledepts [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 Oseledepts’ 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 $n = 2, 3$.³ At each point $x \in M$, the tangent space $T_x M$ is the space of tangent vectors based at $x$. For $M \subset \mathbb{R}^n$, $T_x M \cong \mathbb{R}^n$ using the Cartesian coordinates. The dual space of $T_x M$, the cotangent space $T^*_x M$.

---

³Although we focus on Euclidean spaces $\mathbb{R}^n$ in this paper, the theory covered in Section 3.1 applies to general Riemannian manifolds.
The differential $d\boldsymbol{\xi}$ of a covector field $\boldsymbol{\xi}$ at each point $x \in M$ is denoted by $\langle \xi, \cdot \rangle$ (see inset). A covector field is an assignment of a covector $\xi(x)$ at each point $x \in M$. We let $\mathfrak{X}(M)$ and $\mathfrak{X}^*(M)$ denote the space of all vector fields and covector fields on $M$ respectively.

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 $\int_C \xi f = \int_C \xi^\flat$ for short. Here, $d\xi$ represents an infinitesimal directed element of curve $C$.

### 3.2 Musical isomorphisms

When $T_xM$ is equipped with an inner product $\langle \cdot, \cdot \rangle$, each vector $a \in T_xM$ is uniquely associated with a covector $a^\flat \in T^*_xM$ so that $a^\flat \cdot = \langle a, \cdot \rangle$. Mirroring what $\flat$ (flat) and $\sharp$ (sharp) entail in the musical context, they are the mathematical inverse of each other: $\flat$ turns a vector $a \in T_xM$ into a covector $a^\flat$, so $\sharp$ turns a covector $\xi \in T^*_xM$ 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 $f : M \to \mathbb{R}$ is a covector field defined so that $(df)[v] = \nabla v f$ in the direction $v$. In fact, when there is an inner product structure, we have $(grad f)(v) = df(v)$ for all vectors $v$. 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 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 $\Psi : W \to M$. The pullback of a scalar function $g : M \to \mathbb{R}$ by $\Psi$ becomes a function on $W$ given by

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

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

$$(\Psi^\flat \xi)(w) = \xi(\Psi(w)) \cdot d\Psi(w) \quad \text{for all } w \in T_xW, x \in W. \tag{6}$$

Here, $d\Psi(w) : T_wW \to T_{\Psi(w)}M$ is the Jacobian matrix of $\Psi$ at $x$ (see inset). This definition ensures that for exact differentials

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

and that under the integration sign

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

for each oriented curve $C$ in $W$. The right-hand side of (6) can further be expressed in terms of the adjoint (matrix transpose) $d\Psi^\top(x) : T^*_yM \to T^*_xM$ as follows:

$$\xi(\Psi(x)) \cdot d\Psi^\top(x) \Psi_v = (d\Psi^\top(x) \xi(\Psi(x))) v. \tag{9}$$

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

$$\Psi^\flat \xi = d\Psi^\top(x)(\xi(\Psi(x))). \tag{10}$$

### 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 $\Phi_t : M \to W$, $t \in \mathbb{R}$. At each instance $t$, the time-derivative $L_t \Phi$ represents the velocity of the flow. Each scalar field $g : W \to \mathbb{R}$ and covector field $\xi \in \mathfrak{X}^*(W)$ renders a one-parameter family of pullback scalar fields $\Phi_t^* g$ and pullback covector fields $\Phi_t^\flat \xi$ 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 $\Phi_t = \frac{d\Phi_t}{dt}$ on $W$:

$$\frac{d}{dt}(\Phi_t^* g)(x) = \frac{d}{dt}(g(\Phi_t(x))) = (\Phi_t \cdot \nabla) g(\Phi_t(x)). \tag{11}$$

In terms of pullbacks, $\frac{d}{dt}(\Phi_t^* g) = \Phi_t^\flat (\Phi_t \cdot \nabla) g$. We define the quantity $(\Phi_t \cdot \nabla) g$ as the Lie derivative for the scalar field $g$.

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

$$\frac{d}{dt}(\Phi_t^\flat \xi)(x) = \frac{d\Phi_t}{dt}(\xi(\Phi_t(x))). \tag{12}$$

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

$$\frac{d}{dt} \int_{\Phi_t(C)} \xi = \int_{\Phi_t(C)} L_{\Phi_t} \xi. \tag{13}$$

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 $\xi = a^\flat$ along the vector field $v = \Phi$ is given by\(^3\text{(Appendix A.1)}\)

$$L_{v^\flat} \xi = v \cdot \nabla a + (\nabla v) \cdot a. \tag{14}$$

### 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

- A common notation is $\mathfrak{X}(M) = \Gamma(TM)$ and $\mathfrak{X}^*(M) = \Omega^1(M)$.
- In index notation, $(v \cdot \nabla a + (\nabla v) \cdot a)_i = v_j \partial_j a_i + a_j \partial_j v_i$.

ACM Trans. Graph., Vol. 41, No. 4, Article 113. Publication date: July 2022.
Let the inverse flow map $\Psi_t: M_t \rightarrow M_0$ be the inverse function of the flow map, $\Psi_t = \Phi_t^{-1}$. By taking the derivative of $\Psi_t \circ \Phi_t = \text{id}_M$ and (15), the equation satisfied by $\Psi_t$ is

$$\frac{\partial}{\partial t} \Psi_t = -d\Psi_t \| \nabla \|, \quad \Psi_0 = \text{id}_M .$$

(16)

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

$$\frac{\partial}{\partial t} q_t = \left( \frac{\partial}{\partial t} + v \cdot \nabla \right) q_t = 0,$$

(17)

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

$$\frac{\partial}{\partial t} q_t = 0 \iff q_t = \Psi_t^* q_0.$$

(18)

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

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

(19)

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

$$0 = \frac{\partial}{\partial t} (\Phi_t^* \xi_t) = \Phi_t^* \left( \frac{\partial}{\partial t} \xi_t \right) + \Phi_t^* (\nabla v \cdot \xi_t).$$

(20)

This relation is similarly summarized as

$$\left( \frac{\partial}{\partial t} + L v \right) \xi_t = 0 \iff \xi_t = \Psi_t^* \xi_0.$$

(21)

We call the equation $\left( \frac{\partial}{\partial t} + L v \right) \xi_t = 0$ the transport equation for the covector fields. We shall refer to $\left( \frac{\partial}{\partial t} + L v \right)$ 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 $\xi_t$ along any curve $C$ pushed by the flow $\Phi_t$,

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

(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 $a_t = \frac{d}{dt} \xi_t$, the covector transport equation reads (cf. (14))

$$\frac{d}{dt} a_t + v \cdot \nabla a_t + (\nabla v) \cdot a_t = 0.$$

(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 $\frac{d a_t}{d t} = -(\nabla v) \cdot a_t$. According to (21) and (10), (23) has an explicit integral solution

$$a_t(x) = d\Psi_t^* \partial_t a_0(\Psi_t(x)).$$

(24)

\[\text{In (12), we define Lie derivative for time-independent covector field } \xi \text{ and time-dependent forward flow map } \Phi_t. \text{ Adding } \frac{d}{dt} \Phi_t^* \xi_t \text{ to } \Phi_t^* \xi_t \text{ to get } \Phi_t^* \xi_t = \Phi_t^* \xi_t \text{ with the relation } t_1 = t_2 = t, \text{ we obtain the subsequent equation.}\]
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 \((\nabla u) \cdot u = \frac{1}{2} \nabla |u|^2\) to both sides of (1) yields

\[
\frac{d}{dt} \eta + \mathcal{L}_u \eta = -d\lambda, \quad \eta = \eta^b, \quad \nabla \cdot u = 0. \tag{26}
\]

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 \(u\) of every flowing particle. On the other hand, the left-hand side of (26) measures the rate of change of the line integral \(\int_C \eta = \int_C u \cdot dl\) along every curve \(C\) 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 \(\lambda\) 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 \(u\) by splitting (26) into the following three substeps:

1. Estimate a flow velocity, e.g. \(v \leftarrow u\); freeze flow velocity
2. Solve \((\frac{d}{dt} + \mathcal{L}_u)u^b = 0\) for \(\Delta t\) time; covector advection
3. \(u \leftarrow u - \Delta t \nabla \lambda\) so that \(\nabla \cdot u = 0\) 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),

\[
\Psi(x) \leftarrow (d\Psi(x))^{-1} \Psi(x), \quad x \in M. \tag{27}
\]

where \(\Psi\) is the inverse flow map generated by velocity \(v\) for a time step of \(\Delta t\) (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 \(a\) to point \(b\), then \(\int_C d\lambda = \lambda(b) - \lambda(a)\).

---

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

\[
\left(\frac{d}{dt} + \mathcal{L}_v\right) \eta = 0, \quad \eta = u^b, \tag{28}
\]

or (Remark 1)

\[
\frac{d}{dt} u + v \cdot \nabla u + (\nabla v) \cdot u = 0, \tag{29}
\]

over the traditional advection \((\frac{d}{dt} + v \cdot \nabla)u = 0\).

### 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 \(u\) with a pure gradient which leads to no change in its curl.

Under the classical advection equation \((\frac{d}{dt} + v \cdot \nabla)u = 0\), the vorticity \(w = \nabla \times u\) evolves according to\(^6\) (Appendix A.3)

\[
\frac{d}{dt} w + v \cdot \nabla w - w \cdot \nabla v = (\nabla u \times \nabla v). \tag{30}
\]

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

\[
\frac{d}{dt} w + v \cdot \nabla w - w \cdot \nabla v = 0 \tag{31}
\]

which is the correct vorticity equation. By advancing \(u\) 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 \((\nabla u \times \nabla v)_{ij} = \epsilon_{ijk} \partial_i u^l \partial_j v^l\).
Consider the equivalence classes of $X^r(M)$ where $[ξ] = [η]$ whenever $ξ - η = dϕ$ 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 $[η] ∈ X^r(M)/im(d)$.

Now consider two covector fields $ξ_0, η_0 ∈ X^r(M)$, divergence-free or not, and transport them by the covector advection equation (28) to obtain $ξ_t, η_t$ respectively. Then
\[
[ξ_0] = [η_0] \quad \text{if and only if} \quad [ξ_t] = [η_t].
\] (32)
To see this assertion, use (21) to express $ξ_t - η_t$ as the pullback of $ξ_0 - η_0$ by the inverse flow map
\[
ξ_t - η_t = Ψ_t^* (ξ_0 - η_0)
\] (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 in all 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 $Ψ_t : M_t → M_0$ be the inverse flow map (Eulerian-to-Lagrangian map) which is the Lagrangian marker carried by the history of the solution
\[
\frac{d}{dt} Ψ_t + u_t \cdot \nabla Ψ_t = 0, \quad Ψ_0 = \text{id}_M.
\] (34)
Then the velocity covector field at the current time $t$ is the pressure projection of
\[
η_t = Ψ_t^* η_0.
\] (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 $u_t$ by (1) integrating over time along particle trajectories:
\[
u_t(x) = \nu_0(Ψ_t(x)) + \int_0^t (\nabla ρ_τ (Ψ_t(Ψ_t(x))) \ dt).\] (36)
In CF+MCM, the time integration of (26) yields
\[
η_t = Ψ_t^* η_0 + \int_0^t (Φ_t ⊗ Ψ_t)^* dτ_t, \quad \text{d}τ_t \rightarrow \text{d}τ,
\] (37)
where $dτ_t$ denotes the amount of interpolation in the advection-projection iteration.

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 $\int_0^t (\nabla ρ_t) ⊗ Φ_t ⊗ Ψ_t$ 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 $A(q, v, Δt)$ denote an advection solver that solves $\frac{d}{dt} q + v \cdot \nabla q = 0$ for a $Δt$ timestep for a generic field $q$. Let $Φ_{cov}(u; v, Δt)$ denote a Lie transportation $(\frac{d}{dt} + Φ_{cov}(u; v, Δt))^u = 0$ of a covector field $u^b$. We detail the algorithm of the advection step in Alg. 3 and Alg. 4. Let $Φ : X(M) → X(M)$ be the pressure projection operator.

### 5.1 Base method

Our base method is given by

**Algorithm 1 Covector Fluids (1st order)**

**Input:** Initial velocity $u$; step size $Δt$;

1. **for each** time step do
2. $v ← u$; $\triangleright$ freeze flow velocity
3. $u ← Φ_{cov}(u; v, Δt)$; $\triangleright$ covector Lie advection
4. $u ← Φ(u)$; $\triangleright$ 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):
As we described in Section 4.6, the covector advection integrates (27). A direct application of (27) yields the following semi-Lagrangian (sL) method for the covector advection \( s_{\text{covec}}(\mathbf{u}, v, \Delta t) \).

\[
\begin{align*}
\text{Input:} & \quad \text{Initial velocity } \mathbf{u}, \text{ time step } \Delta t; \\
\text{for each time step do} & \\
& \quad v \leftarrow \mathcal{P}(s_{\text{covec}}(\mathbf{u}, v, \Delta t)); \quad \triangleright \text{ estimate flow velocity at } \Delta t \\
& \quad u \leftarrow \mathcal{P}(s_{\text{covec}}(u, v, \Delta t)); \quad \triangleright \text{ full step} \\
\text{end for}
\end{align*}
\]

Output: \( u \)

5.2 Covector advection

The core covector advection solver is based on (27). A direct application of (27) yields the following covector advection (covec) (Algorithm 4).

\[
\begin{align*}
\text{Input:} & \quad \text{Field to advect } u, \text{ flow velocity } v, \text{ time span } \Delta t \\
\text{for each point } x \in M & \quad \triangleright \text{ construct inverse flow map} \\
\Psi(x) & \leftarrow \text{RK4 backtrace from } x \text{ with flow velocity } v; \quad \triangleright \text{ Pullback (Section 5.4.1)} \\
\text{end for} \\
\text{do} & \\
& \quad u \leftarrow d\Psi^T(u \circ \Psi); \quad \triangleright \text{ Pullback (Section 5.4.1)} \\
\text{end do}
\end{align*}
\]

Output: \( u \)

Our covector advection routine \( s_{\text{covec}} \) 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.

\[
\begin{align*}
\text{Input:} & \quad \text{field to advect } u, \text{ flow velocity } v, \text{ time span } \Delta t \\
\text{do} & \\
& \quad u_1 \leftarrow s_{\text{covec}}^\text{sL}(u, v, \Delta t); \quad \triangleright \text{ back-and-forth advection} \\
& \quad u_0 \leftarrow s_{\text{covec}}(u_1; v, -\Delta t); \quad \triangleright \text{ roundtrip error} \\
& \quad u \leftarrow u_0 - u_1; \quad \triangleright \text{ error correction} \\
\text{end do}
\end{align*}
\]

Output: \( u \)

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 BiMoq [Qu et al. 2019] (CF+MCM) by delaying the-reinitialization of the Lagrangian Marker \( \Psi \). This largely reduces the amount of interpolation during subsequent advection steps, by only needing to advect the map \( \Psi \) and not the velocity components. The calculation of \( d\Psi^T(u \circ \Psi) \) can utilize the values from this map, where \( u \) 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. \( \Phi \circ \Psi \neq \Psi \circ \Phi \)).

\[
\begin{align*}
\text{Input:} & \quad \text{Initial velocity } u, \text{ time step } \Delta t; \\
\text{for each time step do} & \\
& \quad u_0 \leftarrow u; \quad \triangleright \text{ backward map } \Psi \leftarrow \text{id}; \quad \triangleright \text{ forward map } \Phi \leftarrow \text{id} \\
& \quad \Phi \leftarrow \text{ESTIMATEVELOCITY}(u, \Phi, \Delta t); \quad \triangleright \text{ mid-point velocity} \\
& \quad u_1 \leftarrow d\Phi^T(u_0 \circ \Phi); \quad \triangleright \text{ advect inverse flow map} \\
& \quad u_0 \leftarrow d\Psi^T(u_1 \circ \Psi); \quad \triangleright \text{ back-and-forth transport} \\
& \quad u \leftarrow u_0 - u_1; \quad \triangleright \text{ roundtrip error} \\
& \quad u \leftarrow u_1 - d\Psi^T(u_0 \circ \Psi); \quad \triangleright \text{ error correction} \\
& \quad \Psi \leftarrow \text{Pressure projection} \\
& \quad \text{if reinitialization condition then} \\
& \quad \Phi, \Psi \leftarrow \text{id}; \quad \triangleright \text{ reset flow maps} \\
& \quad u_0 \leftarrow \Phi \circ u; \quad \triangleright \text{ reset velocity} \\
\text{end if} \quad \text{end if} \\
\text{end for}
\end{align*}
\]

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 \( v = u \).

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 \( u \circ \Psi \) by tracing back face centers and the Jacobian \( d\Psi^T \) 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 \( u \leftarrow d\Psi^T(u \circ \Psi) \) is computed on the face center \( F_{ij} \) with \( e_1 \) normal located between cell \( C_i \) and \( C_j \):

\[
\begin{align*}
\frac{\partial \Psi}{\partial x} & \leftarrow \left| \frac{\partial \Psi}{\partial x} \left| F_{ij} \right| \frac{\partial \Psi}{\partial x} \right|_{F_{ij}} \left| \frac{\partial \Psi}{\partial x} \right|_{C_i} \left| \frac{\partial \Psi}{\partial x} \right|_{C_j}, \quad x \in \{x, y, z\}.
\end{align*}
\]

5.4.2 BFECC Limiter. The BFECC technique requires an extremum (minmod) limiter [Selle et al. 2008; Qu et al. 2019] to reduce the oscillatory dispersion. When using Alg. 4, the output value \( u \) is clamped to the minimum and maximum value of \( u_1 \) componentwise within the immediate neighboring stencils.

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

\[
C = \frac{U \Delta t_{\text{crit}}}{\Delta x}.
\]

We use \( C \) to compare our conditional stability against various vorticity methods: IVOCK [Zhang et al. 2015] and SCPF [Elcott et al. 2018].
5.4.4 Buoyancy. For gravity or buoyancy force, we apply the Boussinesq model [Qu et al. 2019] \( f = c \rho g \) where is \( c \) is a constant, \( g \) the gravity, and \( \rho \) an advec ted scalar field representing density or temperature. We update the velocities by \( \mathbf{u} \leftarrow \mathbf{u} + \Delta t f \) 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 Dirichelet 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 SCFP [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 CPU-parallelism on a laptop with 2.3GHz 8-Core Intel Core i9 RK4 (2D BiMocq and all CF+MCM) for mapping advection.

due to the 2nd order velocity estimate. This is a relatively small computational overhead for our improvement in capturing vortex dynamics.

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 \( \omega(x) = U/\alpha(2 - r^2/a^2) \exp(0.5(1 - r^2/a^2)) \), where \( r \) is the distance from \( x \) to the vortex center, \( a = 0.3 \text{ m} \) is the core size, and \( U = 1 \text{ m/s} \) 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 SCFP) that target only the splitting error fail to preserve the energy because they still suffer from sampling error. Our method uses covector advection and

\begin{table}[h]
\centering
\caption{Performance and statistics.}
\begin{tabular}{|c|c|c|c|c|}
\hline
Figure & Domain size & Grid resolution & \( \Delta t \) & Comp. time/step \\
\hline
Ink jet (Fig. 1) & 5 x 10 x 5 m\(^3\) & 128 x 256 x 128 & 1/8s & 12.2 s / 9.3 s \\
Bunny meteor (Fig. 2) & 10 x 5 x 5 m\(^3\) & 256 x 128 x 128 & 1/16s & 12.7 s / 10.8 s \\
Von Kármán vortex street (Fig. 4) & 2 x 1 x 1 m\(^3\) & 512 x 256 & 1/32s & 1.3 s / 1.5 s \\
Covector transport (Fig. 5) & 1 x 1 m\(^3\) & 200 x 200 & 1/64s & 21 ms / 15 ms \\
Leapfrogging pairs (Fig. 7) & 2 x 2 x 2 m\(^3\) & 256 x 256 & 1/128s & 0.3 s / 0.5 s \\
Taylor vortices (Fig. 8) & 2 x 2 x 2 m\(^3\) & 256 x 256 & 1/256s & 0.2 s / 0.2 s \\
Trefiol knot (Fig. 9) & 10 x 5 x 5 m\(^3\) & 256 x 128 x 128 & 1/512s & 8.0 s / 4.8 s \\
Leapfrogging rings (Fig. 10) & 10 x 5 x 5 m\(^3\) & 256 x 128 x 128 & 1/1024s & 6.4 s / 4.8 s \\
Ink drop (Fig. 11) & 0.2 x 0.2 x 0.2 m\(^3\) & 512 x 512 & 1/2048s & 1.4 s / 1.2 s \\
SIGGRAPH ink drop (Fig. 12) & 0.2 x 0.2 x 0.2 m\(^3\) & 512 x 512 & 1/4096s & 1.2 s / 1.0 s \\
Pyroclastic cloud (Fig. 13) & 5 x 10 x 5 m\(^3\) & 128 x 256 x 128 & 1/1024s & 11.9 s / 10.7 s \\
Smoke plume (Fig. 14) & 5 x 10 x 5 m\(^3\) & 128 x 256 x 128 & 1/2048s & 11.3 s / 9.7 s \\
Ground truth comp. (Fig. 15) & 2 x 2 x 2 m\(^3\) & 1024 x 512 & 1/4096s & 2.5 s / 2.1 s \\
Moving obstacle (Fig. 16) & 10 x 5 x 5 m\(^3\) & 256 x 128 x 128 & 1/8096s & 9.2 s / 5.1 s \\
Delta wing (Fig. 17) & 5 x 5 x 5 m\(^3\) & 128 x 128 x 128 & 1/32768s & 6.7 s / 5.9 s \\
\hline
\end{tabular}
\end{table}
Fig. 9. Trefoil Knot evolution simulated under different methods. Our methods capture the vortex separation 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).

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

\[
\begin{align*}
x(\tau) &= \sin(\tau)+2\sin(2\tau) \\
y(\tau) &= \cos(\tau)-2\cos(2\tau) \\
z(\tau) &= -\sin(3\tau),
\end{align*}
\]

\[\tau \in [0, 2\pi),\]

to set up a knotted vortex with circulation \(1 \text{ m}^2/\text{s}\). 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.

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/s², 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/s².

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
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 exper-
iment, 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).

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
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 vortex 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.

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

Uriel Frisch and Barbara Villone. 2014.
Nick Foster and Dimitris Metaxas. 1997.
Alexandre Joel Chorin and Jerrold E Marsden. 1990.
Francis H Harlow and J E Welch. 1965.
Alexandre Joel Chorin. 1968.
Clebsch-Gordan coefficients for level set methods and incompressible flow.
ACM Transactions on Graphics (TOG) 40, 6 (2021), 1–18.
Hermann Hankel. 1861.
Hermann Hankel. 1869.
Hermann Hankel. 1869.
J.L Lagrange. 1788.
Giovanni Russo and Peter Smereka. 1999.
Alexandre Joel Chorin and Jerrold E Marsden. 1990.
Hermann Hankel. 1869.
Clebsch-Gordan coefficients for level set methods and incompressible flow.
ACM Transactions on Graphics (TOG) 40, 6 (2021), 1–18.
Hermann Hankel. 1861.
Spatially adaptive techniques for level set methods and incompressible flow.
Changyong Lui, Wei Li, and Mathieu Desbrun. 2018.
Hermann Hankel. 1869.
Hermann Hankel. 1869.
Hermann Hankel. 1869.
Hermann Hankel. 1869.
Hermann Hankel. 1869.
Hermann Hankel. 1869.
A DERIVATIONS

This appendix derives equations (14), (30) and (31).\(^7\)

The derivations involve vector-valued differential forms \(A \in \Omega^k(M; TM) \neq 
\Gamma \left( \bigwedge^k \left( \bigwedge^1 \mathbb{R} \right) \right) \neq \mathbb{T} M \). For example, the differential \(A = \nabla v \) of a vector field \(v \in \mathbb{R}^n \) is a vector-valued 1-form \(A \in \Omega^1(M; TM) \), \(dA \mid v = \nabla v \) for each vector \(v \). Whenever a vector field or vector-valued form is taken as derivative, we use the covariant derivative (Levi-Civita connection) \(\nabla : \Omega^k(M; TM) \to \Omega^{k+1}(M; TM)\) and the exterior covariant derivative \(d^\nabla : \Omega^k(M; TM) \to \Omega^{k+1}(M; TM)\). We will use the internal product \(\omega \cdot A \in \Omega^k(M; TM) \to \Omega^{k-1}(M; TM)\) as well as the identity vector-valued 1-form \(I \in \Omega^1(M; TM), i_A(I) = \nabla v \) for all vector \(v \). Note that for each \(v, w \in \mathbb{R}^n \), \(v \cdot A \in \mathbb{R} \) is the Lie bracket of vector fields \(v, w \).

\[ \text{A.1 Derivation of (14)} \]

Here we show that \(d \mathcal{L} \left( a_1 \right) = \left( \nabla a_1 \right) + \left( \nabla, a_1 \right) : \]

\[ \mathcal{L} \left( a_1 \right) \left( \Omega^2 \mathbb{R} \right) d \mathcal{L} \left( a_1 \right) \setminus 1 \neq 0 \]

\[ \left( \nabla v \right) \left( a_1 \right) + \left( \nabla, a_1 \right) \]

\[ = 0 \text{ for } (eq. (28)). \]

\[ \text{A.2 Derivation of (31)} \]

Here we show that if \(u \) satisfies (29) under a divergence-free \(v \), then \(\frac{\partial}{\partial t} w + \nabla w - \nabla w = 0 \) (eq. (31)) holds for \(w = \text{curl } u \). By \(\text{(14)}, (eq. (29)) implies that \(\eta = u^1 \) satisfies \(\frac{\partial}{\partial t} \eta + \nabla \eta = 0 \) (eq. (28)). Applying \(d \) to (28) and using (42), we obtain \(\frac{\partial}{\partial t} \omega + \nabla \omega = 0 \) for the vorticity 2-form \(\omega = d\eta \). In 3D, the relationship between the 2-form \(\omega = d\eta \) and the vector field \(w = \text{curl } u \) is given by \(\omega = i_w \mu \) where \(\mu \in \Omega^3(M) \) is the volume form. Using this relationship we obtain

\[ \frac{\partial}{\partial t} \omega + \nabla \omega = 0, \]

\[ \text{Finally, we arrive at (31) by substituting } v, w = \nabla w - \nabla w \text{ (eq. (42)).} \]

\[ \text{A.3 Derivation of (30)} \]

We show that if \(v \) is divergence-free and \(u \) satisfies \(\frac{\partial}{\partial t} \nabla w - \nabla w = 0 \), then \(\nabla \times \omega = 0 \) satisfies (30). Differing from Appendix A.2 by the term \(\nabla u \), the 1-form \(\eta = u^1 \) satisfies \(\frac{\partial}{\partial t} \eta + \nabla \eta = \left( \nabla w, u \right) \). Taking \(d \) on both sides of the equation yields

\[ \frac{\partial}{\partial t} \omega + \nabla \omega = 0, \]

\[ \frac{\partial}{\partial t} \nabla w - \nabla w \text{ (eq. (42)).} \]

\[ \frac{\partial}{\partial t} \omega + \nabla \omega = \left( \nabla w, u \right) \]