SMAI-JCM
SMAI Journal of Computational Mathematics

Fast and robust computation of coherent Lagrangian vortices on very large two-dimensional domains

Daniel Karrasch & Nathanael Schilling

Volume 6 (2020), p. 101-124.

<http://smai-jcm.centre-mersenne.org/item?id=SMAI-JCM_2020__6__101_0>

Publication membre du
Centre Mersenne pour l'édition scientifique ouverte
http://www.centre-mersenne.org/

Sousmission sur https://smai-jcm.math.cnrs.fr/index.php/SMAI-JCM

SMAI Journal of Computational Mathematics
Vol. 6, 101-124 (2020)

Fast and robust computation of coherent Lagrangian vortices on very large two-dimensional domains

Daniel Karrasch [1]
Nathanael Schilling [2]

[1] Department of Mathematics, Technische Universität München, Garching bei München, Germany
E-mail address: karrasch@ma.tum.de
[2] Department of Mathematics, Technische Universität München, Garching bei München, Germany.

Abstract. We describe a new method for computing coherent Lagrangian vortices in two-dimensional flows according to any of the following approaches: *black-hole vortices* [24], *objective Eulerian Coherent Structures (OECSs)* [39], *material barriers to diffusive transport* [25, 26], and *constrained diffusion barriers* [26]. The method builds on ideas developed previously in [30], but our implementation alleviates a number of shortcomings and allows for the fully automated detection of such vortices on unprecedentedly challenging real-world flow problems, for which specific human interference is absolutely infeasible. Challenges include very large domains and/or parameter spaces. We demonstrate the efficacy of our method in dealing with such challenges on two test cases: first, a parameter study of a turbulent flow, and second, computing material barriers to diffusive transport in the global ocean.

2010 Mathematics Subject Classification. 65P99; 86-08.

Keywords. Lagrangian coherent structures, coherent vortices, turbulent flows.

1. Introduction

Coherent Lagrangian vortices (CLVs) play an important role in the transport and mixing of passive, possibly diffusive, scalar quantities in fluid flows. Such structures can be found in flows living on a wide variety of scales [28, 20, 1].

Over the past decade, a variety of modeling approaches have been developed to characterize CLVs. Intuitively, CLVs are viewed as material structures that sustain a non-filamenting boundary under advection by the flow [23, 24, 14]; or material structures that resist leakage of a diffusive passive scalar in an advection–diffusion process [31, 25, 26]. Data-driven approaches view CLVs as collections of trajectories that stay together under the motion of the flow [2, 17, 22, 3, 34]. Related Eulerian approaches view coherent sets as space-time structures that do not mix much with their spatial neighborhood [16, 13]. While these approaches intuitively target the same observed phenomenon, they often yield different structures when applied to the same flow [20].

To date, only few methods have been successfully applied to realistic flow problems. See, for instance, [19, 24, 30, 21, 20, 41, 18] for studies of "medium" complexity (in that either the domain is not too large or the number of known/expected structures is low), and [1] for the only—to the best of our knowledge—large-scale study. The reasons for this lack of realistic applications are manifold: (i) most methods are not fully automated or even automizable, cf. [20], (ii) some methods intrinsically do not scale well with the size of the domain, the number of expected coherent structures, or the number of tracked trajectories; and/or (iii) there are no performant and robust implementations available.

Our aim in this paper is to report on our progress towards bridging the gap between large-scale applications and those methods that subsume the "geodesic vortices" class: *black-hole vortices* [24],

This work is supported by the Priority Programme SPP 1881 Turbulent Superstructures of the Deutsche Forschungsgemeinschaft.

objective Eulerian Coherent Structures (OECSs) [39], and *material barriers to diffusive transport* [25, 26]. The algorithms are developed as part of the open-source `CoherentStructures.jl` project. These methods in principle scale well with the size of the domain, but existing implementations related to the publications [33, 30, 21, 40] failed to fully leverage this; we therefore had to make significant conceptual and implementation modifications.

Conceptually, our work is based on the index-theory-based methodology developed in [30], whose implementation was a mixture of methods implemented in [12, 33, 21]. In [40], Serra & Haller identified (i) the detection of tensor field singularities (points of repeated tensor eigenvalues) of tensor fields and (ii) the identification of their topological type as major computational bottlenecks in the implementation of [30]. Moreover, these steps required a number of parameters whose choice had—at times—unpredictable impact on the computational outcome. As an alternative, they derived an automated method for computing geodesic vortices based on the geometry of the underlying geodesic flow. On the upside, their approach (i) does not require singularity detection and type identification at all, and (ii) is designed to, in principle, not miss any coherent vortices at a given computational accuracy and spatial resolution. On the downside, however, (i) the gradient of the underlying computed tensor field is required, (ii) the computation is performed on the whole domain at once without a localization option; and (iii) the currently available implementation is not performant.

In our implementation, we have carefully addressed the issues raised by [40] regarding the implementation of [30]. Specifically, we have improved a number of aspects related to the (inherently robust) topological index-theory-based methods. In the spirit of discrete differential geometry, we now discretize the tensor index computation in a manner such that important properties are preserved, and these properties are exploited efficiently. The main reason why we argue it is worth improving on the index-based approach is that it allows for the identification of a comparatively small number of candidate regions that each potentially contain a CLV, around which one may then restrict subsequent computations. This in particular allows for straightforward parallelization. Ultimately, flow problems of high complexity become manageable.

Implemented in the modern and performant programming language `Julia` [4], our package is able to find geodesic vortices on domains of unprecedented size and orders of magnitude faster than what is the current state of the art. Specifically, we demonstrate our code (i) on a parameter-dependent turbulent flow and (ii) in a global ocean surface simulation using a computational grid of tens of millions of points. The required computational power does not exceed what is available on an ordinary work station or a modern desktop machine. While there remains room for further improvement, this shows that it is possible to effectively compute CLVs in very large-scale 2D flows and/or to perform extensive parameter studies on medium-sized domains.

This paper is organized as follows. In Section 2.1, we recall the mathematical framework of [25] that introduced the concept of "material barriers (to diffusive transport)" as an instance of the methods falling into the category of geodesic vortices. These are then summarized together with a generic computational approach in Section 2.2. Section 3 is devoted to the description of our computational approach based on index theory for planar line fields. For convenience, we have collected related facts in Appendix A. Finally, we demonstrate the outstanding capabilities of our implementation on two non-trivial applications in Section 4: a parameter study based on a minimal two-dimensional turbulence simulation on the torus, and a global ocean surface velocity simulation.

2. Background

2.1. Mathematical setting

We now recall the theory related to material barriers to diffusive transport[25], as the most recent instance of a method that fits into the geodesic vortex framework. This is also the method we use

in the examples in Section 4. Here the setting is a time-dependent incompressible fluid velocity field $\mathbf{v}\colon U \times \mathcal{T} \to \mathbb{R}^2$, where U is an open, simply connected subset of $\mathbb{R}^2$ and $\mathcal{T}$ is a finite time interval. A passive scalar u, i.e., a scalar quantity that does not affect the velocity field, undergoes *advection-diffusion* if it satisfies the partial differential equation (PDE)

$$\partial_t c + \operatorname{div}(c \cdot \mathbf{v}) = \nu \operatorname{div} \nabla c = \nu \Delta c\,. \tag{2.1}$$

In words, the density c is carried by the fluid and diffuses isotropically. The inclusion of anisotropic and/or spatially inhomogeneous and time-dependent diffusion is straightforward, but omitted here for ease of presentation. Equation (2.1) models the evolution of a range of physically relevant quantities, including concentrations of dissolved substances (like salinity and temperature) and vorticity in the 2D Navier–Stokes equations. Strictly speaking, none of these three examples is passive, but certainly temperature and salinity can be regarded as such over time scales of a few weeks or even months. Furthermore, Equation (2.1) on $\mathbb{R}^2$ can be interpreted as the *Fokker–Planck/Kolmogorov forward equation* of the stochastic differential equation

$$dX_t = \mathbf{v}(X_t, t)dt + \sqrt{2\nu}dW_t\,,$$

provided that $\mathbf{v}$ satisfies certain regularity assumptions; cf. [29].

The initial value $c(\cdot, 0) = c_0$ uniquely defines the solution of Equation (2.1) given appropriate boundary conditions on $\partial U \times \mathcal{T}$. The value $\nu > 0$ is the *diffusivity* (or the inverse Péclet number in the non-dimensionalized form) and is very small in many applications. In the absence of diffusion ($\epsilon = 0$), c is conserved and transported along the characteristics of the velocity field $\mathbf{v}$. Characteristics x_t satisfy the ordinary differential equation (ODE) $\frac{\mathrm{d}}{\mathrm{d}t}x_t = \mathbf{v}(x_t, t)$. We denote by $\mathbf{F}_{t_0}^t$ the flow-map for this ordinary differential equation, i.e., $\mathbf{F}_{t_0}^t(p)$ corresponds to the time-t solution of the ODE with initial value $x_{t_0} = p$.

By definition, *Lagrangian* (or *material* structures) are *invariant* under the flow $t \mapsto \mathbf{F}_{t_0}^t$. Hence, the flow map can be used to *define* Lagrangian coordinates in space by labelling a spatialtemporal point (x, t) using the fluid "particle" p that occupies x at time t. Clearly, in Lagrangian coordinates there is no advective transport, moreover with this change of coordinates the advection–diffusion equation (2.1) takes the form of a pure diffusion equation [36, 43, 31, 25]

$$\partial_t \tilde{c} = \nu \operatorname{div}\left(\mathbf{D}_{t_0}^t \nabla \tilde{c}\right), \tag{2.2}$$

where $\mathbf{D}_{t_0}^t(p) = (D\mathbf{F}_{t_0}^t(p,t))^{-1}(D\mathbf{F}_{t_0}^t(p,t))^{-\top}$ and $\tilde{c}(p,t) = c(\mathbf{F}_{t_0}^t(p), t)$ are, respectively, the diffusion tensor and the scalar density in Lagrangian coordinates. For notational simplicity, we omit the tilde in the notation of the scalar density function in Lagrangian coordinates. This coordinate change allows for the separation of the reversible effects of advection from the irreversible effects of the combined advection and diffusion. Note that the Lagrangian diffusion tensor field $\mathbf{D}$ is both t- and p-dependent.

In this framework, material barriers to diffusive and stochastic transport have been defined in [25] as material surfaces which extremize diffusive transport over the finite observation time interval. There, it is shown that in two-dimensional flows, the diffusive transport through a one-dimensional material manifold Γ is given in leading order (with respect to diffusivity ν) by

$$\int_\Gamma \left\langle \nabla c_0, \mathbf{T}_{t_0}^t \mathbf{n} \right\rangle \mathrm{d}A\,, \tag{2.3}$$

where $\mathbf{T}_{t_0}^t$ is the *transport tensor field*, defined as the time-average of the Lagrangian diffusion tensor fields, i.e. $\mathbf{T}_{t_0}^t = \frac{1}{t-t_0}\int_{t_0}^t \mathbf{D}_{t_0}^\tau \,\mathrm{d}\tau$, $\mathbf{n}$ is the outward-pointing normal, and $\mathrm{d}A$ is the canonical (Euclidean) surface measure. After normalizing by the length [1] of Γ and choosing a most "diffusion-prone" distribution of c_0, a functional on closed curves is found whose stationary points are null-geodesics of an

[1] The material barrier theory applies to higher dimensions, but the implementation for 2 dimensions does not generalize easily to 3 or more spatial dimensions.

indefinite metric tensor field. This fits nicely into the "geodesic vortex" framework described in the next section. As a by-product, the trace of the transport tensor, trace($\mathbf{T}$), coined *diffusion barrier strength (DBS)*, is a diagnostic field whose logarithm we will use for visualization purposes as a scalar background field in this work.

2.2. **Coherent Lagrangian vortices as null-geodesics**

Mathematically speaking, the "geodesic vortex" approach consists of the computation of closed null-geodesics of a (possibly indefinite) metric tensor field that has undergone a parameter-dependent shift. Three different vortex approaches can be formulated in this setting:

(1) the "black hole vortex" approach [24], which seeks stationary curves of a functional related to stretching;

(2) the "objective Eulerian coherent structures" (OECS) approach [39], which seeks stationary curves of a functional related to instantaneous stretching (i.e., strain);

(3) the "material barriers to diffusive transport" [25, 26] approach described in the previous section.

In these three cases, the tensor field comes from, respectively, (a) the Cauchy–Green strain tensor $\mathbf{C}^t_{t_0} = (D\mathbf{F}^t_{t_0})^\top D\mathbf{F}^t_{t_0}$, (b) the rate of strain tensor $\mathbf{S}_{t_0}$ (i.e., the symmetric part of the velocity gradient $D\mathbf{v}(\cdot, t_0)$), and (c) the transport tensor $\mathbf{T}^t_{t_0}$ defined in the previous section. In the following, we will use the generic $\mathbf{T}$ to denote any of these tensor fields.

The geodesic vortex approach seeks to find closed null-geodesic curves of $\mathbf{T} - \lambda\mathbf{I}$; here the (real) parameter λ is taken from a physically motivated range. Recall that null-geodesics are smooth curves γ that have "zero length" when measured in the indefinite metric $\mathbf{T} - \lambda\mathbf{I}$, i.e. $\gamma' \cdot (\mathbf{T} - \lambda\mathbf{I})\gamma' = 0$. It is readily verified that they can be computed as integral curves of

$$\eta^\pm_\lambda = \sqrt{\frac{\lambda_2 - \lambda}{\lambda_2 - \lambda_1}}\xi_1 \pm \sqrt{\frac{\lambda - \lambda_1}{\lambda_2 - \lambda_1}}\xi_2\,, \tag{2.4}$$

where $\lambda_1 \le \lambda_2$ are eigenvalues of $\mathbf{T}$ and ξ_1, ξ_2 are corresponding normalized eigenvectors. This is derived from the fact [24] that null-geodesics γ have uniform $\mathbf{T}$-strain along themselves, i.e., along γ one has

$$\sqrt{\frac{\gamma' \cdot \mathbf{T}\gamma'}{\gamma' \cdot \gamma'}} = \lambda\,. \tag{2.5}$$

As such, null-geodesics are closed integral curves of a planar line field, to which a corresponding index theory applies; cf. Appendices A and B. Analogously to index theory for planar vector fields, closed null-geodesics have index 1 relative to their inducing line field. As defined formally in Appendix A, we will (i) assign an index (relative to the line-field) to regions that are sets whose boundary forms a closed Jordan curve, or that are finite disjoint unions of such sets and (ii) refer to all such regions with index 1 as *elliptic regions*.

While the elliptic regions we initially identify will not have geodesic vortices as their boundaries per se, we aim to find geodesic vortices that are homotopic to the elliptic regions found. The identification of elliptic regions is the first step we use for the computation of geodesic vortices (this builds on [24, 30]).

3. **The computational approach**

In this section, we give details of our implementation approach. The structure of this section closely resembles the high-level structure of our implementation provided in `CoherentStructures.jl`. Our computational approach, at the highest level, consists of three steps:

(1) Identify certain elliptic regions as candidate regions near geodesic vortices, based on index theoretical methods applied to the first eigenvector of T; cf. Appendices A and B. for the foundational theoretical aspects.

(2) For each identified elliptic region, localize tensor field data to a neighborhood of the region.

(3) Compute closed orbits (i.e., geodesic vortices) by a shooting method in this neighborhood.

We describe each step in more detail in the following sections.

3.1. **Identification of elliptic regions**

In a computational setting, we know the values of the tensor field $\mathbf{T}$ and, hence, its subdominant eigenvector field $\ell = \xi_1$, only at a finite number of points, say, the nodes of a polygonal triangulation/mesh $\mathcal{P}$. A candidate elliptic region R will be the union of a finite set of polygonal faces $P_1, \dots, P_k$ from $\mathcal{P}$. Such regions are identified in three steps:

(1) Compute indices of every mesh face in a fast and robust manner solely from the given tensor data, without interpolation; cf. Section 3.1.1.

(2) Suitably merge mesh faces into regions with stable index, and extract those that are elliptic; cf. Section 3.1.2.

(3) Optionally, do further merging to obtain larger elliptic regions.

The second and third steps are necessary because generically, the only structurally stable singularities occurring in regular tensor fields have index $\pm\frac{1}{2}$ [10]. Therefore, unless treating a degenerate or artificial tensor field, sufficiently small single polygonal regions are not elliptic. To identify elliptic regions, we merge nearby polygonal faces with non-vanishing indices in step (ii). Consistently with the additivity of the index under curve composition/region merging (see Appendix A), we add indices of polygonal cells when they are merged.

So far we have not yet specified criteria for deciding which polygonal regions should be merged. We argue that a reasonable, robust criterion is to require that a candidate region's index shall not change when (further) enlarged by a specified radius $r > 0$. This is captured by the following definition.

Definition 3.1. We say that a region R is *r-stable* (relative to ℓ), if the set $R_s \coloneqq \{x \in \Omega;\ d(x, R) \leq s\}$ has the same index (relative to ℓ) as R for any $0 \leq s \leq r$.

Clearly, if a region R is r-stable, then within the r-vicinity of its boundary all polygonal faces have index 0. Candidate regions for being homotopic to geodesic vortices are taken to be those that are minimal (with respect to inclusion) unions of polygons that are r-stable and elliptic. We note that in some cases the assumption of minimality is too strong; as observed in [30], it is common for large geodesic vortices to bound exactly two r-stable regions, each of wedge-type (i.e., index $\frac{1}{2}$). In order to include also these elliptic regions, we additionally introduce a number of ways to merge multiple r-stable regions that have indices summing to 1.

Before giving a description of the details of steps (i)–(iii), we summarize that our identification method is fast, works directly on the line field data at mesh points, and can be used on unstructured, irregular meshes/grids. Moreover, it is unnecessary to choose pointwise orientations for the line field, or to use ad-hoc heuristics for singularity type classification. Robustness against local computational errors is achieved by automatically choosing contours large enough so that any enlargement of the contour (up to a specified size given by r) yields the same result. This is the only parameter required by the (indirect) singularity detection method.

3.1.1. *Step (i): Calculating indices*

Assume we have a polygonal mesh $\mathcal{P}$ on Ω consisting of vertices $\mathcal{V}$, edges $\mathcal{E}$, and polygonal faces $\mathcal{F}$. The vertices are points at which the value of the line field ℓ is known. Since we are working in a discrete setting, the natural curves to consider for the computation of indices are concatenations of edges in E. To this end, let γ be a simple closed Jordan curve along n edges of the mesh, i.e., passing through the vertices $v_1, \dots, v_n, v_{n+1} = v_1$ along the edges $e_i = (v_i, v_{i+1})$ and enclosing a union of polygons; cf. Figure 3.2.

Since we know the value of the line field ℓ only at the vertices, we need to approximate the curve $\ell \circ \gamma \colon [1, n+1] \to \mathbb{P}^1$ based on those values in order to approximate θ used in Definition A.1; Appendix A. We cannot apply Definition A.1 to the discrete case directly as the angles θ_i between $\ell(v_i)$ and the x-axis are determined only up to a multiple of π. We follow [44] and choose θ_i such that the angle difference $\Delta_i \coloneqq \theta_{i+1} - \theta_i$ between subsequent angle representations is minimal modulo π for $i = 1, \dots, n$. This is achieved by setting

$$\Delta_i \coloneqq \operatorname{rem}(\alpha_{i+1} - \alpha_i, \pi) = (\alpha_{i+1} - \alpha_i) - \pi \operatorname{round}\left(\frac{\alpha_{i+1} - \alpha_i}{\pi}\right), \tag{3.1}$$

where α_i is *any* angle representation of $\ell(v_i)$. The index is then approximated by

$$\operatorname{ind}_\ell(\gamma) \coloneqq \frac{1}{2\pi}(\theta_n - \theta_1) = \frac{1}{2\pi}\sum_{i=1}^{n}(\theta_{i+1} - \theta_i) = \frac{1}{2\pi}\sum_{i=1}^{n}\Delta_i\,,$$

where the right hand side can be viewed as a discretization of the integral representation of the index in Equation (A.1). We will refer to ind_ℓ as the "computed index" whenever we wish to explicitly distinguish this from the true index, though we will not always make the distinction. We never have to pick an orientation θ_i for the line field at the vertices, but only compute the angle updates Δ_i via Equation (3.1) from any angle representation α_i; the latter is usually obtained by calling the `arctan` function on the line field components. Moreover, the value Δ_i only depends on the (directed) edge e_i and not on the rest of γ. Hence, Δ can be established as a function on the set of edges $\mathcal{V}$. This method is used in [44] for line field simplification by merging of singularities, where it is shown that for linear line fields this approach yields the correct index of an interpolated line-field on triangular meshes – even though these resolve the angle function θ by as few as three values. If γ encloses a region R (and is positively oriented), define $\operatorname{ind}_\ell(R) = \operatorname{ind}_\ell(\gamma)$.

From the definition of Δ_i, we know that it changes sign if the direction of e_i is reversed. This gives an additive property that is consistent with the additive property of the index.

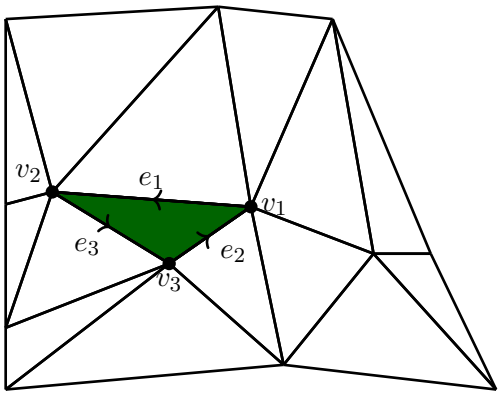
Lemma 3.2. *Let $P_1, \dots, P_k \in \mathcal{F}$ be k distinct faces so that $R = \bigcup_{i=1}^k P_k$. Then $\operatorname{ind}_\ell(R) = \sum_{i=1}^k \operatorname{ind}_l P_i$.*

Lemma 3.2 allows us to ignore faces with vanishing index from all considerations in the following.

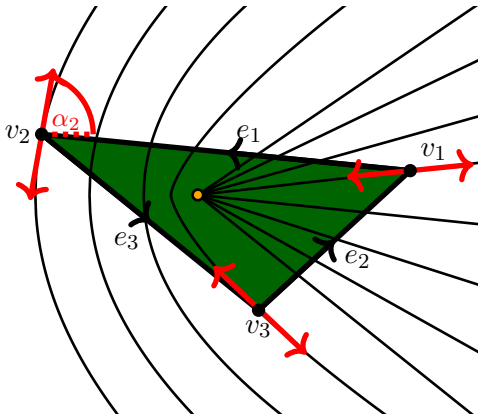
The method just described can also be interpreted as follows. Define the values of θ_i by taking the canonical metric on $\mathbb{P}^1$ as given by the angle between subspaces, and obtain a curve by connecting individual points by the shortest path in this metric (or equivalently by straight lines in the canonical embedding into $\mathbb{R}^2$). To compute the index, we then count the number of times this curve winds around the center of the circle representing $\mathbb{P}^1$, and divide by 2; cf. the (equivalent) definition of the line field index in [42, p. 218].

3.1.2. *Step (ii): Combining polygons*

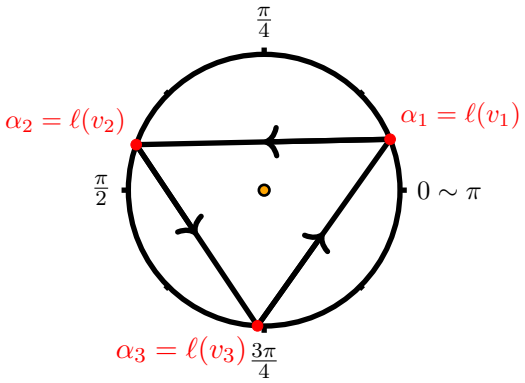
Let $\mathcal{F} = \{P_i;\ i \in I\}$ be the set of polygonal faces/grid cells of the mesh enumerated by an index set I. We identify each polygon $P_i \in \mathcal{F}$ with its center of mass p_i. In the following, the distance between faces P_i and P_j is taken as the distance between the centers of mass p_i and p_j, for simplicity. We wish to detect regions R that are elliptic and r-stable unions of polygons. We do so by finding connected



(A) A triangle in a mesh, with vertices v_1, v_2, v_3 and edges e_1, e_2, e_3 labelled.



(B) Same triangle, with a line field superimposed and values at vertices in red



(C) Values of the line field on $\mathbb{P}^1$, along with straight-line curve.

Figure 3.1. (a) Visualization of an irregular computational domain, a triangular mesh. (b) The line field ℓ and its values $\ell(v_1)$, $\ell(v_2)$, $\ell(v_3)$ around a triangle of the mesh. (c) The angle (mod π) representation $\alpha_i = \ell(v_i)$ and a connecting line of straight line segments. The values θ_i are (directed) arcs from α_i to α_{i+1}. As the curve goes around the center halfway, the line-field index is $\frac{1}{2}$, correctly indicating the enclosed wedge-type singularity.

components of an undirected graph $\mathcal{G}$ whose nodes are the center points $(p_i)_{i\in I}$ and whose faces P_i have non-vanishing index. In this graph, two nodes $p_i \neq p_j$ are connected if and only if $|p_i - p_j| < r$. By Lemma 3.2, the index of such a connected component $\widetilde{\mathcal{G}}$ is given by the sum of the non-vanishing indices $\sum_{i:\ p_i\in\widetilde{\mathcal{G}}} \mathrm{ind}_\ell(P_i)$.

Let $\mathcal{K}$ denote the set of such connected components. Any $K \in \mathcal{K}$ represents a set of faces $R_K := \cup_{k\in K} P_k$ whose index is, according to Lemma 3.2 given by the sum of the indices of P_k, $k \in K$. If these faces have $\mathrm{ind}_\ell(K) = 1$, then the corresponding region R_K is an elliptic region that is (approximately[2]) r-stable (cf. Figure 3.2), provided that the computed index approximates the true index well enough

[2]As we are working with center points and not exact distances, it may not be fully r-stable.

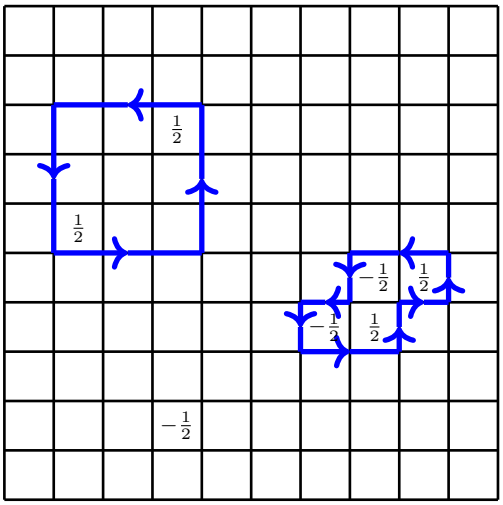


FIGURE 3.2. A quadrilateral mesh, with several r-stable regions (r equals, e.g., two cell diameters) containing singular cells. The upper left contour encloses an elliptic region, whereas the right contour contains 4 cells with non-vanishing indices which sum up to zero. Below, there is an isolated cell of index $-\frac{1}{2}$.

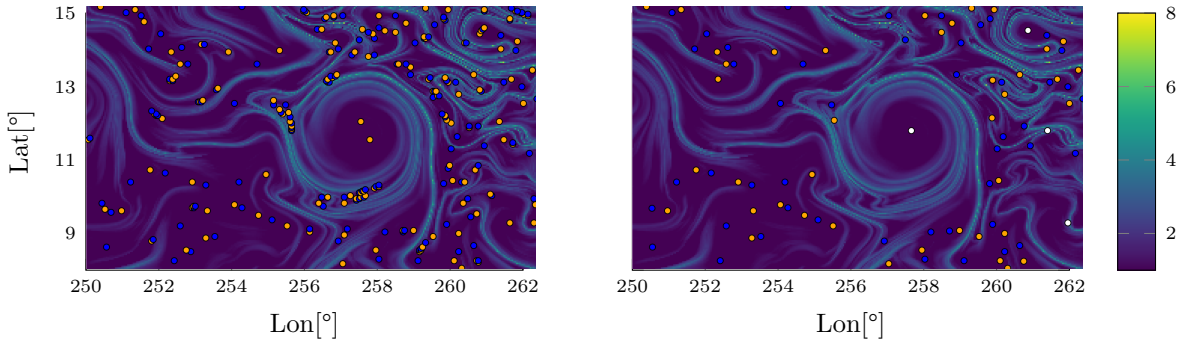
at the chosen coarseness of the polygonal mesh. For a region K, we will call the average over $(p_i)_{i\in K}$ its *center*.

3.1.3. *Step (iii): Additional merge heuristics*

As mentioned above, observations in [30] showed that the procedure in step (ii) may miss large elliptic regions, in which two wedge-type singularities (with index $\frac{1}{2}$) are further apart than r. Thus, we account for special elliptic configurations of r-stable regions by a range of merge heuristics. The simplest of these, which we call `combine_20`, adds to the list of elliptic regions all wedge pairs comprising r-stable regions that are mutually their nearest neighbors (measured by distance between center points) among the r-stable regions (of nonvanishing index). We have a similar heuristic for combining 3 wedge-type r-stable regions with a trisector (index $-\frac{1}{2}$) called `combine_31`. This seems useful in the OECS case but less so for other types of vortices. Additionally, we have also implemented a `combine_20_aggressive` heuristic that combines wedge-pairs under strictly weaker requirements. More specifically, it does so if (i) one of them is the nearest neighbor of the other; and (ii) if the rectangle with vertices given by the r-stable region centers does not contain any further r-stable region of nonzero index. This heuristic is based on examination of the singularity configurations occurring in our turbulence simulation described in Section 4.1. There, we also compare results from the `combine_20` heuristic with those obtained from the `combine_20_aggressive` heuristic. Further heuristics can be developed and neatly included in our implementation. Since the resulting regions only serve as *candidate* regions for the closed orbit computation described in Section 3.3, false-positives at worst add some computational effort.

3.1.4. *Discussion*

Since the above procedure forms one centerpiece of our implementation and earlier implementations of the same index-theory-based considerations have deservedly earned some criticism, we would like to discuss some of its features from a theoretical viewpoint here.



(A) Rectangles in a regular grid with non-vanishing indices.

(B) Centers of regions with nonzero index after the combination steps (with $r = 0.2°$).

Figure 3.3. Distribution of transport tensor $\mathbf{T}$ singularities and their types for a 30-day ocean surface simulation off the coast of Mexico starting on January 7, 2017: orange points correspond to centers of regions with index $\frac{1}{2}$, blue to index $-\frac{1}{2}$, and white to index 1. Background coloring is the DBS field.

First, the procedure described above is a rigorously justified singularity simplification procedure; cf. also [45]. If a singularity contained in a single mesh cell with non-vanishing index is reasonably isolated, merging its enclosing cell with neighboring cells results in a curve homotopy of the boundary which does not change the value of the index, but instead increases the number of "quadrature points" in the discretization of its integral representation; recall Equation (A.1) in Definition A.1. Therefore, it allows us to compute the index more accurately. If a singularity is accompanied by a very close second singularity, then their combined index is computed. This yields either (a) the cancellation of poorly computed/fake non-zero indices, or (b) the computation of the index of a larger region enclosing two singularities. Case (b) sometimes occurs in the center of closed orbits, where two wedges (each with index 1/2) or three wedges and a trisector (index $-\frac{1}{2}$) are nearby and get combined to a joint singularity with index 1, correctly indicating an elliptic region. Case (a) often occurs along the observed boundary of vortices, where many singularities cluster along a line. From a macroscopic perspective, however, the indices of these singularities turn out to cancel each other out, indicating that the net topological effect of this cluster is equivalent to the complete absence of singularities, or the presence of a single singularity.

These effects are shown in Figure 3.3. In Figure 3.3a, cells with nonzero index on a quadrilateral mesh (only wedges (index $\frac{1}{2}$, orange) and trisectors (index $-\frac{1}{2}$, blue)) without any combination steps taken are shown. One can clearly see the isolated wedge pair in the center of the figure, and dense singularity clusters aligned along quasi-one-dimensional strips. In Figure 3.3b, singularities are post-processed according to the above procedure. Here, (i) the isolated wedge pair is combined to an elliptic region (index 1, white), (ii) the dense singularity clusters have been annihilated under combination/index summation, and (iii) a densely clustered 3-wedge-1-trisector configuration in the upper right corner (caused in this particular case by the use of a low order interpolation scheme for the velocity field) has been combined to another elliptic region.

Finally, compared to the preceding (the computation of the tensor field $\mathbf{T}$ via advection of potentially dense grids of particles) and the subsequent computational steps (the computation of closed null-geodesics), the index computations here are negligible in terms of computational effort. This effort consists of:

(1) computing pairwise distances between points in the plane, though only distances between a *small* number (depending on the heuristic) of *nearby* points are required, which massively reduces the number of possible singularity combinations and allows to use a tree structure with `NearestNeighbors.jl` [5] instead of a distance matrix; and

(2) applying some simple logic/filtering on the resulting distance graph.

The possibility to obtain information on where closed null-geodesics might be located allows for the restriction to a small (in comparison to the total number of mesh vertices) number of local domains, which facilitates good scaling behavior when dealing with challenging flow problems on large domains.

3.2. **Restriction to local domains**

One crucial feature of the index-based computation of geodesic vortices is the possibility to localize the subsequent closed-orbit computation to regions of a physically reasonable size. This distinguishes our implementation from that developed in [40]. One immediate and significant effect of data localization is that interpolants do not have to carry global information, allowing for a better utilization of computer memory (and the memory hierarchy) in the subsequent steps. This is especially true when working with several processes, as the amount of data needing to be sent to individual processes is significantly reduced.

Another positive effect of localization is that integral curves which leave the localization box trigger an error in the ODE integrator and are hence no longer followed. Finding a reasonable criterion for automated decision-making when to give up on an integral curve was a rather challenging issue in previous implementations.

The side length $2R$ of the localization box is a parameter that must be supplied and that bounds the size of geodesic vortices that can be found. This parameter is used by our implementation to determine default values for other parameters as well. These include the required return distance for an integral curve to be considered as "closed" and error tolerances of the ODE solver. Each candidate region identified in the previous step can be processed with only local information about the tensor eigenvector ξ_1, ξ_2 and eigenvalue λ_1, λ_2 fields.

3.3. **Computation of closed orbits**

Let K be an elliptic region identified by the procedure described in Section 3.1. That procedure returns a point p_K (typically lying in the convex hull of K) which we view as the potential vortex center. To find closed orbits of $\eta^{\pm}_{\lambda}$ near K, we employ a shooting method and place a Poincaré section of length R from p_K eastwards to $p_K + (R, 0)$. In order to apply the shooting method, i.e., to numerically[3] compute integral curves, we need to turn the line fields into local vector fields.

3.3.1. *Orientation of the line field*

At each point x in the domain, the line field $\eta^{\pm}_{\lambda}(x)$ is identified with a vector $v^{\pm}_{\lambda}(x)$ so that $\mathrm{span}\{v^{\pm}_{\lambda}(x)\} = \eta^{\pm}_{\lambda}(x)$ and $|v^{\pm}_{\lambda}(x)| = 1$. Clearly, integral curves of $v^{\pm}_{\lambda}$ are integral curves of $\eta^{\pm}_{\lambda}$. As for the vector field representation, we choose an orientation such that ξ_1 is spiraling anti-clockwise around p_K, and ξ_2 is pointing away from p_K. This is achieved by setting

$$\tilde{\xi}_1(x) \coloneqq \mathrm{sign}\left(\xi_1(x) \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (x - p_K)\right) \xi_1(x)\,, \qquad \tilde{\xi}_2(x) \coloneqq \mathrm{sign}\left(\xi_2(x) \cdot (x - p_K)\right) \xi_2(x)\,,$$

[3]We use the `DifferentialEquations.jl` package [37].

and then

$$v_\lambda^\pm(x) = \sqrt{\frac{\lambda_2 - \lambda}{\lambda_2 - \lambda_1}}\,\tilde{\xi}_1(x) \pm \sqrt{\frac{\lambda - \lambda_1}{\lambda_2 - \lambda_1}}\,\tilde{\xi}_2(x)\,. \tag{3.2}$$

The local vector field can be interpreted as a *rotated vector field*; cf. [11]. With the previous orientation, increasing λ turns v_λ to the right (for v_λ^+) and to the left (for v_λ^-).

We calculate $v_\lambda^\pm$ at grid points and then interpolate. On a quadrilateral grid, this is done by bilinear interpolation followed by a slight rescaling to ensure that the interpolated value $v(x)$ has unit length. In cases where $v(x) \cdot v(y) < 0$ for adjacent grid points x, y the interpolated vector field will have sharp kinks, suggesting that the orientation for the vector field may have been chosen incorrectly. As a post-processing step, we check that all obtained closed orbits do not lie in such mesh cells. Moreover, Equation (3.2) is undefined when $\lambda \notin [\lambda_1, \lambda_2]$. In our implementation, we nevertheless generate a vector field at such points, and reject closed orbits a posteriori if $v_\lambda^\pm$ (and likewise also $\eta_\lambda^\pm$) has points in a cell in which $v_\lambda^\pm$ is undefined at some vertex. We also reject closed orbits in the unlikely case that they do not enclose the center point p_K.

3.3.2. *The shooting method*

After these preparations, the general approach is the following: Let $\mathrm{Ret}^\pm\colon [\lambda_{\min}, \lambda_{\max}] \times [0, R] \to \mathbb{R}$ denote the Poincaré return distance function for $v_\lambda^\pm$. This is the (signed) distance from an orbit of $v_\lambda^\pm$ starting at $p_K + (0, s)$ to its first return point on the Poincaré section. The function Ret is undefined at points s_0 whose orbit does not return, and in such cases, we assign $\mathrm{Ret}^\pm(s_0) \coloneqq \infty$. Geodesic vortices are given by the zeroes of $\mathrm{Ret}^\pm$.

Abstractly speaking, the problem is to find roots of Ret over the square $[\lambda_{\min}, \lambda_{\max}] \times [0, R]$. Previous implementations for computing geodesic vortices [33, 30, 22, 40] iterate over a fixed range of values $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ and aim to find zeros of $s \mapsto \mathrm{Ret}^\pm(\lambda, s)$.

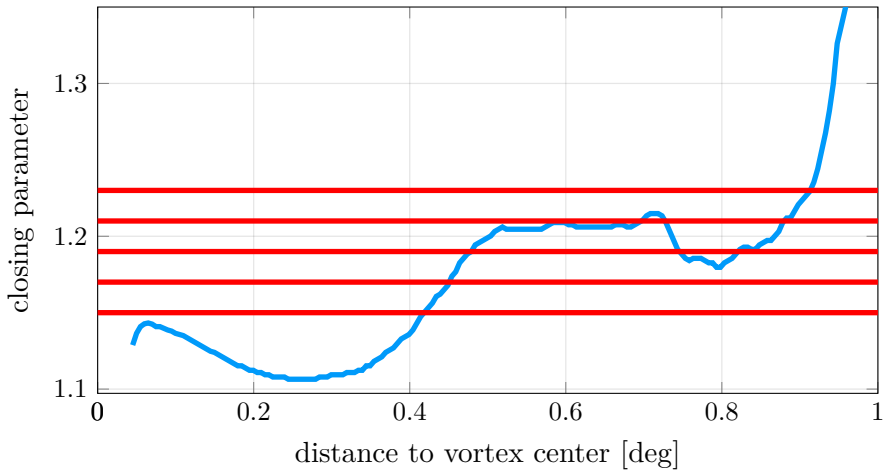
This has the disadvantage of often dealing with poorly conditioned problems. Figure 3.4 shows an example of the behavior of $\mathrm{Ret}^\pm(\lambda, \cdot)$ for various values of λ. In Figure 3.4a, we plot the parameter value $\lambda^*(s)$, for which integral curves close, over the initial condition $(0, s)$ along the Poincaré section. At local extrema, we observe a jump in the number of closed orbits for fixed λ. In Figure 3.4b, we look at the same situation from a different angle. Here, we plot the return distance function Ret again over the initial conditions. The bifurcations in Figure 3.4a correspond to tangencies of the curve with the zero level set $\mathrm{Ret} = 0$ (black) in Figure 3.4b.

We therefore employ a dual approach instead and look for roots of $\mathrm{Ret}^\pm(\cdot, s)$ for a fixed range of values $s \in [0, R]$, corresponding to a range of fixed initial conditions along the Poincaré section. This has the following advantages:

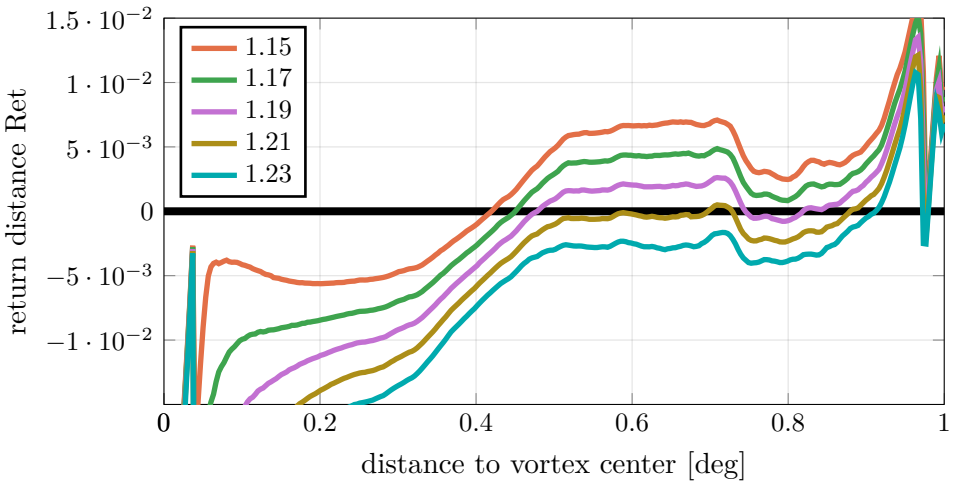
(1) The function $\mathrm{Ret}^\pm(\cdot, s)$ is—as long as the corresponding integral curves go through an annular region around the center singularities in which $v_\lambda^\pm$ is a continuous vector field (recall that the underlying vector field is rotating under parameter variation)—monotone (see Figure 3.5), greatly improving the condition of the problem.

(2) If we only want to find outermost geodesic vortex, we can start with a large s, i.e., at the very right of the Poincaré section, and decrease s until the first closed orbit is found.

(3) Closed orbits that are found tend to be uniformly spatially distributed; see Figure 3.6.



(A) The set of points on the Poincaré section for which a given closing parameter λ (red lines) corresponds to a closed orbit is difficult to determine for some values of λ



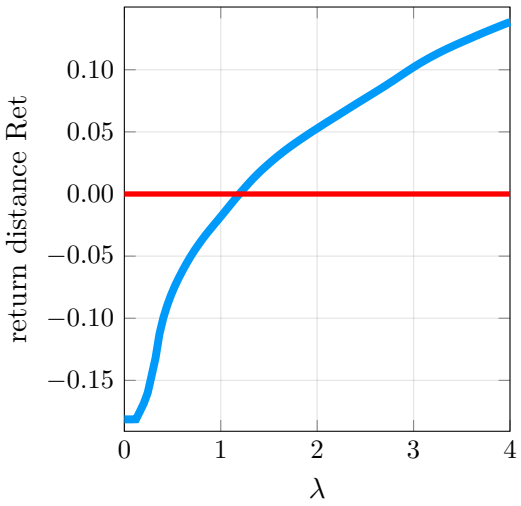
(B) The return distance function (with fixed parameter value λ), whose zeroes correspond to seeding points for closed orbits, may have tangencies with the zero-level set, and, as a consequence, the root finding problem is ill-posed.

FIGURE 3.4. Problematic behavior of $\mathrm{Ret}^{\pm}(\lambda, \cdot)$ with fixed λ.

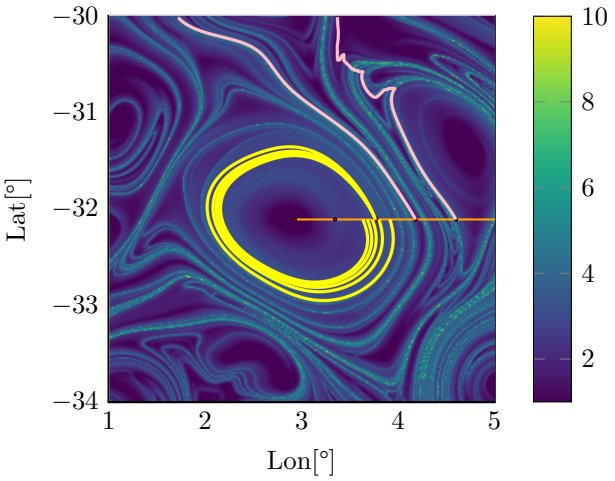
4. Applications with `CoherentStructures.jl`

Both test cases described below were run on a workstation[4] with 4 cores with a Linux (Fedora 27) operating system, except for the computations used to produce Figure 4.5, which was run on a 32-core compute server.

[4]Intel(R) Xeon(R) CPU E3-1235 @ 3.20GHz, 16GB RAM



(A) Return distance function Ret for some fixed initial condition and varying parameter λ.



(B) Yellow lines show integral curves for $\lambda = 0, 1, 2, 3, 4$. Pink lines are with $\lambda = 1$ and do not return.

Figure 3.5. Behavior of the function $\mathrm{Ret}^{\pm}(\cdot, s)$

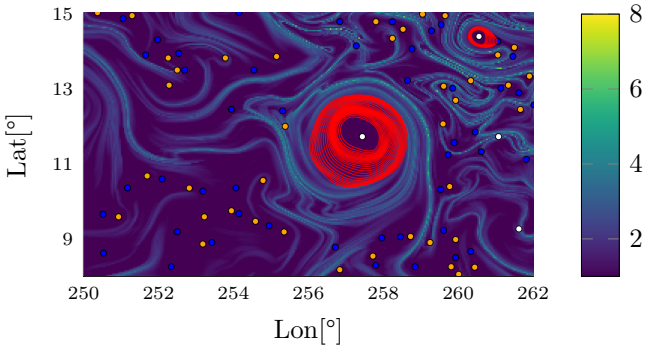


Figure 3.6. Figure 3.3b overlaid with the computed closed orbits.

4.1. **2D turbulence with varying initial time as parameter**

We calculate material barriers of an incompressible turbulent velocity field over a series of time windows $[t, t + 5]$ for 700 equally spaced values of t in $[0, 70]$. This is done in order to test our implementation on a large number of velocity fields without the option of manually adjusting parameters from one time window to the next.

For the convenience of the reader, we provide complete code—here and as a Jupyter notebook in the supplementary material—to reproduce our simulation. Unfortunately, the exact velocity field obtained changes from run to run, but the results should remain qualitatively the same. The velocity field is generated with the help of the packages `FourierFlows.jl` [46] and `GeophysicalFlows.jl` [8].

4.1.1. *Generating a turbulent velocity field*

We begin by importing the relevant packages and by setting up the computational domain.

```
using FourierFlows, GeophysicalFlows, GeophysicalFlows.TwoDTurb
mygrid = TwoDGrid(256, 2π)
x, y = gridpoints(mygrid)
```

To avoid decay of the flow we employ stochastic forcing. The code below is modified from the example given in the `GeophysicalFlows.jl` documentation.

```
ε = 0.001 # Energy injection rate
kf, dkf = 6, 2.0 # Waveband where to inject energy
Kr = [mygrid.kr[i] for i=1:mygrid.nkr, j=1:mygrid.nl]
force2k = @. exp(-(sqrt(mygrid.Krsq) - kf)^2 / (2 * dkf^2))
force2k[(mygrid.Krsq .< 2.0^2) .| (mygrid.Krsq .> 20.0^2) .| (Kr .< 1) ] .= 0
ε0 = FourierFlows.parsevalsum(force2k .* mygrid.invKrsq/2.0,
                              mygrid) / (mygrid.Lx*mygrid.Ly)
force2k .= ε/ε0 * force2k
function calcF!(Fh, sol, t, cl, args...)
    eta = exp.(2π * im * rand(Float64, size(sol))) / sqrt(cl.dt)
    eta[1,1] = 0
    @. Fh = eta * sqrt(force2k)
    nothing
end
```

We now setup the remaining parameters used in the simulation. We numerically solve the vorticity (transport) equation

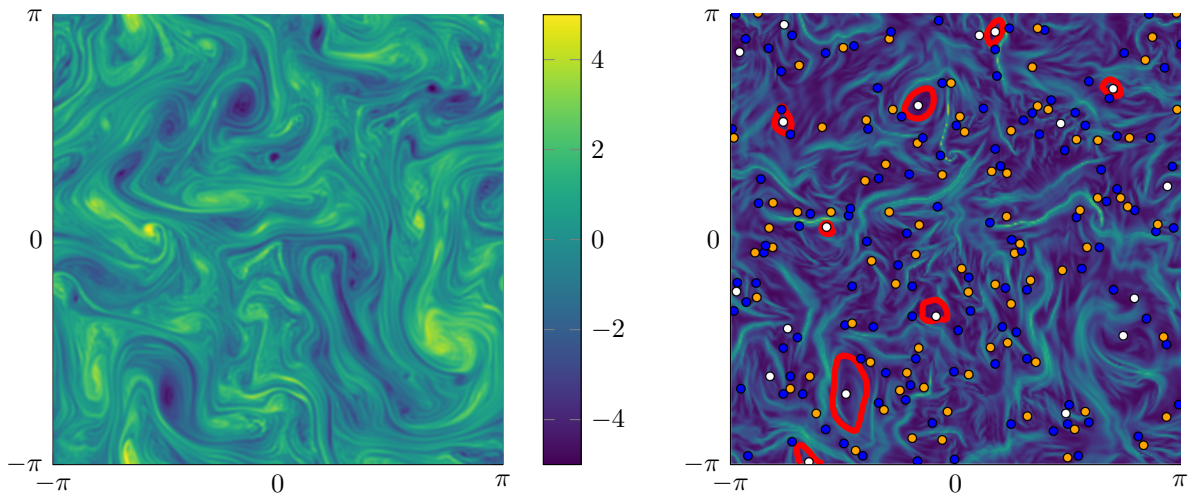
$$\partial_t \zeta = -u \cdot \nabla \zeta - \nu \zeta + f.$$

Here $u(x,y) = (u_1(x,y), u_2(x,y))^T$ is the (incompressible) velocity field, and $\zeta = \partial_x u_2 - \partial_y u_1$ is its vorticity. The parameter ν is set to 10^{-2} and is the coefficient of the drag term, f represents the forcing (see also the `FourierFlows.jl` package and its documentation [46], and [7]).

```
prob = TwoDTurb.Problem(nx=256, Lx=2π, ν=1e-2, nν=0, dt=1e-2,
    stepper="FilteredRK4", calcF=calcF!, stochastic=true)
TwoDTurb.set_zeta!(prob, GeophysicalFlows.peakedisotropicspectrum(mygrid,2,0.5))

using Distributed
addprocs()
using SharedArrays
# we use these variables to store the result
us = SharedArray{Float64}(256, 256, 400)
vs = SharedArray{Float64}(256, 256, 400)
zs = SharedArray{Float64}(256, 256, 400)
ts = Vector{Float64}(undef, 400)
```

We run this simulation until $t = 500.0$ to work in a statistically equilibrated state, and then save the result at time steps of size 0.2.



(A) (B)

Figure 4.1. (a) Vorticity at $t = 500$ in a turbulent two-dimensional velocity field. (b) Centers of regions with index $\frac{1}{2}$ in orange, those with index $-\frac{1}{2}$ in blue, centers of elliptic regions in white. Material barriers in red, background coloring is DBS field.

```
stepforward!(prob, round(Int, 500 / prob.clock.dt))
for i in 1:400
    stepforward!(prob, 20); TwoDTurb.updatevars!(prob)
    vs[:,:,i] = prob.vars.v
    us[:,:,i] = prob.vars.u
    zs[:,:,i] = prob.vars.zeta
end
```

The generation of the velocity field by the above code takes just a few minutes. Figure 4.1a shows the vorticity field for a specific run at $t = 500$.

4.1.2. *Computing material barriers*

We first setup a spatially periodic interpolation of the velocity field, which is performed by the package `OceanTools.jl` [38].

```
@everywhere using CoherentStructures, CopernicusUtils
const CS = CoherentStructures
const CU = CopernicusUtils
ts = range(0.0, step=20prob.clock.dt, length=400)
vdata = CU.ItpMetadata(xs, ys, ts, (us, vs), CU.periodic, CU.periodic, CU.flat)
```

We are now ready to compute material barriers.

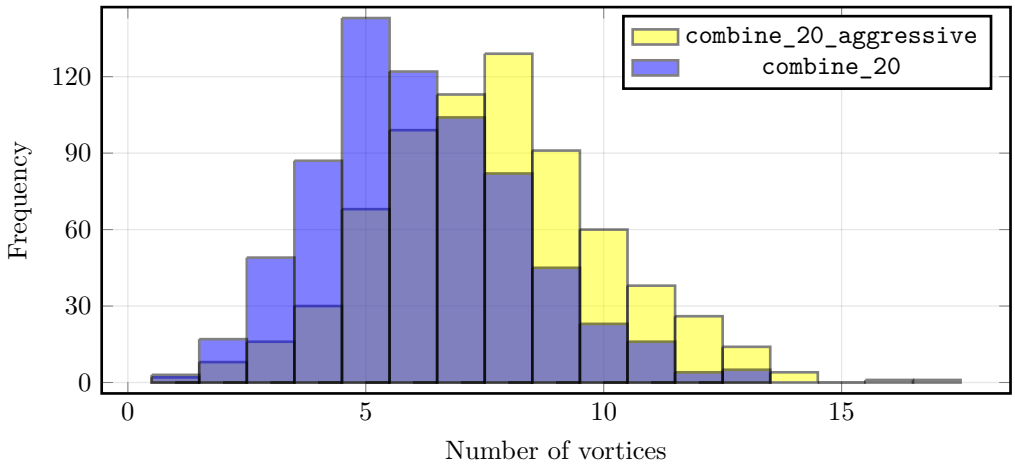


Figure 4.2. Comparison of combination heuristics for 700 different starting time parameters.

```
vortices, singularities, bg = CS.materialbarriers(
       uv_trilinear, xs, ys, range(0.0, stop=5.0, length=11),
       LCSParameters(boxradius=π/2, indexradius=0.1, pmax=1.4,
                     merge_heuristics=[combine_20_aggressive]);
       p=vdata, on_torus=true)
```

The `materialbarriers` function calculates the transport tensor field $\mathbf{T}$ used in the material-barriers approach (using finite differences for the linearized flow map $D\mathbf{F}$) and calculates material barriers. The result is shown in Figure 4.1b.

Running with 700 different values of t took 5h 16min 26 for the less agressive heuristic, and 8h 20min 9s for the more aggressive heuristic. Figure 4.2 shows a histogram of the number of vortices that have been found in the 700 simulations by the two merge heuristics; cf. Section 3.1.3. Clearly, the more aggressive merge heuristic detects more candidate regions and, as a consequence, more vortices. An animation containing the detected vortices over each time window is available in the supplementary material. The animation shows some flutter in the continuation of some vortices, especially close to their "generation" or "death", indicating room for further improvement in the robustness of the method.

4.2. **Global ocean surface flow**

As a very large-scale example, we compute material barriers to diffusive transport in a global ocean surface flow. The velocity field is obtained by tricubic interpolation (using the algorithm from [32] (cf. also [6, Appendix A]) as implemented in the `OceanTools.jl` package) of geostrophic ocean surface velocities from a dataset[5] distributed by the Copernicus Marine Environment Monitoring Service. These velocities are derived from satellite altimetry data. We choose a time window (starting November 26, 2016) for which a small spatial piece has been used repeatedly as a data set "benchmark" case in the literature; cf., for instance, [25, 15], and earlier studies on slightly larger domains [24, 30]. The domain studied in the first references is highlighted by a small white rectangle in Figures 4.3 and 4.4.

[5]More specifically, from the `SEALEVEL_GLO_PHY_CLIMATE_L4_REP_OBSERVATIONS_008_057` one.

We use a regular 10000×5000 quadrilateral grid, i.e., with 50 million grid points. Velocities in land-areas, or where ocean surface velocities are not available, are set to zero. Trajectories are calculated with `CoherentStructures.jl`, which internally uses the `DifferentialEquations.jl` package [37], and relative and absolute tolerances for the ODE solver are set to 10^{-6}. The averaged Cauchy–Green tensor is calculated by approximating the linearization of the flow map using finite differences, with the finite difference stencil reinitialized every 10 days. The resulting tensors are finally combined according to the product rule.

For the first test-case, we approximated the averaged, diffusion-weighted Cauchy–Green tensor for a 90-day period by averaging the Cauchy–Green tensor every 10 days from 0 to 90 days after start. The calculation of the tensors took approx. 21 hours. We find $40,609$ r-stable regions, $r = 0.25°$, of nonzero index and obtain $6,506$ elliptic regions with the `combine_20_aggressive` heuristic. The localization size R is set to 2.5°; cf. Section 3.2. This yields 357 vortices for λ-values in $[0.7, 1.4]$; the results are shown in Figure 4.3. The geodesic vortex computation took approx. 2.5 hours.

The second test case has an identical setup to the one described above, but with an observation time window of only 30 days. The computation of the tensors required approx 6.5 hours. We find $44,160$ r-stable regions of nonzero index, with $6,399$ candidate elliptic regions we obtained $2,255$ vortices. The results are shown in Figure 4.4, the geodesic vortex computation took again roughly 2.5 hours.

While we are not qualified to interpret Figures 4.3 and 4.4 from an oceanographic point of view, we do note that the region of known active vortex generation to the west of the southern tip of Africa due to Agulhas leakage [9] can be nicely recognized.

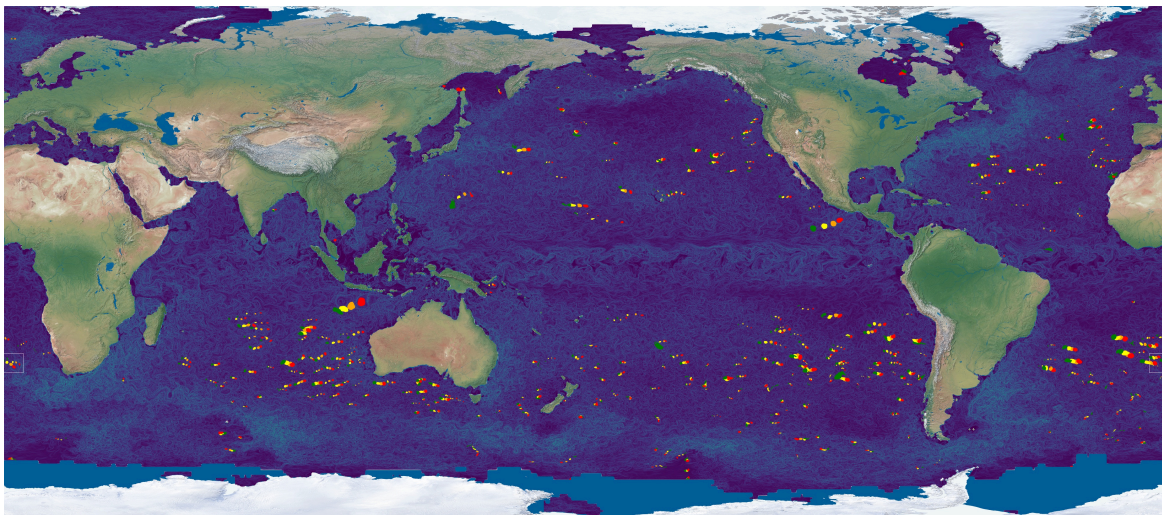
To further demonstrate the capabilities of our method, we have run 36 30-day simulations with a 30-day time lag, i.e., spanning a period of approximately 3 years. We then plot the initial coherent vortex locations superimposed onto each other with a (uniform) transparency in Figure 4.5. This visualization highlights regions where vortices tend to occur more often. Figure 4.5 is part of ongoing work on the statistics of coherent transport in the global ocean.

5. Conclusion

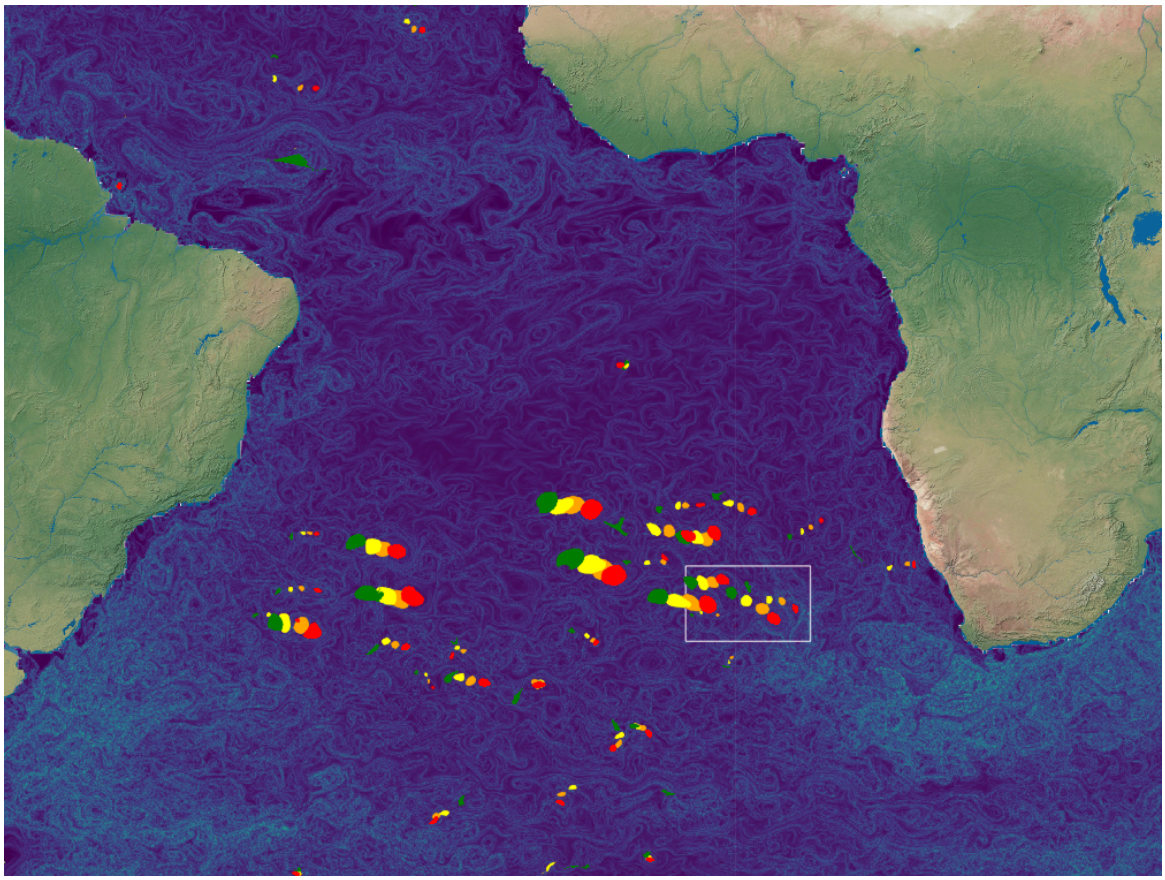
In this work, we have described in detail our implementation of the index-theory-based approach [30] to computing coherent Lagrangian vortices including material barriers to diffusive transport as closed null-geodesics of some Lorentzian metric tensor field. Despite the fact that there is no theoretical guarantee that all closed null-geodesics are found with the currently implemented merge heuristics, the approach is generally well-suited for genuinely challenging flow problems. This is due to the fact that while the effort for the computations related to the index-theory-based determination of candidate regions is basically negligible, its outcome dramatically reduces computational effort in the closed orbit detection via localization. In fact, there are cases (especially when reducing the number of grid points) where we do find *more* vortices with our implementation than with the implementation described in [40]. We suspect that this is because of the high degree of accuracy and resolution required in [40]. We have demonstrated our method on two test cases of considerable size, yet with affordable computational resources. Comparable simulation studies have, to the best of our knowledge and with the possible exception of [1], not been published before.

In this paper, we have restricted ourselves to presenting the theory and the implementation for computing closed null-geodesics of tensor fields. The exact same considerations with minor modifications carry over to the case of *constrained diffusion barriers*, as introduced in [26] as closed orbits of the *transport vector field*. The corresponding implementation building on the same machinery described here is available in `CoherentStructures.jl`.

When applied to such challenging problems, remaining issues become apparent, indicating room for further improvement. Interesting new features that are work in progress or under consideration

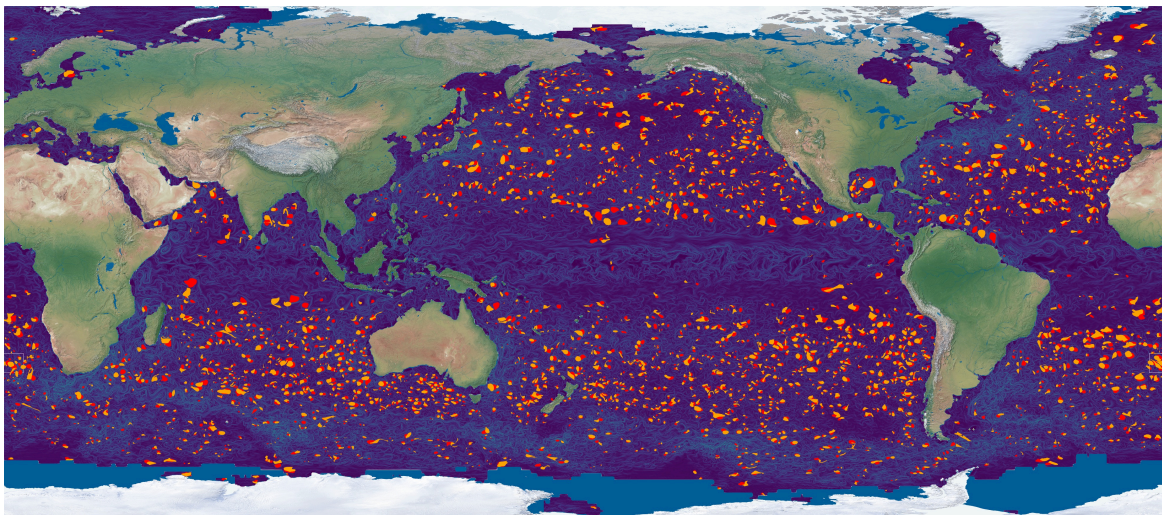


(a)

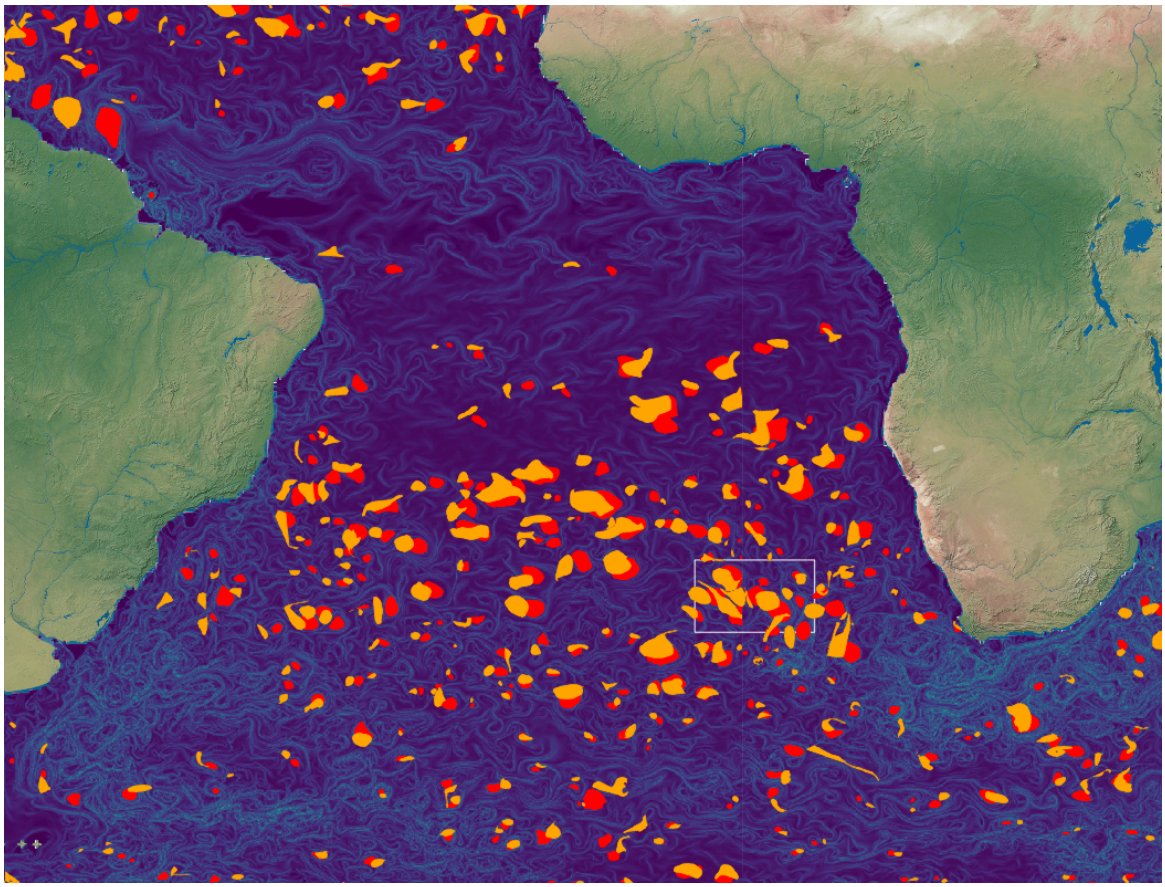


(b) Close-up of Figure 4.3a.

Figure 4.3. 90-Day DBS field with filled in material diffusive transport barriers (red), their advection after 30, 60 and 90 days (orange, yellow, green respectively).



(a)



(b) Close-up of Figure 4.4a.

Figure 4.4. 30-Day DBS Field with filled in material diffusive transport barriers (red), their advection after 30 days in orange.

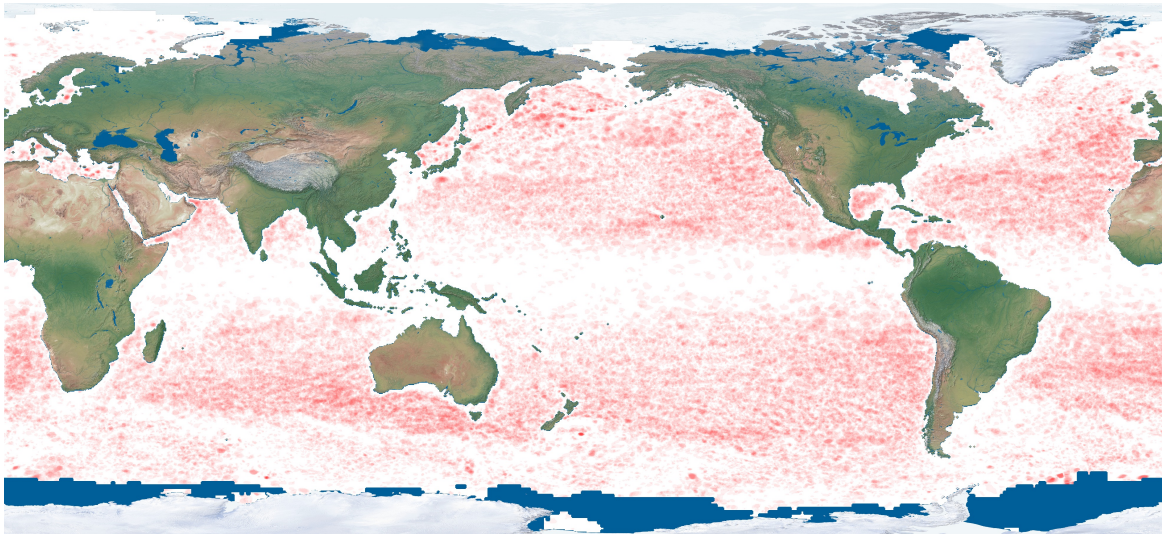


Figure 4.5. Coherent Lagrangian vortices from 36 30-day simulations, covering roughly 3 years. Every coherent vortex is shown in transparent red, with several spatially overlapping vortices increasing the color intensity.

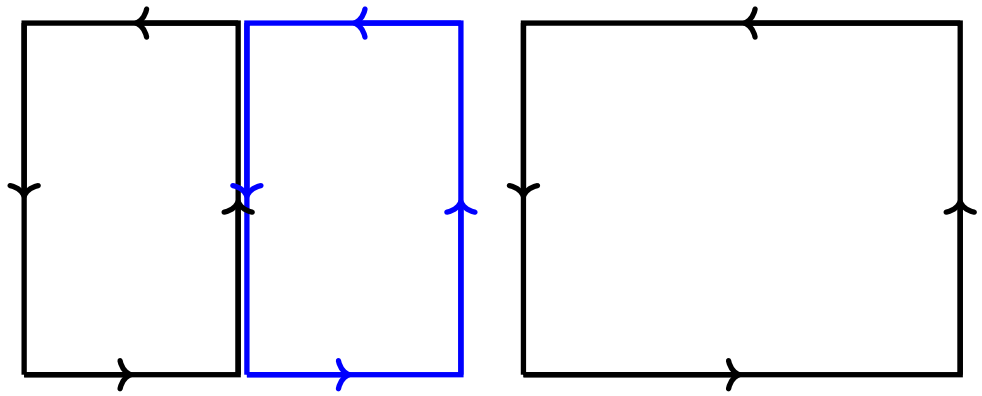
include the approximation of the tensor field $\mathbf{T}$ from scattered trajectories in order to have a purely data-driven approach for calculating coherent Lagrangian vortices, the use of boundary value problem solvers or topological methods à la [47] for the actual closed orbit detection, and other miscellaneous optimizations. In any case, our software, even in its current form, paves the way to global transport studies on the ocean surface from a Lagrangian viewpoint, which we hope may be of interest to the oceanographical community.

Appendix A. Index theory for planar line fields

We give a brief summary of some basic results from the classic index theory of planar line fields. For more details (and proofs connected to the classical theory), we refer the reader to [35] and [27] for the vector field and line field cases, respectively. We outline the theory for fields defined on an open, simply connected subset Ω of $\mathbb{R}^2$, but it can be easily extended to more general Riemannian manifolds by viewing Ω as the domain of a manifold parametrization. Notably, indices are independent of the specific Riemannian metric structure [27, Ch. 3, Thm. 1.4].

First, recall that a *Jordan curve* is a continuous function $\gamma\colon [a,b] \subset \mathbb{R} \to \Omega$ that is closed ($\gamma(a) = \gamma(b)$) and simple, i.e., it does not self-intersect. Such curves decompose Ω into three sets: the curve itself (the image of γ), its interior, and its exterior (which is unbounded if $\Omega = \mathbb{R}^2$ or touches the boundary of Ω otherwise).

A line field on a domain $D \subset \Omega \subset \mathbb{R}^2$ is a continuous map $\ell\colon D \to \mathbb{P}^1$, where $\mathbb{P}^1$ is the one-dimensional real projective space, i.e the set of all one-dimensional subspaces of $\mathbb{R}^2$. The real projective space can be given the structure of a compact, smooth manifold, and can be parametrized by an angle coordinate as follows: assign to any one-dimensional vector space V the angle between (any vector of) the subspace and the x-axis. This assignment is unique up to a multiple of π, taking into account the two possible orientations within V. We assume that ℓ is only defined on D and not on the whole of Ω due to the existence of singularities, see below.



(A) Two curves γ_1 (left, black) and γ_2 (right, blue).

(B) The sum $\gamma_1 + \gamma_2$.

FIGURE A.1. Addition of curves: segments that are traversed in opposing directions annihilate each other in the sum.

Definition A.1 (Line-field index of a curve)**.** Let γ be a Jordan curve in D, and θ be a continuous function giving the angle between $\ell(\gamma(s))$ and the x-axis, i.e., this means $\ell(\gamma(s)) = \operatorname{span}\{(\cos\theta, \sin\theta)^T\}$ for $s \in [a, b]$. Then the *index of γ relative to ℓ* is given by

$$\operatorname{ind}_\ell(\gamma) \coloneqq \frac{1}{2\pi}(\theta(b) - \theta(a)) = \frac{1}{2\pi}\int_\gamma \mathrm{d}\theta, \tag{A.1}$$

where, for a piecewise C^1-curve γ, $\mathrm{d}\theta$ is the derivative of the enclosed angle θ along γ.

Clearly, for $\operatorname{ind}_\ell(\gamma)$ to be defined, γ must not pass through points in $\Omega \setminus D$. The index of γ is invariant under orientation-preserving reparametrizations, and has the opposite signs under orientation-reversing reparametrizations. We can therefore identify γ with its image $\gamma([a, b])$, fixing its orientation. Moreover, the index is (i) invariant under homotopies, i.e., continuous deformations of γ in D, and (ii) additive under composition of curves in the sense that if γ_1, γ_2 are two closed curves and $\gamma = \gamma_1 + \gamma_2 \coloneqq \overline{(\gamma_1 \cup \gamma_2) \setminus (\gamma_1 \cap \gamma_2)}$ is again a closed curve (see Figure A.1), then

$$\operatorname{ind}_\ell(\gamma) = \operatorname{ind}_\ell(\gamma_1) + \operatorname{ind}_\ell(\gamma_2),$$

provided that each index is defined. We may also define an index for subsets of Ω whose boundaries are Jordan curves (in D) or finite disjoint unions of such subsets, the additivity of the index then extends to additivity under disjoint unions, and unions that are disjoint except for a common segment of a Jordan curve on each boundary.

Definition A.2 (Index of a region)**.** Let $R \subset \Omega$ be the interior of a (positively oriented) Jordan curve γ with values in D. Then we define the *index of R (relative to ℓ)* as

$$\operatorname{ind}_\ell(R) \coloneqq \operatorname{ind}_\ell(\gamma).$$

Definition A.3. A *closed orbit* of a line field $\ell\colon D \to \mathbb{P}^1$ is defined as a smooth curve $\gamma\colon [a, b] \to D$ which is

(1) closed: $\gamma(a) = \gamma(b)$, and

(2) an integral curve, i.e., $\gamma'(s) \in \ell(\gamma(s))$ for $s \in [a, b]$.

Since tangent vectors along closed orbits turn once, we have the following well-known property.

Theorem A.4. *The index of a closed orbit of ℓ relative to ℓ is* 1.

There may exist points in Ω at which a line field ℓ is not defined and, importantly, to which it cannot be extended continuously. Such points are referred to as *(line field) singularities*. As with critical points of vector fields, the angle θ in Equation (A.1) is not well-defined at singularities and hence Jordan curves must not pass through singularities for their index to be well-defined. If a curve γ encloses no singularities, then it must have index 0. The theorem above hence shows that closed orbits of line-fields must contain at least one singularity.

Definition A.5 (Line-field index of a region)**.** Let γ be an anticlockwise Jordan curve in D whose interior (with respect to Ω) is a set R. Then we define

$$\mathrm{ind}_\ell(R) \coloneqq \mathrm{ind}_\ell(\gamma).$$

Definition A.6. We call a set R with index 1 (with respect to ℓ) *elliptic (with respect to ℓ)*.

We note in passing that if a singularity p of ℓ is isolated, we can assign an index to it as the index of an arbitrary small region containing p. Therefore, if a region R contains finitely many isolated singularities, index additivity implies that the index of R is the sum of the indices of its enclosed singularities.

Clearly, every vector field v gives rise to a line field by applying the span operation at each point $x \in \Omega$, but the converse is not true in general: the obstruction comes exactly from line field singularities. For a line field $\ell = \mathrm{span}\, v$ induced by a vector field v, singularities correspond exactly to critical points of v, and, moreover, their indices w.r.t. v and ℓ coincide. Note, however, that in general the line field index is an element of the set of half-integers $\frac{1}{2}\mathbb{Z}$, in contrast to vector field indices that take only integer values. This shows that half-integer singularities form an obstruction to representing a line field ℓ as a vector field locally.

Appendix B. Applications of index theory to calculating geodesic vortices

Geodesic vortices are closed orbits of the line-field $\eta^{\pm}_{\lambda}$. By the theory outlined above, it follows that if γ is a geodesic vortex, it must have index 1 relative to $\eta^{\pm}_{\lambda}$. Geodesic vortices thus enclose elliptic regions (with respect to $\eta^{\pm}_{\lambda}$).

Following the approach in [30], we can continuously deform the line-field $\eta^{\pm}_{\lambda}$ to the line-field ξ_1, and continuity of the index (and the fact that it only takes half-integer values) implies that the index of γ relative to ξ_1 must also be 1. Therefore geodesic vortices enclose elliptic regions with respect to ξ_1. Changing the line field of reference has the following advantage: depending on the parameter value λ, $\eta^{\pm}_{\lambda}$ may not be defined in open regions, as opposed to ξ_1, which—for generic tensor fields—is defined everywhere except at isolated singularities; cf. [30, Sect. 4(b)].

Acknowledgements

This study has been conducted using E.U. Copernicus Marine Service Information. We wish to thank Navid Constantinou for technical help regarding the usage of the `GeophysicalFlows.jl` package, and Tom Patterson, `www.shadedrelief.com`, for providing the global Earth map that we have used in Section 4.2.

References

[1] R. Abernathey and G. Haller. Transport by Lagrangian Vortices in the Eastern Pacific. *J. Phys. Ocean.*, 48(3):667–685, 2018.

[2] M. R. Allshouse and J.-L. Thiffeault. Detecting coherent structures using braids. *Physica D*, 241(2):95–105, 2012.

[3] R. Banisch and P. Koltai. Understanding the geometry of transport: diffusion maps for Lagrangian trajectory data unravel coherent sets. *Chaos*, 27(3), 2017.

[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Rev.*, 59(1):65–98, 2017.

[5] K. Carlsson. NearestNeighbors.jl: High performance nearest neighbor data structures and algorithms for Julia, 2018. `https://github.com/kristofferc/nearestneighbors.jl`.

[6] M. Chilenski, I. Faust, and J. Walk. eqtools: Modular, extensible, open-source, cross-machine Python tools for working with magnetic equilibria. *Comput. Phys. Commun.*, 210:155–162, 2017.

[7] N. C. Constantinou. *Formation of large-scale structures by turbulence in rotating planets*. PhD thesis, National and Kapodistrian University of Athens, 2015.

[8] N. C. Constantinou and G. L. Wagner. FourierFlows/GeophysicalFlows.jl: GeophysicalFlows v0.3.0, 2019. `https://github.com/fourierflows/geophysicalflows.jl`.

[9] W. P. M. de Ruijter, A. Biastoch, S. S. Drijfhout, J. R. E. Lutjeharms, R. P. Matano, T. Pichevin, P. J. van Leeuwen, and W. Weijer. Indian-Atlantic interocean exchange: Dynamics, estimation and impact. *J. Geophys. Res. Oceans*, 104(C9):20885–20910, 1999.

[10] T. Delmarcelle and L. Hesselink. The topology of symmetric, second-order tensor fields. In *Proceedings of the conference on Visualization '94*, pages 140–147. IEEE, 1994.

[11] G. F. D. Duff. Limit-Cycles and Rotated Vector Fields. *Ann. Math.*, 57(1):15–31, 1953.

[12] M. Farazmand, D. Blazevski, and G. Haller. Shearless transport barriers in unsteady two-dimensional flows and maps. *Physica D*, 278-279:44–57, 2014.

[13] G. Froyland. An analytic framework for identifying finite-time coherent sets in time-dependent dynamical systems. *Physica D*, 250(0):1–19, 2013.

[14] G. Froyland. Dynamic isoperimetry and the geometry of Lagrangian coherent structures. *Nonlinearity*, 28(10):3587–3622, 2015.

[15] G. Froyland and O. Junge. Robust FEM-Based Extraction of Finite-Time Coherent Sets Using Scattered, Sparse, and Incomplete Trajectories. *SIAM J. Appl. Dyn. Syst.*, 17(2):1891–1924, 2018.

[16] G. Froyland, S. Llyod, and N. Santitissadeekorn. Coherent sets for nonautonomous dynamical systems. *Physica D*, 239(16):1527–1541, 2010.

[17] G. Froyland and K. Padberg-Gehle. A rough-and-ready cluster-based approach for extracting finite-time coherent sets from sparse and incomplete trajectory data. *Chaos*, 25(8), 2015.

[18] G. Froyland, Chr. P. Rock, and K. Sakellariou. Sparse eigenbasis approximation: Multiple feature extraction across spatiotemporal scales with application to coherent set identification. *Commun. Nonlinear Sci. Numer. Simul.*, 77:81–107, 2019.

[19] G. Froyland, N. Santitissadeekorn, and A. Monahan. Transport in time-dependent dynamical systems: Finite-time coherent sets. *Chaos*, 20(4), 2010.

[20] A. Hadjighasem, M. Farazmand, D. Blazevski, G. Froyland, and G. Haller. A critical comparison of Lagrangian methods for coherent structure detection. *Chaos*, 27(5), 2017.

[21] A. Hadjighasem and G. Haller. Geodesic Transport Barriers in Jupiter's Atmosphere: A Video-Based Analysis. *SIAM Rev.*, 58(1):69–89, 2016.

[22] A. Hadjighasem, D. Karrasch, H. Teramoto, and G. Haller. Spectral-clustering approach to Lagrangian vortex detection. *Phys. Rev. E*, 93, 2016.

[23] G. Haller and F. Beron-Vera. Geodesic theory of transport barriers in two-dimensional flows. *Physica D*, 241(20):1680–1702, 2012.

[24] G. Haller and F. Beron-Vera. Coherent Lagrangian vortices: the black holes of turbulence. *J. Fluid Mech.*, 731, 2013.

[25] G. Haller, D. Karrasch, and F. Kogelbauer. Material barriers to diffusive and stochastic transport. *Proc. Natl. Acad. Sci. USA*, 115(37):9074–9079, 2018.

[26] G. Haller, D. Karrasch, and F. Kogelbauer. Barriers to the Transport of Diffusive Scalars in Compressible Flows. submitted preprint, `https://arxiv.org/abs/1902.09786`, 2019.

[27] H. Hopf. *Lectures on differential geometry in the large*, volume 1000 of *Lecture Notes in Mathematics*. Springer, 1989.

[28] F. Huhn, W. M. van Rees, M. Gazzola, D. Rossinelli, G. Haller, and P. Koumoutsakos. Quantitative flow analysis of swimming dynamics with coherent Lagrangian vortices. *Chaos*, 25(8), 2015.

[29] I. Karatzas and S. Shreve. *Brownian Motion and Stochastic Calculus*, volume 113 of *Graduate Texts in Mathematics*. Springer, 1991.

[30] D. Karrasch, F. Huhn, and G. Haller. Automated detection of coherent Lagrangian vortices in two-dimensional unsteady flows. *Proc. A, R. Soc. Lond.*, 471(2173), 2015.

[31] D. Karrasch and J. Keller. A geometric heat-flow theory of Lagrangian coherent structures. `https://arxiv.org/abs/1608.05598`, 2016.

[32] F. Lekien and J. E. Marsden. Tricubic interpolation in three dimensions. *Int. J. Numer. Meth. Engng.*, 63(3):455–471, 2005.

[33] K. Onu, F. Huhn, and G. Haller. LCS Tool: A computational platform for Lagrangian coherent structures. *J. Comput. Sci.*, 7:26–36, 2015.

[34] K. Padberg-Gehle and C. Schneide. Network-based study of Lagrangian transport and mixing. *Nonlinear Process. Geophys.*, 24(4):661–671, 2017.

[35] L. Perko. *Differential Equations and Dynamical Systems*, volume 7 of *Texts in Applied Mathematics*. Springer, 3rd edition, 2001.

[36] W. H. Press and G. B. Rybicki. Enhancement of Passive Diffusion and Suppression of Heat Flux in a Fluid with Time Varying Shear. *Astrophys. J.*, 248:751–766, 1981.

[37] C. Rackauckas and Q. Nie. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *J. Open Res. Softw.*, 5(1):15, 2017.

[38] N. Schilling. OceanTools.jl. `https://github.com/CoherentStructures/OceanTools.jl`, 2020.

[39] M. Serra and G. Haller. Objective Eulerian coherent structures. *Chaos*, 26(5), 2016.

[40] M. Serra and G. Haller. Efficient computation of null geodesics with applications to coherent vortex detection. *Proc. A, R. Soc. Lond.*, 473(2199), 2017.

[41] M. Serra, P. Sathe, F. Beron-Vera, and G. Haller. Uncovering the Edge of the Polar Vortex. *J. Atmos. Sci.*, 74(11):3871–3885, 2017.

[42] M. Spivak. *A Comprehensive Introduction to Differential Geometry*, volume 3. Publish or Perish Inc., 3rd edition, 1999.

[43] J.-L. Thiffeault. Advection–diffusion in Lagrangian coordinates. *Phys. Lett. A*, 309(5–6):415–422, 2003.

[44] X. Tricoche and G. Scheuermann. Topology simplification of symmetric, second-order 2D tensor fields. In *Geometric Modeling for Scientific Visualization*, Mathematics and Visualization, pages 275–291. Springer, 2004.

[45] X. Tricoche, G. Scheuermann, H. Hagen, and St. Clauss. Vector and Tensor Field Topology Simplification on Irregular Grids. In D. S. Ebert, J. M. Favre, and R. Peikert, editors, *Data Visualization*, pages 107–116. Springer, 2001.

[46] G. L. Wagner, N. C. Constantinou, and M. Piibeleht. FourierFlows/FourierFlows.jl: FourierFlows v0.3., 2019. `https://zenodo.org/record/2530192`.

[47] T. Wischgoll and G. Scheuermann. Detection and visualization of closed streamlines in planar flows. *IEEE Trans. Visual. Comput. Graph.*, 7(2):165–172, 2001.