Eurographics Symposium on Geometry Processing 2014
Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

Volume 33 (2014), Number 5

Functional Fluids on Surfaces

Omri Azencot[1]* Steffen Weißmann[2]* Maks Ovsjanikov[3] Max Wardetzky[4] Mirela Ben-Chen[1]

[1]Technion – Israel Institute of Technology [2]TU Berlin [3]LIX, École Polytechnique [4]University of Göttingen

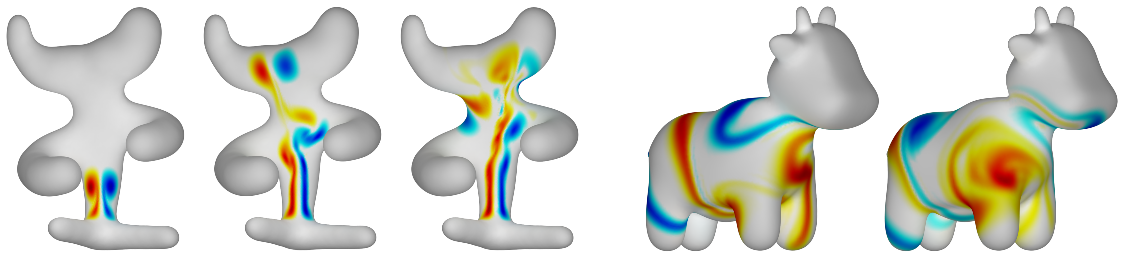

Figure 1: *Jet flow (left) and shear layer flow (right) on curved surfaces.*

Abstract

Fluid simulation plays a key role in various domains of science including computer graphics. While most existing work addresses fluids on bounded Euclidean domains, we consider the problem of simulating the behavior of an incompressible fluid on a curved surface represented as an unstructured triangle mesh. Unlike the commonly used Eulerian description of the fluid using its time-varying velocity field, we propose to model fluids using their vorticity, i.e., by a (time varying) scalar function on the surface. During each time step, we advance scalar vorticity along two consecutive, stationary velocity fields. This approach leads to a variational integrator *in the space continuous setting. In addition, using this approach, the update rule amounts to manipulating functions on the surface using* linear operators*, which can be discretized efficiently using the recently introduced functional approach to vector fields. Combining these time and space discretizations leads to a conceptually and algorithmically simple approach, which is efficient, time-reversible and conserves vorticity by construction. We further demonstrate that our method exhibits no numerical dissipation and is able to reproduce intricate phenomena such as vortex shedding from boundaries.*

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation; Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

Fluids are fascinatingly complex and challenging to simulate, with applications ranging from aerodynamics and meteorology to special effects in computer animation, to name just a few. While fluids in Euclidean domains have been extensively studied in both computational fluid dynamics and computer graphics [Bri08], fluid simulation on *curved* surfaces has mostly been limited to special cases (e.g., spheres) or particular surface representations, such as subdivision surfaces [Sta03], and practical numerical simulation methods are scarce. This is unfortunate, as simulation of fluids on surfaces has practical value in a variety of domains, including, e.g., atmospheric research [MWC92], the investigation of liquid crystal films [CM05,Bae12], and the entertainment industry.

* These authors contributed equally to this work.

The main obstacle to adopting successful numerical algorithms from Euclidean domains to surfaces stems from the fact that a fluid is most often represented by its velocity field, and the equations governing the behavior of the physical system require computing derivatives of vector fields, which is challenging on a discrete surface.

Many fluids are naturally *incompressible*, i.e., the flow preserves the volume of the fluid. In this case the flow can be represented by its *vorticity*, given by the *curl* of the velocity field. In general, vorticity is a vector field that describes the local spinning motion of the fluid. On two dimensional domains, such as surrfaces in 3D, vorticity can be represented as a (time-varying) *scalar function*. This change of perspective significantly simplifies the analysis and simulation of a fluid, since its behavior can be succinctly described using linear operators that act on real-valued functions on the surface.

Although this fact is well known [Saf92], it has, somewhat surprisingly, received little attention in the context of designing numerical methods for simulating fluids on surfaces. We make use of this formulation in order to construct a time integrator for vorticity on smooth surfaces, which is solely based on first principles of vortex dynamics. Our time symmetric advection scheme is intuitive and easy to implement; yet, it turns out to be *variational*, i.e., belonging to the class of structure preserving Lie group integrators for so-called Lie–Poisson systems [MV91, BS99, MPS99], which can be described in analogy to rigid body dynamics. Thus our method preserves momentum (i.e., vorticity) exactly, despite being of low numerical order. This in turn leads to a method that is qualitatively correct, numerically stable, and largely independent of the chosen time step.

Our resulting integration scheme is based on updating the scalar vorticity function in time. It involves the *push-forward* or *advection* of vorticity along the flow lines of a given vector field. Unlike existing methods which require the explicit computation of the flow lines of a vector field on a surface, we show that this advection can be simply computed as a product of a matrix exponential with a vector in the discrete setting, by leveraging the recently proposed functional framework for vector fields [ABCCO13] and mappings [OBCS*12]. As we show in this paper, this change of viewpoint considerably simplifies the implementation and improves the accuracy of our method.

We demonstrate that our results on Euclidean domains are comparable with existing fluid integrators, while being conceptually simple and straightforward to implement. Furthermore, we describe various experiments where our simulation reproduces the results of analytically derived configurations (for example, spherical solutions), validating its numerical fidelity. Finally, we use our method for simulating flow near the inviscid limit, including effects from vortex shedding at boundaries.

1.1. Related Work

The research dedicated to computational fluid dynamics fills numerous books, and a complete survey is beyond the scope of the paper. We thus restrict the discussion of related work to Eulerian methods on compact *two dimensional* manifolds.

Velocity-based approaches. A fluid on a Euclidean domain can be modeled by a time varying vector field by representing the components of the vector field as functions on the domain (see, e.g., [Sta99]). This approach does not immediately generalize to curved surfaces, however, where the representation of vector fields using coordinates is problematic. One option is to use global or local surface parameterizations [LWC05, HAW*09], which may introduce undesired distortion, or to restrict to special cases, such as subdivision surfaces [Sta03]. A pioneering approach for fluid simulation on general triangle meshes was suggested by Shi et al. [SY04], and later extended to deforming surfaces in [NMZ07]; however, these methods require explicit computation of flow lines, which is a challenging task. A related method [FZKH05] used an unstructured Lattice Boltzmann Model to simulate fluid behavior on triangulated surfaces by considering interactions between mesh vertices. This approach, however, also requires an explicit representation for the fluid velocity, unlike our method which relies on manipulating real-valued functions. Auer and colleagues [AMT*12] proposed a different technique based on simulating the flow on a surrounding Euclidean grid and projecting it onto the surface using the Closest Point Method. While simple and efficient, this approach requires a careful construction of the grid, whereas our method works directly on the triangle mesh itself.

Vorticity-based approaches. Discrete Exterior Calculus (DEC) approaches have been developed for simplicial manifolds. In principle, by avoiding parameterization, these approaches provide a natural framework for simulating fluid flow. One of the first methods to adopt this perspective was proposed by Elcott and colleagues [ETK*07], using the vorticity formulation of incompressible fluid flow. Their method preserves circulation, while ours preserves vorticity. We improve on their work by avoiding the computation of flow lines of the velocity vector field, which tends to be both challenging to implement and numerically unstable for triangle meshes. Additionally, our variational approach does not suffer from significant energy dissipation.

Another vorticity based method is proposed by De Witt and coleagues [DWLF12], who use the eigenfunctions of the Laplacian for accelerating the computation, and in order to avoid the somewhat costly Poisson step when computing the velocity from the vorticity. While this method could potentially be extended to curved surfaces by using the eigenfunctions of the Laplace-Beltrami operator, such an approach would require a large number of eigenvectors to correctly represent a detailed flow, which would be prohibitively costly for large models.

Several existing methods exploit the principles of DEC in combination with structure preserving variational time integrators [PMT*11, MCP*09]. We improve on these works by exploiting additional structure that is only available in two dimensions: the real-valued vorticity function. As a consequence, our approach requires only about a fifth of the number of unknowns (vorticity vs. flux and pressure). Further, our formulation avoids the computation of the Lie–Poisson bracket of vector fields, which is used to express the *time continuous* fluid motion on smooth surfaces, but is difficult to discretize. Indeed, different from [PMT*11, MCP*09], by *first* discretizing time and *then* space, we circumvent this difficulty and the attendant need of projecting back onto the subspace of divergence-free vector fields. Then, using the framework of functional vector fields on discrete surfaces [OBCS*12, ABCCO13], the *spatial discretization* is straightforward in our setup, in particular by avoiding the costly computation of the flow lines of a vector field on a surface for advecting the vorticity function.

Thus while being intrinsic, intuitive and easy to implement, our method is variational, and thus preserves many structural properties of the flow even for large time steps.

Our method is time reversible and exhibits no numerical diffusion. Additionally, it conserves vorticity by construction. As a consequence, we *automatically* obtain correct vortex shedding. This contrasts our method with (i) Lagrangian frameworks, where vortex shedding is modeled by adding fractions of the boundary layer vorticity to the flow [PK05, WP10] and (ii) Eulerian approaches where vortex shedding is hampered by numerical diffusion, requiring additional constructions, for instance using precomputed boundary layers [PTSG09] or hybrid approaches [ZLCW13].

2. Fluid Flow on Surfaces

The motion of an incompressible and inviscid fluid on a two dimensional Riemannian manifold $(\mathcal{M}, \langle .,. \rangle)$ is described by a time-dependent, divergence-free velocity vector field v_t, whose time evolution is governed by Euler's equation. Here we discus Euler's equation in its *vorticity* formulation, since this is the formulation that we work with. Before introducing our temporal and spatial discretizations below, we first present the time and space continuous setting. For further details on the vorticity formulation of Euler's equation, we refer to e.g. [Saf92, CM00].

Fluid velocity. If we restrict our attention to a two dimensional manifold $\mathcal{M}$, then the divergence free velocity vector field v_t that characterizes the fluid motion has a simple representation in terms of a time varying scalar function σ_t, known as the *stream function*. This relationship is given by

$$v_t = -\mathcal{J}\nabla\sigma_t + \eta_0 \,, \tag{1}$$

where ∇ is the usual gradient operator acting on functions, $\mathcal{J}$ denotes a (pointwise) rotation of a vector field by $\pi/2$ in each tangent plane, and η_0 is a time constant *harmonic* vector field (i.e., both divergence and curl-free). If the manifold has a boundary, then we further require that σ_t vanishes identically on it. Notice that (1) corresponds to the Hodge–Helmholtz decomposition of v_t, using zero boundary conditions for σ_t.

The curl of the velocity vector field,

$$\omega_t = \operatorname{curl} v_t \,, \tag{2}$$

is known as *vorticity*. While in 3D domains vorticity is itself a vector field, for two dimensional manifolds it can be represented by a *scalar vorticity function*. In this case, the stream function and vorticity are related through

$$\omega_t = -\Delta\sigma_t \,, \tag{3}$$

where Δ is the Laplace–Beltrami operator. Note that v_t is defined by ω_t up to the harmonic component η_0 and, for closed surfaces, σ_t is defined by ω_t up to an additive constant.

Vortex dynamics. The fluid motion is governed by Euler's equation, most compactly expressed in *vorticity form*,

$$\dot{\omega}_t = -\langle \nabla\omega_t, v_t \rangle \,, \tag{4}$$

where $\dot{\omega}_t = \frac{d}{dt}\omega_t$ is the time derivative of the vorticity function. A direct consequence of this equation is that ω_t is *frozen-in*, i.e., it is transported in the same way as fluid particles (see e.g. [Dav04] p. 49). The velocity field can be recovered from vorticity by first computing the stream function σ_t using the *linear* equation (3) and then plugging the result into (1). Hence, Eq. (4) can be viewed as an evolution equation for both ω_t alone and for the whole fluid motion.

Viscosity. So far we have assumed the fluid to be inviscid, i.e., that there is no energy loss due to viscous friction. Nonetheless, the *limit* of zero viscosity yields the dynamical and visual complexity of fluids, such as smoke. The assumption of zero viscosity differs, however, from the limit of zero viscosity: in the absence of viscosity there exists no mechanism for the creation of vorticity, while in the limit vorticity is created through vortex shedding from boundary layers.[*] While the above exposition only treats the inviscid case, viscosity is readily incorporated into the equations of motion using Arnold's observation [AK98] that viscocity can be regarded as an external force acting on the fluid,

$$\dot{\omega}_t = -\langle \nabla\omega_t, v_t \rangle - \rho\Delta\omega_t \,. \tag{5}$$

Note that this equation represents the vorticity form of the general *Navier-Stokes* equation of fluid motion (in the absence of additional external forces), where ρ is scalar representing the kinematic viscosity. We return to viscosity later

[*] This insight explained *d'Alembert's paradox*, which predicts vanishing drag on bodies moving with constant velocity [AJ05].

in our exposition and for now confine the discussion to the inviscid setting.

Flows of divergence-free vector fields. A key aspect of our method is the evolution of the vorticity function along the flow-lines of the fluid's velocity field. For this, we recall the notion of the flow of a time-varying vector field.

Given a time-dependent velocity field v_t, its *flow* $\varphi_t(p)$ describes the position after time t of a particle that starts at a point p at time 0. Formally, the flow is defined via

$$\dot{\varphi}_t(p) = v_t(\varphi_t(p)), \quad \varphi_0(p) = p, \tag{6}$$

for all $p \in \mathcal{M}$. Thus $\varphi_t(p)$ defines a curve on $\mathcal{M}$, and $\dot{\varphi}_t(p)$ is its tangent vector at the point $\varphi_t(p)$.

The flow φ_t is an invertible self-map on $\mathcal{M}$, i.e., $\varphi_t : \mathcal{M} \to \mathcal{M}$, and as such it can also be used to transport real-valued functions on $\mathcal{M}$. In particular, the flow φ_t acts linearly on smooth functions $f\colon \mathcal{M} \to \mathbb{R}$ through the pushforward:

$$\Phi_t(f) = f \circ \varphi_t^{-1} \ . \tag{7}$$

Note that Φ_t is a linear operator acting on real-valued functions defined on $\mathcal{M}$. In terms of the flow of the velocity field, vorticity satisfies

$$\omega_t = \Phi_t(\omega_0) \ , \tag{8}$$

where $\omega_0 = \operatorname{curl} v_0$ and Φ_t is the pushforward of the flow φ_t associated with v_t. Physically, this resembles the well known fact that vorticity is advected along the fluid flow.

We introduce the flow as a conceptual tool here. However, our implementation does not require an explicit calculation of the flow. Indeed, one of our key observations is that discretizing Eq. (8) directly is much simpler than computing the flow φ_t, and can be done efficiently through a simple matrix exponential, by utilizing the recently proposed functional representation for vector fields [ABCCO13], and mappings [OBCS*12]. In this framework, Φ_t is referred to as the *functional map* that corresponds to the flow φ_t.

3. Time Discretization

When discretzing time (but not yet space), we seek to determine from an initial vorticity ω_0, a sequence ω_i that *exactly* respects the defining properties of ideal fluid flow (Eqs. (1), (3), and (8)), independently of the time step. We achieve this by introducing a sequence of time-discrete flow updates, resembling the time-continuous setting. Indeed, notice that Eq. (8) in the time-continuous case implies that $\omega_{t+s} = \Phi_{t+s}(\omega_0) = \Phi_t(\omega_s)$, suggesting that the time-discrete flow update can be performed incrementally. Notice further that the identity $\Phi_{t+s}(\omega_0) = \Phi_t(\omega_s)$ implies $\Phi_{-t}(\omega_t) = \omega_0$, which is known as *conservation of vorticity*.

Accordingly, we define in the time-discrete case:

Vorticity conservation: Each ω_{i+1} is obtained by pushing forward ω_i, i.e., for two consecutive ω_i, ω_{i+1} there is a functional update map $\Phi_{i\to i+1}$ such that

$$\omega_{i+1} = \Phi_{i\to i+1}(\omega_i) \ . \tag{9}$$

It remains to specify the exact structure of the linear operators $\Phi_{i\to i+1}$, which we do as follows:

Self advection: The update $\Phi_{i\to i+1}$ is obtained by the flows of the (time-constant) divergence free velocity fields v_i and v_{i+1}, which are (linearly) coupled with vorticity via $\omega_{i+1} = \operatorname{curl} v_{i+1}$ and $\omega_i = \operatorname{curl} v_i$. Namely, we first flow on v_i for a fraction $1-s$ of the time step, and then on v_{i+1} for a fraction s of the time step. Hence, for $s \in [0,1]$ we require

$$\Phi_{i\to i+1} = \Phi_s^{i+1} \circ \Phi_{1-s}^i \ , \tag{10}$$

where Φ_s^i is the functional representation of the flow of v_i for time $t = sh$ for a given time step h (see Fig. 2).

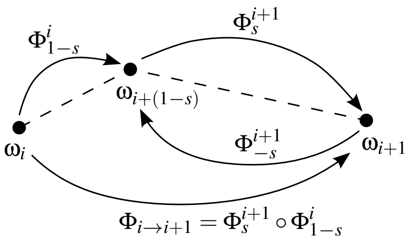

Figure 2: *The new vorticity ω_{i+1} is obtained by first advecting ω_i along v_i for a fraction $1-s$ of the time step, and then along v_{i+1} for the remainder of the time step. This is equivalent to matching the forward advected ω_i (along v_i) with the backward advected ω_{i+1} (along v_{i+1}).*

Combining Eqs. (9) and (10) we obtain a one-parameter family of time integrators,

$$\Phi_{-s}^{i+1}(\omega_{i+1}) = \Phi_{1-s}^i(\omega_i) \ . \tag{11}$$

Note that Eq. (11) is a non-linear implicit equation for ω_{i+1} since Φ^{i+1} depends non-linearly on ω_{i+1}. We describe a way to solve this equation in practice in Sec. 4, in particular using Eq. (16), which considers the explicit dependence of Φ^{i+1} on v^{i+1} (and thus ω^{i+1}) in the discrete setting.

Two particular choices of s stand out: For $s = 0$ we obtain the explicit update equation

$$\omega_{i+1} = \Phi_1^i(\omega_i) \ ,$$

which gives rise to a particularly efficient (but less accurate and stable) implementation. Instead, in order to maintain *structure preservation*, we work with $s = 1/2$ to obtain an implicit, *time-reversible* trapezoidal scheme,

$$\Phi_{-1/2}^{i+1}(\omega_{i+1}) = \Phi_{1/2}^i(\omega_i) \ . \tag{12}$$

In summary, the above update method computes ω_{i+1} from ω_i through the flows $\Phi_{-1/2}^{i+1}$ and $\Phi_{1/2}^{i+1}$ of the two stationary vector fields v_{i+1} and v_i. We stress again that in our

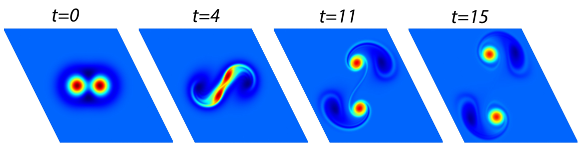

Figure 3: *Taylor vortices in the plane with periodic boundary conditions. Compare with [MCP*09, Fig. 4].*

implementation we avoid explicit computation of flows, as explained further below.

Note that the reversible nature of the time-integrator immediately implies that there is no loss of information in between time steps, i.e., our algorithm has no numerical diffusion. As we further conserve vorticity by construction, we achieve plausible and stable fluid simulations even for large time steps over long simulation periods.

Discussion. Our method is a trapezoidal rule and thus second order in time. Apart from the invariants enforced by construction, our time discretization also preserves the Hamiltonian structure of ideal fluid flow in continuous space. In fact, our method can be derived along the lines of [BS99] using a suitable discrete Lagrangian and thus belongs to the class of structure preserving Lie group integrators for Lie–Poisson systems [MV91, BS99, MPS99]. Note, however, that this depends crucially on the fact that the space of smooth functions on $\mathcal{M}$ carries a so-called Poisson structure. In the spatial discretization we are unaware of such a structure, leading to a method which is no longer Poisson but still variational, similar to existing variational fluid integrators on spatial discretizations [PMT*11].

4. Spatial Discretization

In the discrete case, we are given a triangle mesh $\mathscr{M} = (\mathscr{V}, \mathscr{E}, \mathscr{F})$, and need to represent scalar functions, vector fields, and the corresponding operators that map between them. In addition, we require a spatial discretization for the advection operators Φ^i_s.

We represent real-valued functions as scalars on the vertices of the mesh, i.e., $f: \mathscr{V} \to \mathbb{R}$, and extend them to the whole surface using the standard piecewise linear hat basis functions. We also consider vector fields as being piecewise constant on the faces of the mesh, i.e, $\mathrm{v}: \mathscr{F} \to \mathbb{R}^3$, s.t. each v is parallel to the plane spanned by face i.

Differential operators. We use two sets of functions as the representatives of our vector fields: the stream functions σ_t, and the vorticities ω_t. To mimic the continuous case, we require operators ∇, curl, div, and Δ to be such that the following relationships hold:

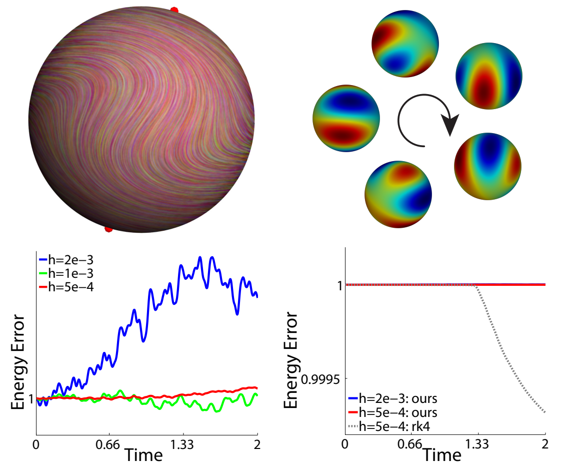

Figure 4: *When the vector field on the top left is used as initial condition, the corresponding vorticity simply rotates on the sphere (top right). The graph (bottom left) shows the relative kinetic energy of the vector field at time t compared to the initial energy, for different time steps h. The maximum change is on the order of 10^{-5}. Compared to the Runge-Kutta time integrator our method is more stable for a longer time with a larger time step (bottom right).*

$$\begin{gathered}\mathrm{v}_t = -\mathcal{J}\nabla\sigma_t + \eta_0, \quad \omega_t = \operatorname{curl}\mathrm{v}_t \\ \Downarrow \\ \operatorname{div}\mathrm{v}_t = 0, \quad \omega_t = -\Delta\sigma_t \ .\end{gathered} \tag{13}$$

Remarkably, the standard operators used in the literature fulfill these properties (as is shown in [PP03]), making spatial discretization straightforward in our setting.

Functional operators. In the time-continuous setting, vorticity is pushed forward from the initial vorticity ω_0 using the (functional representation) Φ_t of the flow of v_t. In the time-discrete setting, where the vector fields v_i are stationary in between time steps, we can in principle directly compute the flow of v_i and advect ω_i along this flow. This computation, however, is both costly, difficult to implement and is prone to instabilities. Instead, by considering the recently proposed functional representation of vector fields [ABCCO13] we show that the advection of vorticity can be done directly without computing the flow.

In particular, we follow [ABCCO13] to represent vector fields by their action on scalar functions. Namely, given a vector field v, the authors propose to consider the associated derivation operator, given by:

$$\mathcal{V}(f) = \langle \nabla f, \mathrm{v} \rangle \ . \tag{14}$$

Following [ABCCO13], we discretize Eq. 14 so that given a scalar function f on the vertices of the mesh, $g = \mathcal{V}(f)$ is

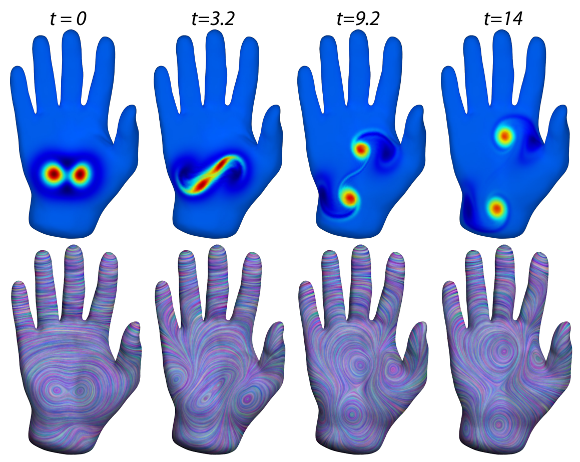

Figure 5: *Taylor vortices on curved surfaces exhibit the same qualitative behavior as in the plane.*

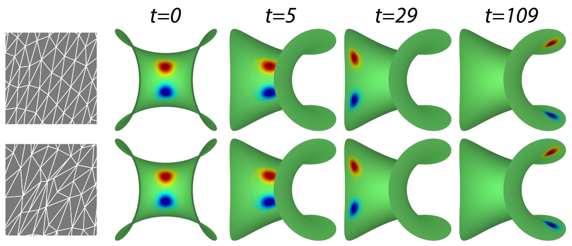

Figure 6: *A pair of vortices with equal but opposite strength on a hyperbolic surface. The vortices move to the boundary where they separate, travel independently along the boundary, and join again at the opposite side. The bottom row shows the same experiment on a poor triangulation, emphasizing the robustness of our method to the underlying mesh. This experiment is with zero viscosity. Compare with Fig. 11 for similar initial conditions, with* non zero *viscosity, which yields vortex shedding from the boundary.*

another such function, whose value at vertex i is given by:

$$g_i = \frac{1}{\sum_{j\in N(i)} A_j} \sum_{j\in N(i)} \langle \nabla f_j, \mathrm{v}_j \rangle A_j \,, \qquad (15)$$

where the sums run over all faces *adjacent* to vertex i, ∇f_j denotes the value of ∇f in face j, and A_j is the area of the j^{th} face. The resulting linear operator is given by a sparse matrix of size $|\mathscr{V}| \times |\mathscr{V}|$, which is obtained by applying $\mathcal{V}$ to the piecewise linear hat basis functions. In the following we identify $\mathcal{V}$ with this matrix.

The main advantage of representing vector fields as linear operators acting on functions in our setup is that (the functional representation of) the flow of a stationary vector field v is simply given by the matrix exponential of $\mathcal{V}$ (see Lemma 2.5 in [ABCCO13]). Thus, the functional map corresponding to the flow of v can be computed directly from $\mathcal{V}$ via

$$\Phi_s = \exp(sh\mathcal{V}) \;. \qquad (16)$$

Furthermore, advecting a function f with the flow Φ_s of v, can be done simply by the matrix vector multiplication $\exp(sh\mathcal{V})f$. Crucially, in the discrete setting, this product can be computed efficiently without evaluating the full matrix exponential, which can be both dense and difficult to approximate [AMH11]. We leverage this insight in the context of fluid simulation by using Equation (16) to derive an accurate and stable advection procedure, which circumvents the need to compute the full flow of the velocity field.

This leads to the following space-discrete version of Eq. (12):

$$\exp\left(-\frac{h}{2}\mathcal{V}_{i+1}\right)\omega_{i+1} = \exp\left(\frac{h}{2}\mathcal{V}_i\right)\omega_i \,. \qquad (17)$$

Notice that the above equation is an *exact* integration of ω advected along piecewise constant flows. This is in contrast to approaches advecting velocity [MCP*09, PMT*11], where by construction only low order approximations can be used. We practically observe that for our method a first order approximation of the exponential map is sufficient and does not decrease simulation quality. This amounts to time integration using the trapezoidal rule in the spatial discretization, preserving second order accuracy from the space continuous setting. Since, to our knowledge, a proper spatially discrete Poisson structure is missing, our time integrator may no longer be Lie–Poisson in this case. Nonetheless, it is still variational, time reversible, and vorticity conserving.

5. Implementation

Our temporal and spatial discretizations lead to a simple algorithm, which evolves the vorticity in time, by computing ω_{i+1} from a given ω_i, so that the whole fluid motion is obtained from an initial vorticity ω_0. Below we discuss implementation details required for making our method practical and efficient.

Solution of the non-linear equation. The update rule we suggest in Eq. (17) is a non-linear equation for ω_{i+1}, since $\mathcal{V}_i$ and $\mathcal{V}_{i+1}$ depend (linearly) on ω_i and ω_{i+1} through $\omega_i = \operatorname{curl} \mathrm{v}_i$ and $\omega_{i+1} = \operatorname{curl} \mathrm{v}_{i+1}$, respectively. For an efficient solve, we (i) express ω_{i+1} in terms of the stream function σ_{i+1} and (ii) use the first order approximation $\exp(\varepsilon A) \approx I + \varepsilon A$ of the matrix exponential. That is, we solve

$$-\left(I + \frac{h}{2}\mathcal{V}_i\right)\omega_i = \left(I - \frac{h}{2}\mathcal{V}_{i+1}\right)\Delta\sigma_{i+1} \,, \qquad (18)$$

which is quadratic in σ_{i+1}, and then recover ω_{i+1} and v_{i+1} using Equations (13): $\omega_{i+1} = -\Delta\sigma_{i+1}$, $\mathrm{v}_{i+1} = -\mathcal{J}\nabla\sigma_{i+1} + \eta_0$, where η_0 is computed from v_0. This formulation allows for deriving an analytic expression of the attendant Jacobian as a sparse matrix—an essential feature for efficiency. This,

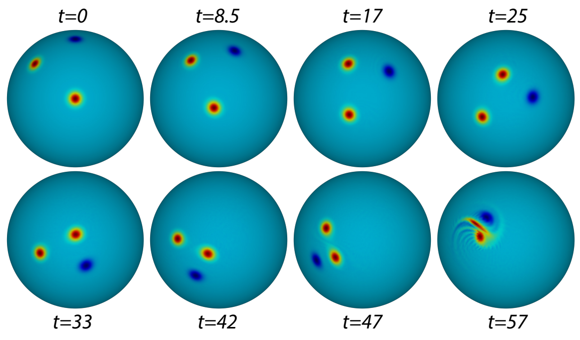

Figure 7: *The above initial configuration on a sphere is known to collapse for singular point vortices [VL13]. Our method successfully reproduces this result.*

in turn, allows for using a standard Gauss–Newton solver, which typically converges in two to five iterations. As an initial guess for the solver, we use the one-point quadrature $-\Delta\sigma_{i+1} \approx \exp(h\mathcal{V}_i)\,\omega_i$, which can be computed efficiently [AMH11].

Viscosity. So far we have only discussed inviscid fluids. However, as explained above, the treatment of viscosity can be readily integrated into our method. Adding the viscous force to both sides of the update equation (18) leads to

$$-\left(I+\rho\frac{h}{2}\Delta\right)\left(I+\frac{h}{2}\mathcal{V}_i\right)\omega_i = \left(I-\rho\frac{h}{2}\Delta\right)\left(I-\frac{h}{2}\mathcal{V}_{i+1}\right)\Delta\sigma_{i+1}. \quad (19)$$

Adding viscosity not only allows for simulating complex physical phenomena, such as *vortex shedding*, as we explain in the next section, but also has numerical advantages. Indeed, various Eulerian methods suffer inherently from numerical dissipation, to the extent that makes it necessary to re-inject lost vorticity [FSJ01]. Other Eulerian methods with no numerical dissipation (such as [MCP*09, PMT*11] and ours) require explicit addition of viscosity in order to prevent discretization artefacts. Indeed, while in the spatially continuous case energy cascades to smaller and smaller scales [Cho94], the number of available frequencies in the spatially discrete setting is limited. Without viscosity, this results in accumulation in the highest available frequencies, leading to discretization artefacts.

Boundaries. The treatment of domains with boundary is straightforward in our approach. We solve Eq. (19) under the constraint that the stream function σ is zero along every boundary component. In practice we use a sparse matrix that maps functions from all mesh vertices to interior vertices (by simply ignoring boundary vertices). This matrix is applied to Eq. (19) as well as to its Jacobian, and we solve for inner vertices only using Gauss–Newton's method.

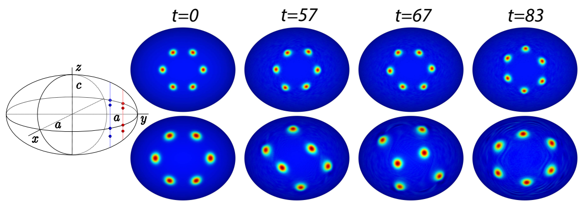

Figure 8: *A ring of 6 vortices exhibits different behavior when placed on different parts of an oblate spheroid (left). The top row shows vortices placed closer to the tip of the spheroid (red dots in the illustration), and the bottom row shows the behavior for similar vortices placed closer to the $x-z$ plane (blue dots in the illustration).*

6. Evaluation

We have extensively evaluated our algorithm, with focus on numerical stability, energy behavior, and physical correctness. We have simulated known solutions on planar domains in order to compare our results with existing methods, and used flows with known analytic solutions on curved surfaces. Further, we have simulated a number of interesting flows on curved surfaces, including effects from vortex shedding at boundaries near the inviscid limit, which are inspired by previous work. All the results are also shown in the accompanying video. All figures, unless mentioned otherwise, show vorticity of the flow, color coded on the surface. In figures 4 and 5 the flow is additionally visualized using the method of [PZ11]. Finally, we investigated how the time varying operator $\mathcal{V}_t$ can be used to uncover global behavior of the flow.

Performance. We used a non-optimized MATLAB implementation, with a standard Gauss–Newton solver for Eq. (19), using the analytic, sparse Jacobian. In all our experiments, the method was very stable and converged in 2-5 Newton iterations (using a tolerance of 10^{-7}), depending on time step size and flow complexity. The experiments were performed on an Intel i7 processor, with 16 GB RAM. In our experiments the method scaled linearly with mesh size, and a single Newton iteration typically took around 1 second per 10K vertices.

6.1. Analytic solutions

Planar Taylor vortices. We first tested our method on the well-known Taylor vortices configuration on a planar Euclidean domain with periodic boundary conditions (see e.g., [MCP*09]). In this experiment, two Taylor vortices either merge or separate depending on their initial distance. Taking this distance to be just above the critical bifurcation threshold (i.e., the vortices should separate) provides an excellent test case for fluid simulation methods (see [McK07, Eq. (1.16)] for initial conditions). Our method reproduces

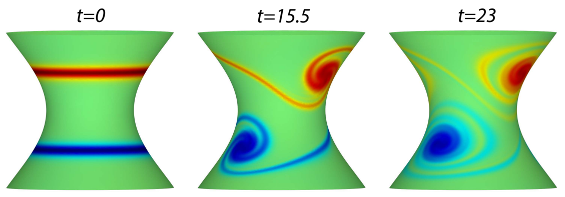

Figure 9: *A double shear flow on a hyperboloid. Note the thinning of the shear layers, and the creation of vortices. See [SS13] for the reference behavior in the plane.*

this complex dynamical behavior and produces correct results as shown in Fig. 3 (compare with [MCP*09, Fig. 4]).

Rotating sphere flow. On the sphere an analytic solution exists for fluid flow, whose initial condition consist of the combination of a Killing vector field with a rotated gradient of an eigenfunction of the Laplace-Beltrami operator. In particular, it can be shown that the energy of *inviscid* flow with these initial conditions remains in the low frequencies, giving a periodic motion. This makes the configuration a good test case for energy conservation, and for validating qualitative behavior of our solver. Energy plots for different time steps are shown in Fig. 4. The relative change in energy is on the order of 10^{-5}. Note that while our method remains stable for a time step of $h = 2 \cdot 10^{-3}$, replacing our time integrator with a Runge-Kutta time integrator leads to significant energy loss for a much smaller time step.

6.2. Vortex configuration on surfaces

Taylor vortices on a curved surface. We generated initial conditions similar to the Taylor vortices in the plane on a curved surface representing a hand. We used the same parameters as in [McK07, Eq. (1.16)], measured as geodesic distances (using [CWW13]) on the mesh. Fig. 5 shows frames from the animation, yielding the same qualitative behavior as in the plane.

Vortex pair. In the plane, two point vortices of equal and opposite strength translate along the orthogonal bisector of their connecting line. This can be viewed as a zero dimensional vortex ring, i.e., the 2D equivalent to a circular vortex filament in 3D. In Fig. 6 we demonstrate the same qualitative behavior on a hyperbolic surface. In the absence of viscosity the vortices travel towards the boundary, where they separate and move independently along the boundary, until they meet again at the opposite side. This configuration is extremely stable over very long simulation times (performing the periodic motion several times), demonstrating the absence of numerical diffusion, vorticity conservation, and excellent energy behavior of our method.

The qualitative behavior of other point vortex configurations is also reliably reproduced by our method. For instance,

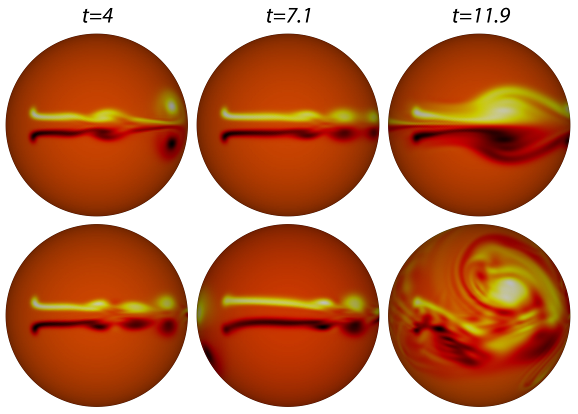

Figure 10: *Two jet simulations on a sphere with a symmetric triangulation. On the top row the initial vorticity function shares the symmetry of the mesh. Our method preserves this symmetry, yielding a symmetric flow. On the bottom row the initial vorticity is no longer symmetric, yielding a more realistic turbulent flow. For such unstable flows the simulation is sensitive to the discretization of the initial vorticity.*

Fig. 7 shows a configuration on the sphere which is known to collapse [VL13].

Oblate sphere. It is known that N point vortices (with $N < 7$) on a round sphere stay in a stable relative equilibrium, when equally spaced along a latitude circle below a critical colatitude [VL13]. We demonstrate a similar configuration on a stretched (along one axis) sphere, where point vortices are replaced by Gaussian vortices. As shown in Fig. 8 in the top row, when placing the vortices around a "flat" pole below the critical angle (for point vortices on a round sphere), the flow preserves the six fold symmetry, as in on the round sphere. In the bottom row, the vortices are positioned *above* the critical angle (i.e. further from the tip), and symmetry breaks. Still, the vortices show periodic behavior, indicating that the corresponding point vortex configuration might be integrable. To our knowledge such configurations have not been studied analytically.

Double shear flow. Two vortex layers of equal but opposite strength with small perturbations generate a vortical flow with a symmetric structure [SS13, Section 4.2]. We use similar initial conditions on the hyperboloid. The resulting flow exhibits the same qualitative behavior, including the intricate symmetries. Fig. 9 shows frames from the resulting simulation. Note how vortices curl up, creating thinner and thinner vortex lines, similarly to the reference behavior in the plane.

6.3. Turbulent flow and vortex shedding

Jet on a sphere. A jet is modeled by steadily injecting vorticity of equal but opposite strength at both sides of the jet's

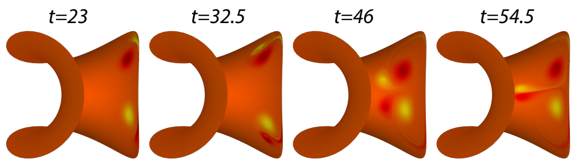

Figure 11: *Two vortices collide with the boundary on the Enneper's surface. Note the details in shed vorticity generated by the boundary due to viscosity. Compare with Fig. 6 for similar initial conditions but zero viscosity.*

orifice. Fig. 10 shows two jet simulations on a symmetric triangulation of the sphere. In the top row, the initial vorticity function shares the symmetry of the mesh, and due to the stability of our method, this symmetry is exactly preserved by the flow. In the bottom row, the initial vorticity is not exactly symmetric with respect to the triangulation, which introduces instabilities in the flow, leading to a more realistic simulation. In general, unstable flows such as this are sensitive to the discretization of the initial conditions.

Vortex shedding. As explained before, our method naturally handles vortex shedding from boundaries. We used a pair of point vortices on the Enneper's surface, as in Fig. 6, but with non-zero viscosity. In contrast to the inviscid case, where the two vortices separate and travel along the whole boundary, with viscosity vortex shedding from the boundary generates two additional small vortices which "trap" the original vortices and prevent them from circulating, see Fig. 11.

In the accompanying video we also show the wake behind a two dimensional cylinder on a round sphere, generated through vortex shedding.

6.4. Flow properties

Globally invariant functions. In addition to being instrumental in the computation of vorticity advection, the functional representation of vector fields of Azencot et al. [ABCCO13] also allows us to gain information about different properties of the flow, that would be difficult to obtain otherwise. Here, we briefly outline one such application and leave further exploration as future work.

Given the solution to a flow, we may be interested in finding regions of the surface that are invariant under the flow, i.e., regions from which the fluid does not leave or regions into which the fluid does not enter during the simulation. We relax this problem to consider all functions f such that $\Phi_t(f) = f$ for all t. In order to find such a function, note that if f is mapped to itself under the flow of a constant vector field v, then $\exp(t\mathcal{V})f = f$ for all t, which means that $\mathcal{V}f = 0$, or equivalently that f is in the kernel of $\mathcal{V}$. Therefore, we are looking for a function f that is *simultaneously* in the kernels of all $\mathcal{V}_t$. Notice that this is the case if and only if f is in the kernel of $\sum_t \mathcal{V}_t^T \mathcal{V}_t$, where we are using that in

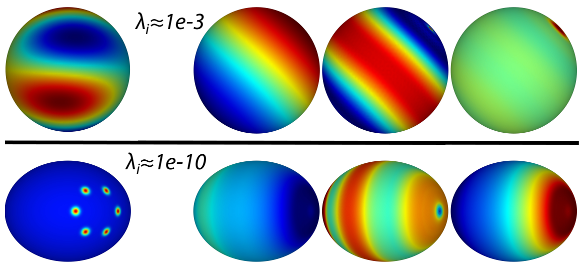

Figure 12: *A function which is invariant to the whole flow, computed from the flow's kernel. Top row, for the flow from Fig. 4, and bottom row for the flow from Fig. 8.*

the time-discrete case there exist only finitely many temporal sample points. Using this observation, we compute the kernel of $\sum_t \mathcal{V}_t^T \mathcal{V}_t$ for the rotating flow from Fig. 4 and the stable vortex ring from Fig. 8. The resulting functions in the respective kernels are shown in Fig. 12.

7. Conclusion

Building on the vorticity formulation of fluids, we presented a method for temporally and spatially discretizing the equations of fluid flow. The attendant time integrator for the inviscid case is variational, preserves vorticity exactly, is time reversible, and does not exhibit numerical diffusion. Additionally, our approach allows for adding a principled treatment of viscosity, enabling the simulation of complex phenomena, such as vortex shedding, without any special or unphysical treatment of boundary layers. The resulting algorithm is efficient, simple to implement, and leads to unprecedented simulation results of fluid flow on curved surfaces, which were previously only possible for flat Euclidean domains.

In our derivation, we have *first* discretized time and *then* space, thereby suggesting a variational integrator in the spatially continuous setting. At the core of our integration lies the observation that each time step can be performed along two consecutive stationary velocity segments. The flow corresponding to these stationary segments can efficiently be computed using the functional point of view in the spatially discrete case, where it simply corresponds to a matrix exponential. We have demonstrated how to achieve efficiency by approximating this exponential up to first order terms, *without* sacrificing stability.

Acknowledgements We thank Keenan Crane for the Spot model, the AIM@SHAPE repository for the hand model, and Alexandre Djerbetian for his geodesics computation code. The authors acknowledge the Marie Curie Career Integration Grant 334283-HRGP, the CNRS chaire d'excellence, a Google Faculty Research Award, the DFG Research Center Matheon, the SFB/Transregio 109 "Discretization in Geometry and Dynamics", ISF grant 699/12, Marie Curie CIG 303511 and the state of Lower-Saxony and the Volkswagen Foundation, Hannover, Germany.

References

[ABCCO13] Azencot O., Ben-Chen M., Chazal F., Ovsjanikov M.: An operator approach to tangent vector field processing. In *Comp. Graph. Forum (Proc. SGP)* (2013), vol. 32, pp. 73–82. 2, 3, 4, 5, 6, 9

[AJ05] Anderson Jr J. D.: Ludwig Prandtl's boundary layer. *Physics Today 58*, 12 (December 2005). 3

[AK98] Arnold V. I., Khesin B. A.: *Topological Methods in Hydrodynamics*. Springer, 1998. 3

[AMH11] Al-Mohy A. H., Higham N. J.: Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM journal on scientific computing 33*, 2 (2011), 488–511. 6, 7

[AMT*12] Auer S., Macdonald C. B., Treib M., Schneider J., Westermann R.: Real-time fluid effects on surfaces using the Closest Point Method. *Computer Graphics Forum 31*, 6 (2012), 1909–1923. 2

[Bae12] Baek S. K.: Vortex interaction on curved surfaces. *Phys. Rev. E 86* (Nov 2012), 056603. 1

[Bri08] Bridson R.: *Fluid Simulation for Computer Graphics*. A K Peters, 2008. 1

[BS99] Bobenko A. I., Suris Y. B.: Discrete time Lagrangian mechanics on Lie groups, with an application to the Lagrange top. *Comm. Math. Phys. 204*, 1 (1999), 147–188. 2, 5

[Cho94] Chorin A. J.: *Vorticity and Turbulence*. Springer, 1994. 7

[CM00] Chorin A. J., Marsden J. E.: *A Mathematical Introduction to Fluid Mechanics*. Springer, 2000. 3

[CM05] Crowdy D., Marshall J.: Analytical solutions for rotating vortex arrays involving multiple vortex patches. *Journal of Fluid Mechanics 523* (2005), 307–337. 1

[CWW13] Crane K., Weischedel C., Wardetzky M.: Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph. 32* (2013). 8

[Dav04] Davidson P. A.: *Turbulence : an introduction for scientists and engineers*. Oxford University Press, Oxford, UK New York, 2004. 3

[DWLF12] De Witt T., Lessig C., Fiume E.: Fluid simulation using Laplacian eigenfunctions. *ACM Transactions on Graphics (TOG) 31*, 1 (2012), 10. 2

[ETK*07] Elcott S., Tong Y., Kanso E., Schröder P., Desbrun M.: Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG) 26*, 1 (2007), 4. 2

[FSJ01] Fedkiw R., Stam J., Jensen H. W.: Visual simulation of smoke. In *Proc. ACM/SIGGRAPH Conf.* (2001), pp. 15–22. 7

[FZKH05] Fan Z., Zhao Y., Kaufman A., He Y.: Adapted unstructured LBM for flow simulation on curved surfaces. In *Proc. SCA* (2005), pp. 245–254. 2

[HAW*09] Hegeman K., Ashikhmin M., Wang H., Qin H., Gu X.: GPU-based conformal flow on surfaces. *Commun. Inf. Syst. 9*, 2 (2009), 197–212. 2

[LWC05] Lui L. M., Wang Y., Chan T. F.: Solving PDEs on manifolds with global conformal parametriazation. In *Variational, Geometric, and Level Set Methods in Computer Vision*. Springer, 2005, pp. 307–319. 2

[McK07] McKenzie A.: *HOLA: a High-Order Lie Advection of discrete differential forms, with applications in fluid dynamics*. PhD thesis, California Institute of Technology, 2007. 7, 8

[MCP*09] Mullen P., Crane K., Pavlov D., Tong Y., Desbrun M.: Energy-preserving integrators for fluid animation. *ACM Trans. Graph. 28*, 3 (2009), 38:1–38:8. 3, 5, 6, 7, 8

[MPS99] Marsden J. E., Pekarsky S., Shkoller S.: Discrete Euler-Poincaré and Lie-Poisson equations. *Nonlinearity 12*, 6 (1999), 1647–1662. 2, 5

[MV91] Moser J., Veselov A. P.: Discrete versions of some classical integrable systems and factorization of matrix polynomials. *Communications in Mathematical Physics 139*, 2 (1991), 217–243. 2, 5

[MWC92] Miller J., Weichman P. B., Cross M. C.: Statistical mechanics, Euler's equation, and Jupiter's Red Spot. *Phys. Rev. A 45* (Feb 1992), 2328–2359. 1

[NMZ07] Neill P., Metoyer R., Zhang E.: Fluid flow on interacting deformable surfaces. In *ACM SIGGRAPH 2007 Posters* (2007), SIGGRAPH '07, ACM. 2

[OBCS*12] Ovsjanikov M., Ben-Chen M., Solomon J., Butscher A., Guibas L.: Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph. 31*, 4 (July 2012), 30:1–30:11. 2, 3, 4

[PK05] Park S. I., Kim M. J.: Vortex fluid for gaseous phenomena. In *Proc. SCA* (2005), pp. 261–270. 3

[PMT*11] Pavlov D., Mullen P., Tong Y., Kanso E., Marsden J. E., Desbrun M.: Structure-preserving discretization of incompressible fluids. *Physica D: Nonlinear Phenomena 240*, 6 (2011), 443–458. 3, 5, 6, 7

[PP03] Polthier K., Preuss E.: Identifying vector field singularities using a discrete hodge decomposition. *Visualization and Mathematics 3* (2003), 113–134. 5

[PTSG09] Pfaff T., Thuerey N., Selle A., Gross M.: Synthetic turbulence using artificial boundary layers. *ACM Trans. Graph. 28*, 5 (2009), 121:1–121:10. 3

[PZ11] Palacios J., Zhang E.: Interactive visualization of rotational symmetry fields on surfaces. *Visualization and Computer Graphics, IEEE Transactions on 17*, 7 (2011), 947–955. 7

[Saf92] Saffman P. G.: *Vortex Dynamics*. Cambridge Univ Pr, 1992. 2, 3

[SS13] San O., Staples A. E.: A coarse-grid projection method for accelerating incompressible flow computations. *Journal of Computational Physics 233* (2013), 480–508. 8

[Sta99] Stam J.: Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128. 2

[Sta03] Stam J.: Flows on surfaces of arbitrary topology. In *ACM Transactions On Graphics (TOG)* (2003), vol. 22, ACM, pp. 724–731. 1, 2

[SY04] Shi L., Yu Y.: Inviscid and incompressible fluid simulation on triangle meshes. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 173–181. 2

[VL13] Vankerschaver J., Leok M.: A novel formulation of point vortex dynamics on the sphere: geometrical and numerical aspects. *Journal of Nonlinear Science* (2013), 1–37. 7, 8

[WP10] Weissmann S., Pinkall U.: Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph. 29*, 4 (2010), 115:1–115:12. 3

[ZLCW13] Zhu J., Liu Y., Chang Y., Wu E.: Animating turbulent water by vortex shedding in pic/flip. *Science China Information Sciences 56*, 3 (2013), 1–11. 3