Numerical and Theoretical Developments for Coherent Structures

Albert J. Jarvis

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Engineering Mechanics

Shane D. Ross, Chair
Erik Bollt
Hosein Foroutan
Traian Iliescu

April 23, 2025
Blacksburg, Virginia

Keywords: Time-dependent Dynamical Systems, Lagrangian Coherent Structures, Material transport, Automatic differentiation.

Numerical and Theoretical Developments for Coherent Structures

Albert J. Jarvis

(ABSTRACT)

The field of nonautonomous dynamical systems has undergone an exceptional amount of growth over the last few decades, with the geometric viewpoint leading to the development of Lagrangian and objective Eulerian coherent structures. This theory extends notions developed for autonomous systems to those that may have arbitrary dependence on time and whose flow data may not be known or available for all time. The structures obtained from this theory are material surfaces that have an exceptional effect on the contents of a flow and are key for understanding transport. These structures organize the contents of a flow and lead to a qualitative description of the fate of material under the influence of a flow. While these methods have been adopted in geophysical and engineering applications, the theory is still relatively new. This dissertation aims to contribute to the maturation of this theory and its numerical implementations by providing an efficient and user-friendly software package for these tools, demonstrating the usefulness of these methods on a large-scale transport application, further extending the theory in a specific context, and presenting a new numerical framework for computing LCS from their variational theory.

Numerical and Theoretical Developments for Coherent Structures

Albert J. Jarvis

(GENERAL AUDIENCE ABSTRACT)

Understanding how some material moves through a fluid flow has wide-reaching applications in the physical sciences. Dust or smoke moving through the atmosphere, oil drifting through the ocean, or even a missing person carried by ocean currents can all be studied through the lens of material transport in a fluid flow. When the velocity field that describes the fluid flow does not change in time (time-independent), there is a rich theory dating back over a hundred years that defines key regions and structures in a flow, leading to a qualitative understanding of the fate of material being transported in that flow. However, when the velocity field changes with time (time-dependent) in a complex way, this classical theory falls short, as it was not developed with such flows in mind. These time-dependent flows are the rule rather than the exception in many geophysical applications (like those described above). Over the last few decades, a new theory has been developed to extend many of the notions of the time-independent setting to the more general time-dependent setting. This dissertation aims to contribute to the maturation of this theory and its numerical implementations by providing an efficient and user-friendly software package for these tools, demonstrating the usefulness of these methods on a large-scale transport application, further extending the theory in a specific context, and presenting a new numerical framework for computing certain versions of these structures.

Dedication

To my family and close friends, without whom I would not be where I am today.

Acknowledgments

I am thankful to many people – those who supported me through difficult times, those that taught me about life and mathematics, and those that did both.

To my committee: Thank you for serving on my committee and for the many thought-provoking discussions. I am grateful for your time, insight, and guidance.

To my MSU professors: Thank you for showing me the beauty of mathematics, for teaching me to think like a mathematician, and for putting up with all the shenanigans.

To my MSU colleagues: Thank you for helping me grow, making me laugh, and for partaking in the shenanigans.

To my VT professors: Thank you for teaching me things I never thought I'd have the opportunity to learn, for your time and patience, and for answering my countless questions.

To the Ross Dynamics Lab: Thank you for making the very painful process of getting a PhD just a little less painful, for all the thoughtful discussions, and for all the laughs.

To Eric Forgoston: Thank you introducing me to and teaching me about the field I have come to love, for helping me grow as a mathematician and a person, and for all the time you invested in me. Your impact on my life cannot be overstated.

To Shane Ross: Thank you for your unwavering support over the years, for everything you taught me, and for giving me the freedom to explore the questions that sparked my curiosity. You were a wonderful advisor, and I truly appreciate all that you've done for me.

To my friends: Thank you for being there for me, through good times and bad, and for brightening the dark days.

To my family: You have been the anchor of my life. I would not have made it a fraction as far as I did without your support. I am eternally grateful, and I love you all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding how material will be transported under the influence of a flow is of great importance in a variety of applications ranging from contaminant transport in a geophysical flow (an oil spill in the ocean [113, 114, 117] or dust containing unwanted pathogens moving through the atmosphere [75]) to developing search and rescue strategies for a person lost at sea [145]. When the underlying flow does not change with time, there is rich theory, originating in the late 1800s with Poincaré's work on celestial mechanics [122, 123], which allow one to obtain a full qualitative understanding of the long-term fate of particles advected by the flow. This theory relies on identifying important invariant objects (such as fixed points, corresponding stable and unstable manifolds, limit cycles, attractors, etc.) by looking at asymptotic properties of trajectories. These structures partition the state-space and provide a way to delineate dynamically distinct regions of the flow. Once these structures are obtained, a complete qualitative description of the fate of trajectories can be inferred.

When the underlying flow has general time-dependence, and may not be defined for all time, the asymptotics from which these structures are derived become ill-defined. This is precisely the setting we find ourselves in for many "real-world" geophysical and engineering flows. Over the last few decades, extending the notions from autonomous dynamical systems theory to the more general nonautonomous setting has received a great deal of attention and has seen a great deal of success[1]. Though many have contributed to this extension

[1]We note that there is a complementary viewpoint in dynamical systems which can be viewed as the

of the theory over the years, much of the pioneering work was done by various researchers at Cal Tech at the turn of the century (Haller, Shadden, Lekien, Marsden, Ross, etc.). Much of the variational theory was developed by George Haller and collaborators through his definition and refinement of Lagrangian coherent structures (LCS), and later by Serra and Haller through their definition of objective Eulerian coherent structures (OECS). These structures (which will be defined precisely below) can loosely be thought of as the finite-time and instantaneous analogues of the important invariant objects defined by Poincaré over a century ago. These structures are not invariant (unless viewed in the extended phase space) since they move through the state-space and come in and out of existence, but can provide similar qualitative information as their autonomous counterparts. This dissertation focuses on the numerical implementation of these methods, applying these methods to "real-world" scenarios, and contributing to further extending the theory.

For the remainder of this chapter, we mathematically set up the problem to be solved and cover the relevant theory which serves as the foundation upon which this dissertation is built.

1.1 Setup

Consider the following initial value problem, viewed as a dynamical system over some general n-dimensional smooth manifold M,

statistical or probabilistic point of view, known as ergodic theory. This too has a rich history, originating with Boltzman's work on statistical mechanics and the ergodic hypothesis, and later put on rigorous mathematical footing by Birkhoff and Von Neumann. Applying this viewpoint to nonautonomous systems has also seen a great deal of attention and success over the last few decades through important work on transfer operators by Froyland (Perron-Frobenius) and Mezić (Koopman), among others. This dissertation will not focus on this viewpoint.

$$\begin{aligned} \frac{d}{dt}\mathbf{x} &= \mathbf{v}(\mathbf{x},t), \qquad \mathbf{x}\in U\subset M, \quad t\in I\subset\mathbb{R} \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{aligned} \tag{1.1}$$

For the majority of this dissertation, $M=\mathbb{R}^2$, though for the sake of generality, we will often define things in terms of an arbitrary smooth manifold M. We require $\mathbf{v}(\mathbf{x},t)$ be uniformly Lipschitz continuous in $\mathbf{x}$ on U and continuous in t so that solutions exist and are unique $\forall\mathbf{x}\in U$ and $\forall t\in I$ (by the Picard-Lindelöf theorem [56]). Then, there exists a family of (at least local) diffeomorphisms $\{\mathbf{F}_{t_0}^t\}$ (known as the *flow maps*) associated with the dynamical system given by,

$$\begin{aligned} \mathbf{F}_{t_0}^t &: U\to U \\ &: \mathbf{x}_0\mapsto \mathbf{x}(t;t_0,\mathbf{x}_0) \end{aligned} \tag{1.2}$$

in which either $t>t_0$ (mapping forward in time) or $t<t_0$ (mapping backward in time). $\mathbf{F}_{t_0}^t$ maps initial positions to final positions under the action of the flow,

$$\mathbf{F}_{t_0}^t(\mathbf{x}_0) := \mathbf{x}_0 + \int_{t_0}^{t}\mathbf{v}(\mathbf{x}(s),s)ds \tag{1.3}$$

To gain insight into material transport, we look at the *linearized flow map* (often called the *deformation gradient*)

$$\begin{aligned} \nabla\mathbf{F}_{t_0}^t(\mathbf{x}_0) &: T_{\mathbf{x}_0}U\to T_{\mathbf{x}}U \\ &: \mathbf{u}_{\mathbf{x}_0}\mapsto \mathbf{u}_{\mathbf{x}} \end{aligned} \tag{1.4}$$

$\nabla \mathbf{F}_{t_0}^t(\mathbf{x}_0)$ acts on elements of the tangent space (tangent vectors) at $\mathbf{x}_0$ and maps them to elements of the tangent space downstream at $\mathbf{x} = \mathbf{F}_{t_0}^t(\mathbf{x}_0)$ (i.e., the deformation gradient maps material vectors to spatial vectors). Due to the conditions we impose on $\mathbf{v}(\mathbf{x}, t)$, it follows that $\nabla \mathbf{F}_{t_0}^t(\mathbf{x}_0)$ exists almost everywhere (by Rademacher's theorem [43]). We also require that $\det(\nabla \mathbf{F}_{t_0}^t(\mathbf{x}_0)) > 0$ (we will often require $\det(\nabla \mathbf{F}_{t_0}^t(\mathbf{x}_0)) = 1$ to impose incompressibility).

1.1.1 Example flow

Throughout the rest of this chapter we will cover the theory behind many coherent structure methods. We will use the double gyre equations as an example flow from which we will illustrate the effect these structures have on material moving with the flow. In subsequent chapters, we will see these methods applied to a variety of analytical and numerical flows (flows represented by numerical velocity data). The double gyre equations produce an ocean-like gyre flow which consists of two counter rotating gyres. When $t = 0$, there is an instantaneous seperatrix which acts as a boundary between the two gyres. As t varies, this seperatrix oscillates back and forth in a volume preserving manner. While this flow is time-periodic (and not aperiodic), it serves as a good test bed for many of these methods and is often used as a common benchmark. The double gyre is typically defined over the domain $U = [0, 2] \times [0, 1]$ with its velocity field given by

$$\mathbf{v}(x, y, t) = \begin{bmatrix} u(x, y, t) \\ v(x, y, t) \end{bmatrix} = \begin{bmatrix} -\pi A \sin(\pi f) \cos(\pi y) \\ \pi A \cos(\pi f) \sin(\pi y) \frac{df}{dx} \end{bmatrix} \tag{1.5}$$

where

$$f(x, t) = \epsilon \sin(\omega t) x^2 + (1 - 2\epsilon \sin(\omega t)) x$$

A determines the magnitude of velocity vectors, ϵ determines the magnitude of the oscillation, and ω determines the period of the oscillation. The parameters used in all examples in this chapter are $A = 0.1,\ \epsilon = 0.25,$ and $\omega = \frac{2\pi}{10}$. In Figure 1.1 we show the velocity field at a few select times for the given parameters. Notice the ω value we use results in a period of 10 units of time.

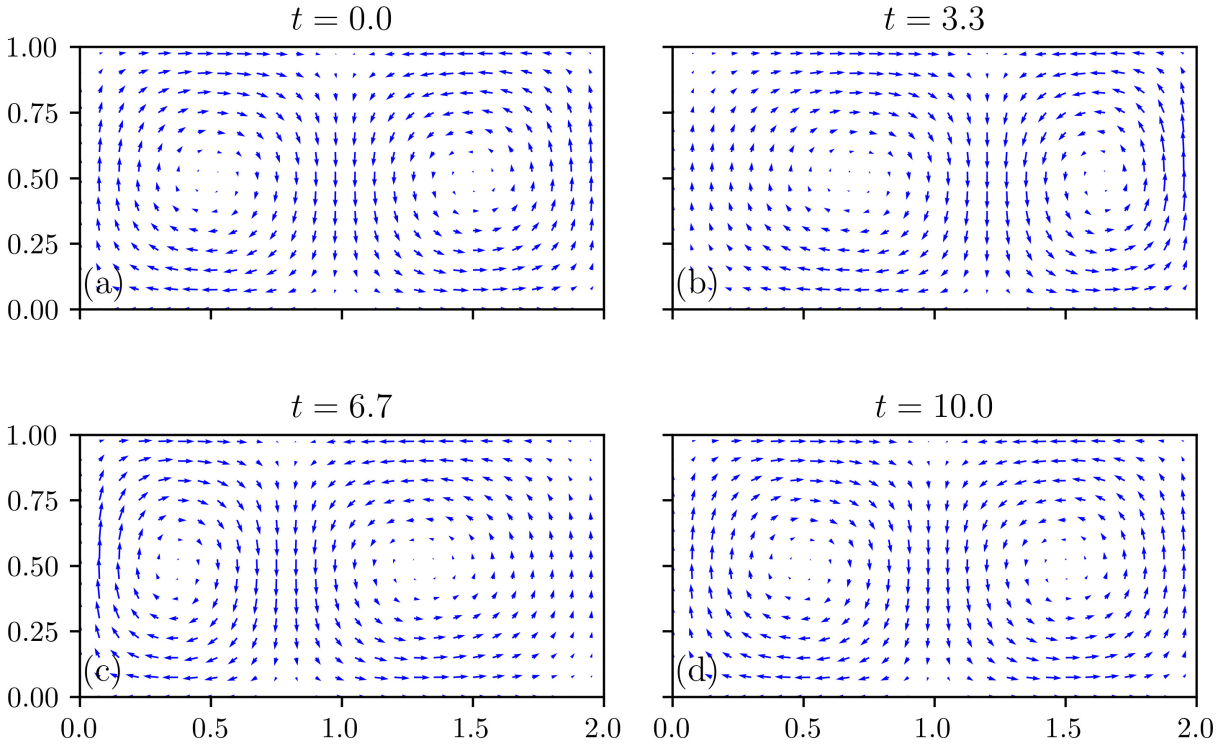

Figure 1.1: Velocity field for double gyre flow at $t =$ (a) 0.0, (b) 3.3, (c) 6.7, (d) 10.0.

1.2 Hyperbolic Lagrangian coherent structures

In the finite-time setting, we seek influential material surfaces (co-dimension 1) that show exceptional hyperbolic behavior relative to nearby material surfaces [65]. More precisely, we seek material surfaces that attract or repel nearby particles maximally, in a local sense.

These surfaces will act analogously to stable manifolds (repelling LCS) or unstable manifolds (attracting LCS) locally, over the the finite time window from which they were derived. We turn to the linearized flow map as the first step in defining these structures.

The linearized flow map encodes information about how infinitesimal perturbations to initial conditions grow under the influence of the flow and provides a linear approximation to the action of the flow map. Finding regions of the domain in which these infinitesimal perturbations grow or shrink the most can illuminate regions that will repel or attract nearby material in an extreme way. Following Shadden et al. [147], let $\mathbf{x}_0 + \delta\mathbf{x}_0$ be an infinitesimally nearby point to $\mathbf{x}_0$ at t_0 where $\delta\mathbf{x}_0 = \delta\mathbf{x}(t_0)$ is an arbitrarily oriented small perturbation to $\mathbf{x}_0$ s.t. $||\delta\mathbf{x}_0|| << 1$. Then, after time T, this perturbation evolves under the action of the flow map as,

$$\begin{aligned}\delta\mathbf{x}(t_0+T) &= \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0+\delta\mathbf{x}_0) - \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0) \\ &= \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0) + \nabla\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)(\mathbf{x}_0+\delta\mathbf{x}_0-\mathbf{x}_0) + \mathcal{O}(||\delta\mathbf{x}_0||^2) - \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0) \\ &= \nabla\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\delta\mathbf{x}_0 + \mathcal{O}(||\delta\mathbf{x}_0||^2)\end{aligned} \tag{1.6}$$

where we Taylor expanded $\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0+\delta\mathbf{x}_0)$ around $\mathbf{x}_0$. Since $\delta\mathbf{x}_0$ is an infinitesimal perturbation, the $\mathcal{O}(||\delta\mathbf{x}_0||^2)$ term is negligible. The magnitude can be computed by looking at the norm of this evolved perturbation

$$\|\delta\mathbf{x}(t_0+T)\| = \sqrt{\left\langle \nabla\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\delta\mathbf{x}_0, \nabla\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\delta\mathbf{x}_0 \right\rangle} \tag{1.7}$$

Using properties of the inner product, this can be rewritten as,

$$\|\delta\mathbf{x}_0(t_0+T)\| = \sqrt{\left\langle \delta\mathbf{x}_0, \left(\nabla\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\right)^* \nabla\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\delta\mathbf{x}_0 \right\rangle} \tag{1.8}$$

where $()^*$ denotes the conjugate transpose. Then,

$$\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0) = \left(\nabla \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\right)^* \nabla \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0) \tag{1.9}$$

is the right *Cauchy-Green strain tensor* which is both symmetric and positive-definite. This implies that $\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0)$ has real eigenvalues λ_i and corresponding orthonormal eigenvectors $\boldsymbol{\xi}_i$ with $i \in \{1, 2, ..., n\}$ such that,

$$\lambda_1 \geq ... \geq \lambda_n > 0 \text{ and,} \tag{1.10}$$

$$\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j \rangle = \delta_{ij}. \tag{1.11}$$

Now to find the direction in which an infinitesimal perturbation will grow the most, we note that Eq. (1.8) is maximized when $\delta\mathbf{x}_0$ aligns with $\boldsymbol{\xi}_1$, the eigenvector corresponding the the maximum eigenvalue of $\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0)$. Let $\delta\mathbf{x}_0^* = \|\delta\mathbf{x}_0\| \, \boldsymbol{\xi}_1$. Then,

$$\max_{\delta\mathbf{x}_0} \|\delta\mathbf{x}(t_0+T)\| = \sqrt{\langle \delta\mathbf{x}_0^*, \lambda_1 \delta\mathbf{x}_0^* \rangle} = \sqrt{\lambda_1} \, \|\delta\mathbf{x}_0\| \tag{1.12}$$

Therefore, for any $\mathbf{x}_0 \in U$, the maximum an infinitesimal perturbation to $\mathbf{x}_0$ will grow over the time window $[t_0, t_0+T]$ will be $\sqrt{\lambda_1}$ and this will occur when the direction of the initial perturbation aligns with $\boldsymbol{\xi}_1$.

For a perhaps more geometrically intuitive viewpoint, we note that the preceding result can equivalently be arrived at by looking at the SVD of the linearized flow map. Recall the spectral norm of a matrix $\mathbf{A}$ (equivalent to the induced vector 2-norm [73]) represents the maximum magnitude that $\mathbf{A}$ can stretch a vector by (and is given by the largest singular value). Therefore, to obtain the linear approximation to the action of the flow map on infinitesimal perturbations, we note that the most a perturbation will be deformed is given

by its largest singular value and this will happen when the initial perturbation aligns with the corresponding right singular vector. To see this, let $\nabla \mathbf{F}_{t_0}^{t_0+t}(\mathbf{x}_0) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ be the SVD of the linearized flow map where $\mathbf{U}, \mathbf{V} \in \mathrm{M}_n(\mathbb{R})$ are unitary (they are in fact orthogonal since $\nabla \mathbf{F}_{t_0}^{t_0+t}(\mathbf{x}_0)$ is real) and $\mathbf{\Sigma} \in \mathrm{M}_n(\mathbb{R})$ is a nonnegative diagonal matrix (it is in fact positive diagonal since $\nabla \mathbf{F}_{t_0}^{t_0+t}(\mathbf{x}_0)$ is nonsingular). Then, $\nabla \mathbf{F}_{t_0}^{t_0+t}(\mathbf{x}_0)$ (applied to a set of n orthogonal basis vectors of the tangent space represented by an infinitesimal n-sphere) can be viewed as an initial rotation (by $\mathbf{V}^*$), followed by pure scaling in the coordinate directions (by $\mathbf{\Sigma}$), and a final rotation (by $\mathbf{U}$) resulting in the final deformed ellipsoid. In this viewpoint, the linear approximation to the action of the flow map will cause specific initial vectors given by the the right singular vectors ($\boldsymbol{v}_i$) to grow (or shrink) according to the corresponding singular values (diag($\mathbf{\Sigma}$)) and deform into the left singular vectors ($\boldsymbol{u}_i$) scaled by those singular values. Therefore, the meaning of the eigenvalues and eigenvectors of $\mathbf{C}_{t_0}^{t_0+T}$ can equivalently be seen through the SVD of $\nabla \mathbf{F}_{t_0}^{t_0+T}$ by noting the singular values (diag($\mathbf{\Sigma}$)) are equal to the square root of the eigenvalues ($\sqrt{\lambda_i}$) of $\mathbf{C}_{t_0}^{t_0+T}$ and the right singular vectors ($\boldsymbol{v}_i$) are equal to the eigenvectors ($\boldsymbol{\xi}_i$) of $\mathbf{C}_{t_0}^{t_0+T}$. Therefore, the maximum argument made in Eq. (1.12) could equivalently be made with the leading right singular vector in Eq. (1.7). For $n = 2$, we show this in Figure 1.2.

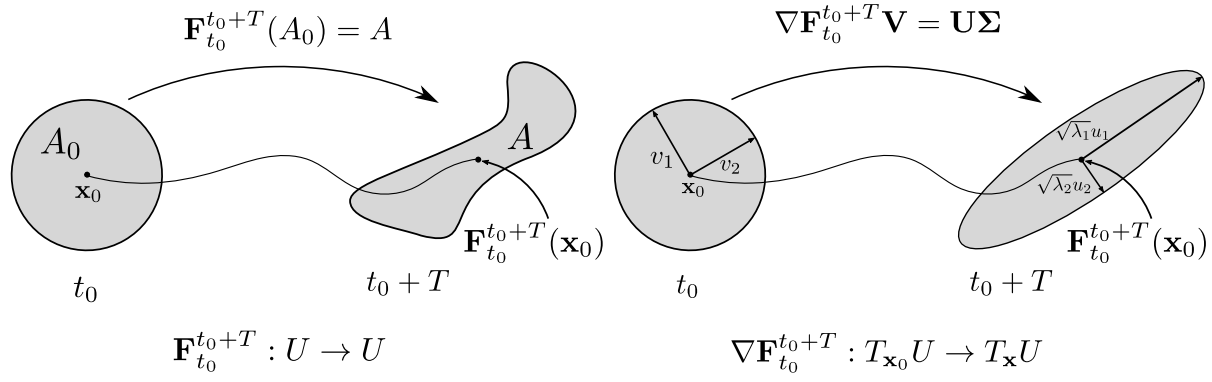

Figure 1.2: Behavior of flow map (left) and linearized flow map (right) on an infinitesimal set.

1.2.1 Finite time Lyapunov exponent

Note that Eq. (1.12) can be written as,

$$\max_{\delta \mathbf{x}_0} \|\delta \mathbf{x}(t_0+T)\| = e^{\sigma_{t_0}^{t_0+T}(\mathbf{x}_0)|T|} \|\delta \mathbf{x}_0\| \tag{1.13}$$

where

$$\sigma_{t_0}^{t_0+T}(\mathbf{x}_0) := \frac{1}{2|T|} \log(\lambda_1) \tag{1.14}$$

is the *finite-time Lyapunov exponent (FTLE)*, a finite-time version of the classic Lyapunov exponent. The FTLE at a point $\mathbf{x}_0$ will provide information about how nearby particles will behave over the time window of interest. Where there is high FTLE values, the flow will be very sensitive to perturbations to the initial condition. The FTLE is often computed for the entire spatial domain and can be computed both forward and backward in time. Regions of high FTLE reveal dominant repelling structures in forward time and dominant attracting structures in backward time (see Figure 1.3). This idea of using finite-time stretching to diagnose mixing and coherence was first proposed by Pierrehumbert [120, 121], later used in the original definition of LCS by Haller and Yuan [57], and further expanded on by Shadden et al. [147]. These works, among others, laid the foundation for the modern theory of hyperbolic LCS. Even today, FTLE is still the most common method used for hyperbolic LCS detection due to its efficiency and simplicity.

We make a note here that the requirements we put on $\mathbf{v}(\mathbf{x}, t)$ have been sufficient up to this point but in many of the subsequent arguments, additional smoothness will be required. From here on out we require that $\mathbf{v}(\mathbf{x}, t)$ be C^2 in space and C^1 in time.

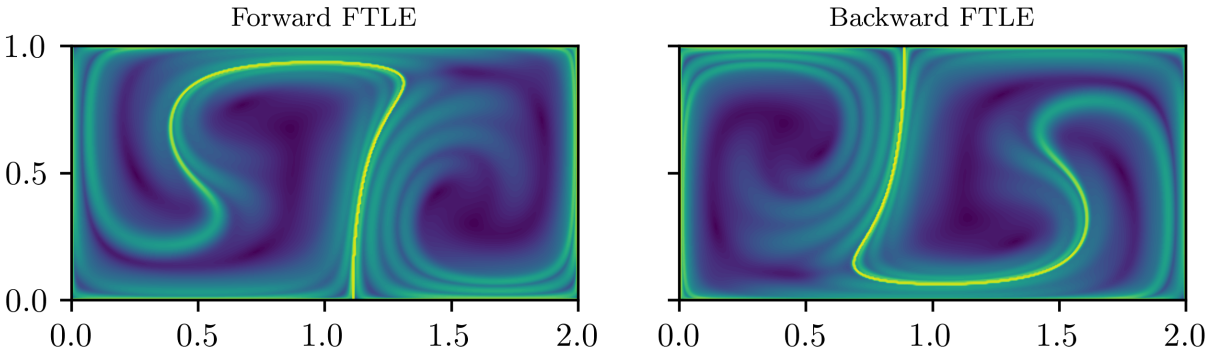

Figure 1.3: (a) Forward and (b) Backward FTLE for double gyre at $t = 0$ with integration time $|T| = 10$

1.2.2 FTLE ridges

In Shadden et al. [147], the authors proposed that LCS are ridges of the FTLE field. Arguable issues were eventually raised with this definition when a later variational formulation defined them as extremizing material curves of a certain functional [59] (covered below in Sect. 1.2.3). The main argument was that FTLE ridges are not true material curves and therefore not Lagrangian. Shadden et al. comment on this in their work and show that the material flux through the ridge is proportional to the inverse of integration time and more "well-defined" ridges tend to be more Lagrangian. This FTLE ridge definition provides a different school of thought than what will be defined later by Haller and arguments could be made in favor of either one. One clear advantage of the FTLE ridge method is that it is relatively simple to apply and extremely efficient – the same cannot be said for variational LCS as we will see later. Below, we detail the evolution of FTLE ridge extraction methods.

FTLE ridge extraction methods use differential geometric properties of the FTLE field (viewed as a surface in 3 dimensions) to extract the ridges of interest. These methods follow the methods developed in the image processing community designed to extract ridges from images [76, 93, 118, 148]. In image processing, images are usually converted to gray

scale and the gray values are essentially used as a discrete height function. Most methods suggest smoothing the data with a Gaussian, providing a scale-space representation. From here, first and second derivatives of this height function are computed, often with derivatives of the Gaussian, and these derivatives are used to extract differential geometric properties of the height field. These ridges are often referred to as height ridges (or sometimes, a *second-derivative ridge*). Given a height function f in 2 dimensions, *height ridge* points are defined as all $\mathbf{x}_0 \in U$ such that,

$$\begin{aligned} &1.\ \langle \nabla f, \boldsymbol{\eta}_{max} \rangle = 0; \\ &2.\ \langle \boldsymbol{\eta}_{max}, \mathbf{H}_f \cdot \boldsymbol{\eta}_{max} \rangle < 0 \end{aligned} \tag{1.15}$$

where $\mathbf{H}_f$ denotes the Hessian of f at $\mathbf{x}_0$, $\boldsymbol{\eta}_{max}$ is the eigenvector of the Hessian corresponding to the largest (in magnitude) eigenvalue (representing the direction normal to the ridge), and it is implied that all quantities are evaluated at $\mathbf{x}_0$. The first condition guarantees that the point is at an extrema of f (in the $\boldsymbol{\eta}_{max}$ direction) and the second, that f is concave down (in the $\boldsymbol{\eta}_{max}$ direction). Shadden et al. [147] first defined second-derivative FTLE ridges with these ideas in mind, using the FTLE field σ as the height field (though Haller first proposed the idea of using FTLE ridges as LCS [58]). Later Haller [59] defined FTLE ridges in a similar fashion to second-derivative ridges but required $\nabla \lambda_1$ be tangent to the ridge and replaced $\boldsymbol{\eta}_{max}$ with $\boldsymbol{\xi}_1$ in the second condition. Finally, Schindler et al. [134] further refined the criteria by using $\boldsymbol{\xi}_1$ in the first condition (they referred to these as *C-ridges*), resulting in the most common definition of *FTLE ridges* today,

$$\begin{aligned} &1.\ \langle \nabla \sigma, \boldsymbol{\xi}_1 \rangle = 0; \\ &2.\ \langle \boldsymbol{\xi}_1, \mathbf{H}_\sigma \cdot \boldsymbol{\xi}_1 \rangle < 0 \end{aligned} \tag{1.16}$$

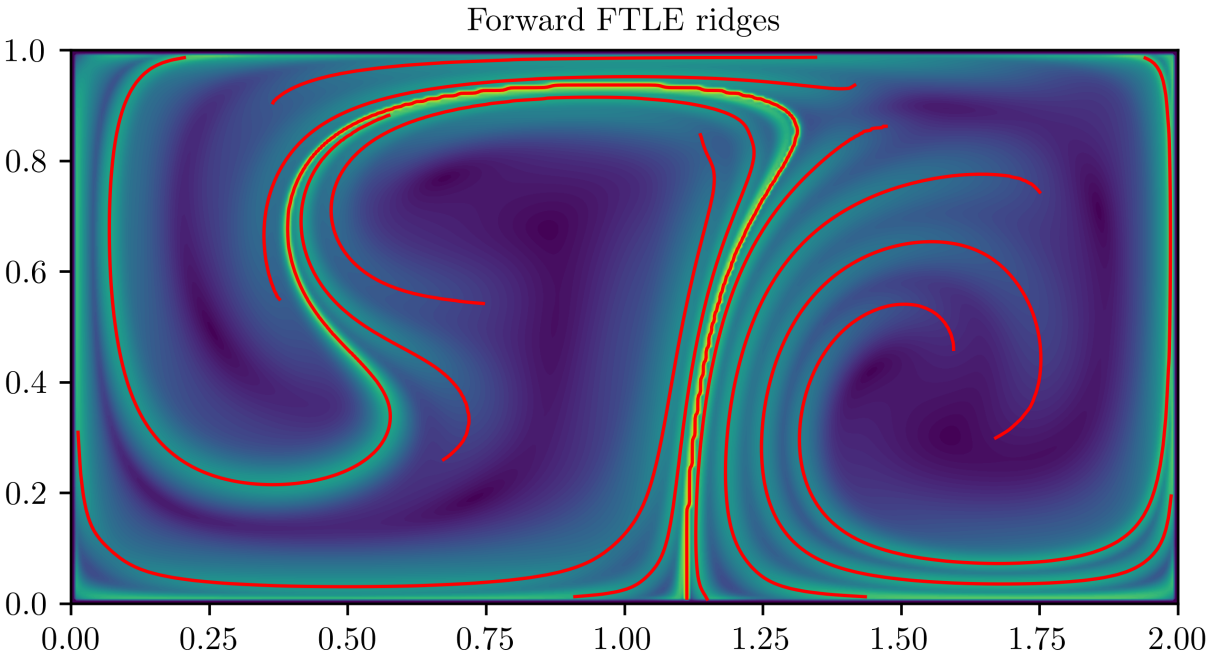

Figure 1.4: Forward FTLE ridges (red) overlaid on FTLE field for double gyre at $t = 0$ with integration time $T = 10$

An approach first proposed by Mathur et al. [105], and later revisited by Senatore and Ross [140] using a scale-space representation which they called *dynamical sharpening*, is another way to obtain accurate FTLE ridges that uses the fact the FTLE ridges are attractors of the gradient dynamical system given by,

$$\frac{d\mathbf{x}_0}{ds} = \nabla\sigma(\mathbf{x}_0). \tag{1.17}$$

This approach allows for accurate extraction of FTLE ridges that gets around some of the numerical issues with the previous definitions (numerical details will be covered in Ch. 2).

1.2.3 Variational Hyperbolic LCS

While the above method provides an efficient way to extract approximations of hyperbolic LCS, the extracted curves are often not truly Lagrangian (they are not material curves – the material flux through them is, in general, nonzero [59, 147]). In addition, FTLE may lead to false positives (due to regions of high shear, see [59]). In practice, the FTLE often captures the most influential structures. That being said, providing a precise definition of hyperbolic LCS and a numerical algorithm based off of this definition remained important to the development of the theory. Haller provides the precise definition in [59] and Farazmand and Haller provide a numerical implementation in [41]. Below we briefly cover the definition of hyperbolic LCS. Details about the numerical implementation are covered in Ch. (2). Here we present the relaxation of Theorem 1 from [41] as this is what is often implemented in practice. Assume we have a dynamical system given by Eq. (1.1) and the resulting Cauchy-Green tensor defined in Eq. (1.9) with eigenvalues and eigenvectors given in Eq. (1.11). Recall we seek material curves that repel or attract nearby fluid maximally in a local sense. For a given compact material curve $\mathcal{M}(t)$, let $\rho_{t_0}^{t_0+T}$ denote the *normal repulsion rate*, given by,

$$\rho_{t_0}^{t_0+T}(\mathbf{x}_0, \mathbf{n}_0) = \left\langle \mathbf{n}_{t_0+T}, \nabla \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\mathbf{n}_0 \right\rangle = \frac{1}{\sqrt{\left\langle \mathbf{n}_0, [\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0)]^{-1}\mathbf{n}_0 \right\rangle}} \tag{1.18}$$

with $\mathbf{n}_0$ and $\mathbf{n}_{t_0+T}$ denoting the normal to $\mathcal{M}(t)$ at $t = t_0$ and $t = t_0 + T$ respectively. Then, a *normally repelling material line* is a compact material curve over $\mathcal{M}(t)$ over $[t_0, t_0 + T]$ which satisfies,

$$\rho_{t_0}^{t_0+T}(\mathbf{x}_0, \mathbf{n}_0) > 1, \quad \rho_{t_0}^{t_0+T}(\mathbf{x}_0, \mathbf{n}_0) > \left| \nabla \mathbf{F}_{t_0}^{t_0+T} \mathbf{u}_{\mathbf{x}_0} \right| \tag{1.19}$$

for any $\mathbf{x}_0 \in \mathcal{M}_0$ and $\mathbf{u}_{\mathbf{x}_0} \in T_{\mathbf{x}_0}\mathcal{M}_0$. The first condition guarantees that $\mathcal{M}(t)$ is normally

repelling over the time window and the second guarantees the normal growth is larger than the tangential growth. Then, *hyperbolic LCS* are defined as compact material curves that are normally repelling material curves and obtain a maximum of the normal repulsion rate relative to nearby material curves. This is summed up in Theorem 1 from [41]. Consider a compact material line $\mathcal{M}(t) \in U$ evolving over the interval $[t_0, t_0 + T]$. Then $\mathcal{M}(t)$ is a *repelling LCS* over the $[t_0, t_0 + T]$ if an only if all the following hold for all initial conditions $\mathbf{x}_0 \in \mathcal{M}(t_0)$:

(A) $\lambda_2(\mathbf{x}_0) \neq \lambda_1(\mathbf{x}_0) > 1$;

(B) $\left\langle \boldsymbol{\xi}_1(\mathbf{x}_0), \nabla^2 \lambda_1(\mathbf{x}_0) \boldsymbol{\xi}_1(\mathbf{x}_0) \right\rangle \leq 0$;

(C) $\boldsymbol{\xi}_2(\mathbf{x}_0) || \mathcal{M}(t_0)$; (1.20)

(D) $\overline{\lambda}_1(\gamma)$, the average of λ_1 over a curve γ, is maximal on $\mathcal{M}(t_0)$
among all nearby curves γ satisfying $\gamma || \boldsymbol{\xi}_2(\mathbf{x}_0)$.

$\mathcal{M}(t)$ is an *attracting LCS* if these conditions hold for the backward time window $[t_0, t_0 - T]$. As covered in [41], condition (A) guarantees $\mathcal{M}(t)$ is a normally repelling material line. Condition (B) ensures that λ_1 achieves local maxima at $\mathbf{x}_0$ relative to nearby curves in the perpendicular direction (ensuring $\rho_{t_0}^{t_0+T}$ achieves a local maxima). Condition (C) ensures that the curve is normal to the maximum strain eigenvector field (recall $\boldsymbol{\xi}_1 \perp \boldsymbol{\xi}_2$)), and condition (D) picks out the strongest and most influential material curves by ensuring $\mathcal{M}(t)$ is the curve on which the averaged normal repulsion rate is maximal relative to nearby curves. Therefore, hyperbolic LCS are solution curves of the following ODE,

$$\mathbf{r}'(s) = \boldsymbol{\xi}_2(\mathbf{r}(s)), \quad |\boldsymbol{\xi}_2| = 1 \tag{1.21}$$

given they satisfy conditions (A), (B), and (D) (condition (C) is satisfied by Eq. (1.21)). These curves are often referred to as *shrinklines* in the repelling case (as their image under the flow will shrink drastically) and *stretchlines* in the attracting case (as their image under the flow will stretch drastically). Repelling LCS overlaid on the FTLE field for the double gyre are shown in Figure 1.5.

We note that what was presented above would be referred to as the "local variational theory" for hyperbolic LCS in which the variational problem sought extremizing curves of the normal repulsion rate. Haller and Beron-Vera proposed a global theory [67] in which all types of LCS are found as extremizing curves of the advected length functional and the type is determined by the proposed boundary conditions. Their results showed that the curves resulting from the global theory with hyperbolic boundary conditions agree with those from the local theory.

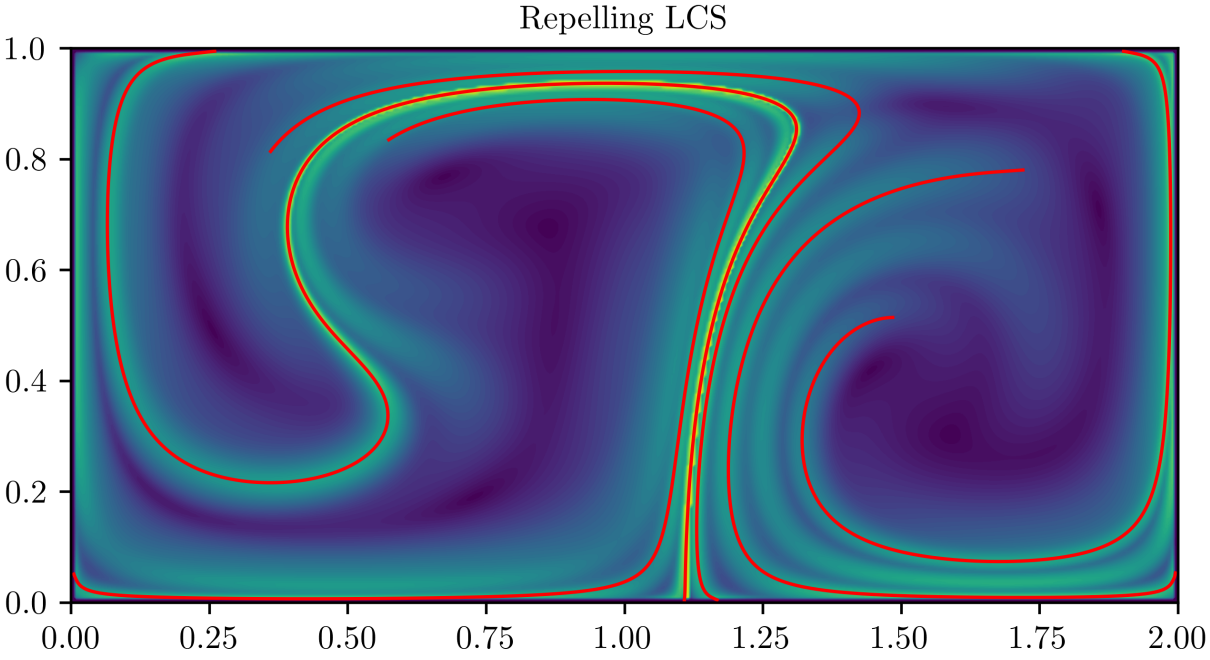

Figure 1.5: Repelling LCS (red) overlaid on FTLE field for double gyre at $t = 0$ with integration time $T = 10$

The influence of a repelling LCS on material moving with the flow can be seen in Figure

1.6 with a set of initial conditions (green circle) straddling a repelling LCS and another set of initial conditions (orange circle) away from a repelling LCS. Clearly the set of initial conditions straddling the LCS stretch, spread out, and mix with the rest of the flow while the set away from the LCS does not show such extreme behavior.

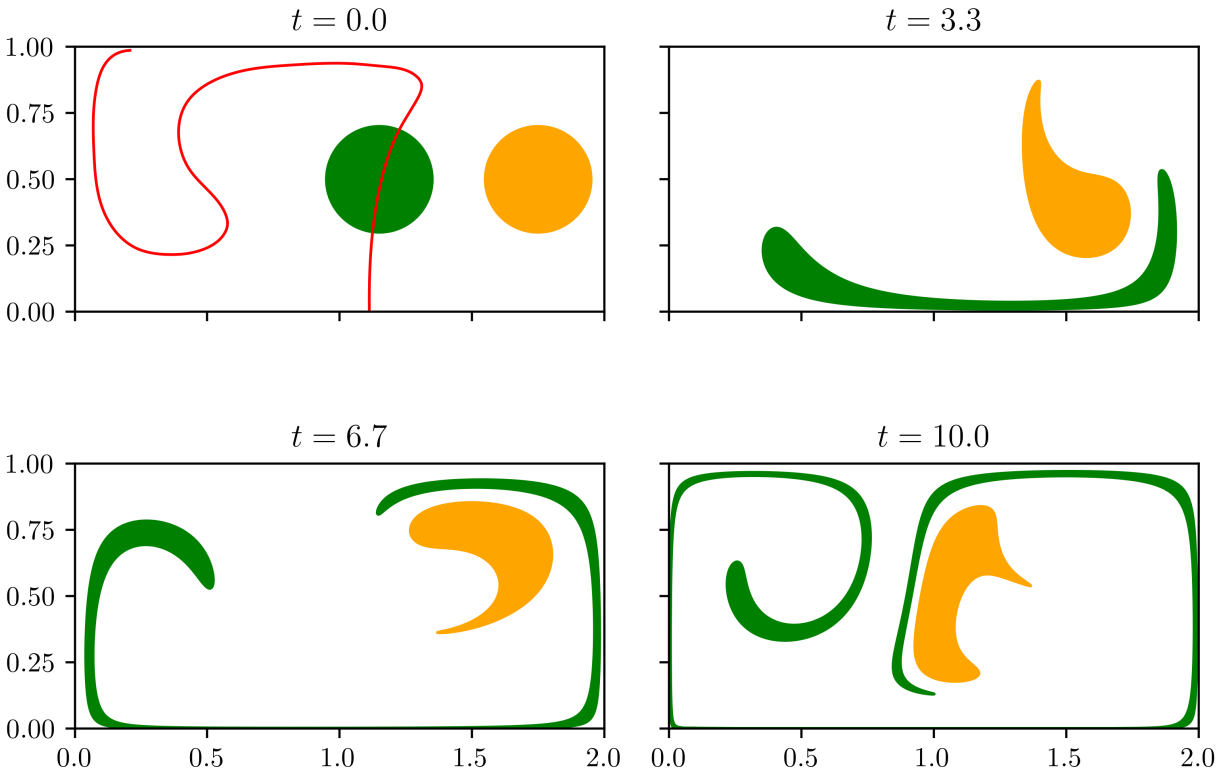

Figure 1.6: (a) Initial particles straddling (green) a repelling LCS (red) and initial particles away (orange) from a repelling LCS. Advected images of those initial sets at $t =$ (b) 3.3, (c) 6.7, and (d) 10.0 for double gyre with integration time $T = 10$. See video.

What conditions guarantee that a FTLE ridge is a hyperbolic LCS? Haller showed that the definition of an FTLE ridge leads to necessary (but not sufficient) conditions for hyperbolic LCS. He provided additional conditions which are sufficient for hyperbolic LCS detection [59] and later, Karrasch [79] further simplified these conditions in the case of differentiable eigenvectors. He showed that, given γ is a FTLE ridge as defined above, if all $\mathbf{x}_0 \in \gamma$ satisfy conditions A and C from the variational definition for hyperbolic LCS (Eq. (1.20)), then γ

is a hyperbolic LCS. Given that $\nabla\sigma$ approximates the tangent to the ridge and that $\boldsymbol{\xi}_1 \perp \boldsymbol{\xi}_2$, condition C is satisfied in an approximate sense and then the method employed simply needs to only consider points at which $\sigma > 0$ (equivalent to condition A).

1.3 Hyperbolic objective Eulerian coherent structures

All of the diagnostics and structures defined in what was presented up to this point were concerned with finite-time transport and tied to a specific time window $[t_0, t_0+T]$ (and thus, specific integration time). A natural question would be about what is obtained when one takes the limit of these structures as the integration time goes to zero. This idea was first explored by Serra and Haller [141] in defining objective Eulerian coherent structures (OECS) and later by Nolan et al. [112] when defining the instantaneous Lyapunov exponent (iLE). We review these works below.

1.3.1 Variational Hyperbolic OECS

Serra and Haller [141] define hyperbolic OECS in a similar manner to hyperbolic LCS. They are curves which extremize attraction or repulsion but this time, instantaneously rather than over some finite time window. We begin by introducing an important Eulerian tensor. Given our setup, let

$$\mathbf{S}(\mathbf{x}, t) = \frac{1}{2}\left(\nabla\mathbf{v}(\mathbf{x}, t) + (\nabla\mathbf{v}(\mathbf{x}, t))^\top\right) \tag{1.22}$$

be the Eulerian rate-of-strain tensor, i.e., the symmetric part of the gradient of the velocity. Since $\mathbf{S}$ is symmetric, it has real eigenvalues s_i and corresponding orthonormal eigenvectors $\mathbf{e}_i$ with $i \in \{1, 2, ..., n\}$ such that,

$$s_1 \geq ... \geq s_n \text{ and,} \tag{1.23}$$

$$\langle \mathbf{e}_i, \mathbf{e}_j \rangle = \delta_{ij}. \tag{1.24}$$

In a 2D incompressible flow, $s_1 = -s_2$. It can easily be shown that $\mathbf{S}$ is objective (see [55]). Much like $\mathbf{C}_{t_0}^{t_0+T}$ encodes information about how infinitesimally nearby particles will diverge, $\mathbf{S}$ encodes information about the rate at which infinitesimally nearby particles will diverge at the initial time. Also, note that for small integration time $|T| << 1$,

$$\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0) = \mathbf{I} + 2T\mathbf{S}(\mathbf{x}_0, t_0) + \mathcal{O}(T^2) \tag{1.25}$$

showing that for short times, the deviation of $\mathbf{C}_{t_0}^{t_0+T}$ from the identity is governed by $\mathbf{S}$ to leading order. Now let a material curve γ be parameterized by $\mathbf{r}(s)$ with $s \in [0, \sigma]$ s.t. $\gamma = \mathbf{r}(s)$ and $\mathbf{r}'(s)$ represents the tangent to γ at $\mathbf{r}(s)$. Then, Serra and Haller define two important material deformation rates, namely the material stretching rate

$$\dot{q}(\mathbf{r}(s), \mathbf{r}'(s), t) = \frac{\langle \mathbf{r}'(s), \mathbf{S}(\mathbf{r}(s), t)\mathbf{r}'(s) \rangle}{\langle \mathbf{r}'(s), \mathbf{r}'(s) \rangle} \tag{1.26}$$

and the material shear rate

$$\dot{p}(\mathbf{r}(s), \mathbf{r}'(s), t) = \frac{\langle \mathbf{r}'(s), [\mathbf{S}(\mathbf{r}(s), t)\mathbf{R} - \mathbf{R}\mathbf{S}(\mathbf{r}(s), t)]\mathbf{r}'(s) \rangle}{\langle \mathbf{r}'(s), \mathbf{r}'(s) \rangle} \tag{1.27}$$

where $\mathbf{R} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ is a 90° rotation matrix. For the hyperbolic case, we focus on the material shear rate. The averaged material shear-rate is given by

$$\dot{P}_t(\gamma) = \frac{1}{\sigma} \int_\gamma \dot{p}(\mathbf{r}(s), \mathbf{r}'(s), t) ds \tag{1.28}$$

By looking for curves on which the first variation vanishes (i.e., $\delta \dot{P} = 0$) and applying specific boundary conditions, Serra and Haller define hyperbolic OECS as curves which extremize the material shear-rate. They show that these curves are solutions to the differential equations

$$\mathbf{r}'(s) = \mathbf{e}_i(\mathbf{r}(s)), \quad i = \{1, 2\} \tag{1.29}$$

where the $\mathbf{e}_i$ are the eigenvectors of the Eulerian rate-of-strain tensor. *Attracting OECS* are defined as solution curves obtained from the $\mathbf{e}_1$ field that contains a local minima of the s_2 field but contains no other local minima of s_2. Similarly, *repelling OECS* are defined as solution curves of the $\mathbf{e}_2$ field that contains a local maxima of the s_1 field but contains no other local maxima of s_1. As mentioned, in the case of 2D incompressible flow, $s_2 = -s_1$ and the extrema will coincide. In this case, structures of particular interest tied to these extrema points are referred to as *objective saddle points* (generalizing the notion of classic saddle points from time-independent flows) and highlight cores of short term hyperbolic behavior. In Figure 1.7 we show the objective saddles for the double gyre flow at $t = 0$. The red and blue curves correspond to the repelling and attracting OECS respectively. It can be seen that the small circle around the objective saddle quickly is attracting to the attracting OECS, highlighting short term hyperbolic behavior. The numerical implementation is covered in Ch. 2.

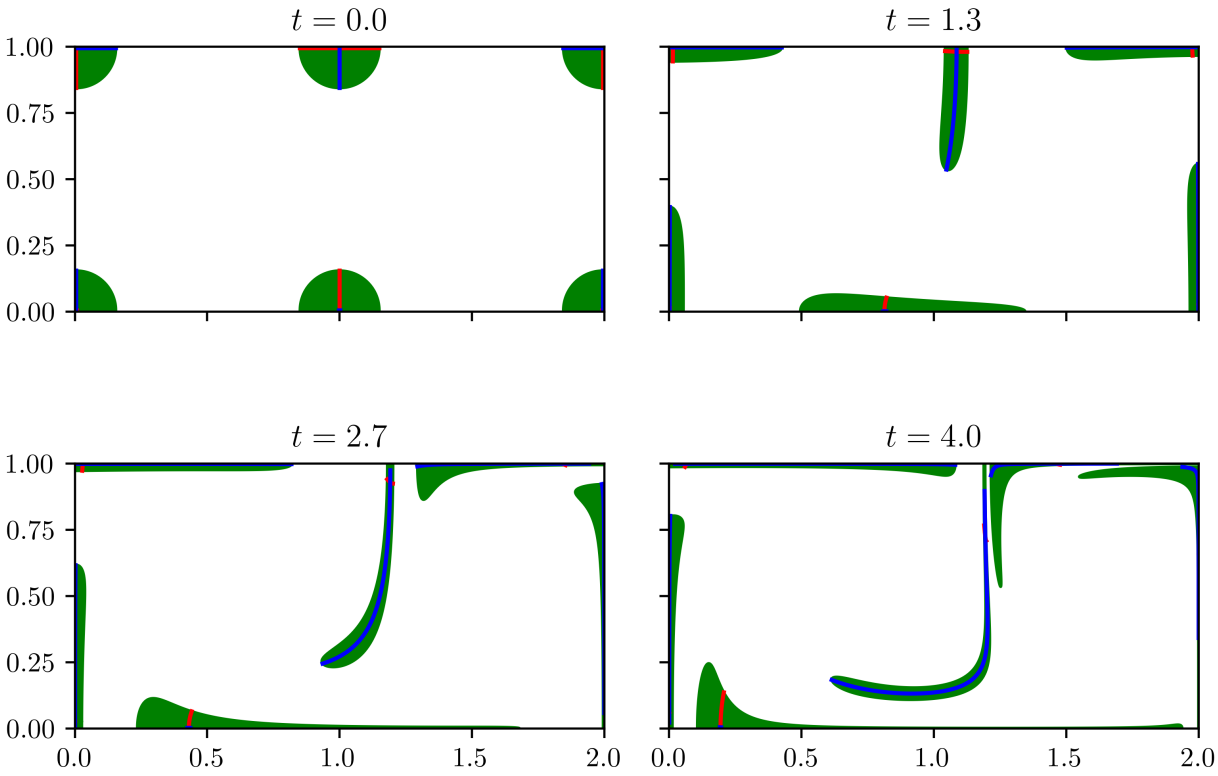

Figure 1.7: Repelling (red) and attracting (blue) OECS and initial points on the objective saddle at $t =$ (a) 0.0, (b) 3.3, (c) 6.7, and (d) 10.0 for double gyre with $t_0 = 0$

1.3.2 Instantaneous Lyapunov exponent

Shortly after the work on OECS, Nolan et al. [112] defined the *instantaneous Lyapunov exponent (iLE)* as the limit of FTLE as the integration time goes to zero. They do this by Taylor expanding the FTLE and show that, to leading order, the FTLE is governed by the iLE for short integration times. Therefore, they define the iLE as,

$$\lim_{T \to 0^{\pm}} \sigma_{t_0}^{t_0+T}(\mathbf{x}_0) = \pm s_{\pm}(\mathbf{x}_0, t_0) \tag{1.30}$$

where $s_+ = s_2$, $s_- = s_1$ and the superscript $\pm$ on 0 denotes whether the limit is from above $(+)$ or below $(-)$. Much like FTLE, the iLE provides a diagnostic field that highlights

extreme regions of instantaneous attraction and repulsion. They go on to define iLE ridges in essentially the same manner as we defined FTLE ridges in sec. (1.2.2). In Figure 1.8 we show the iLE field with the objective saddles from the last section overlaid

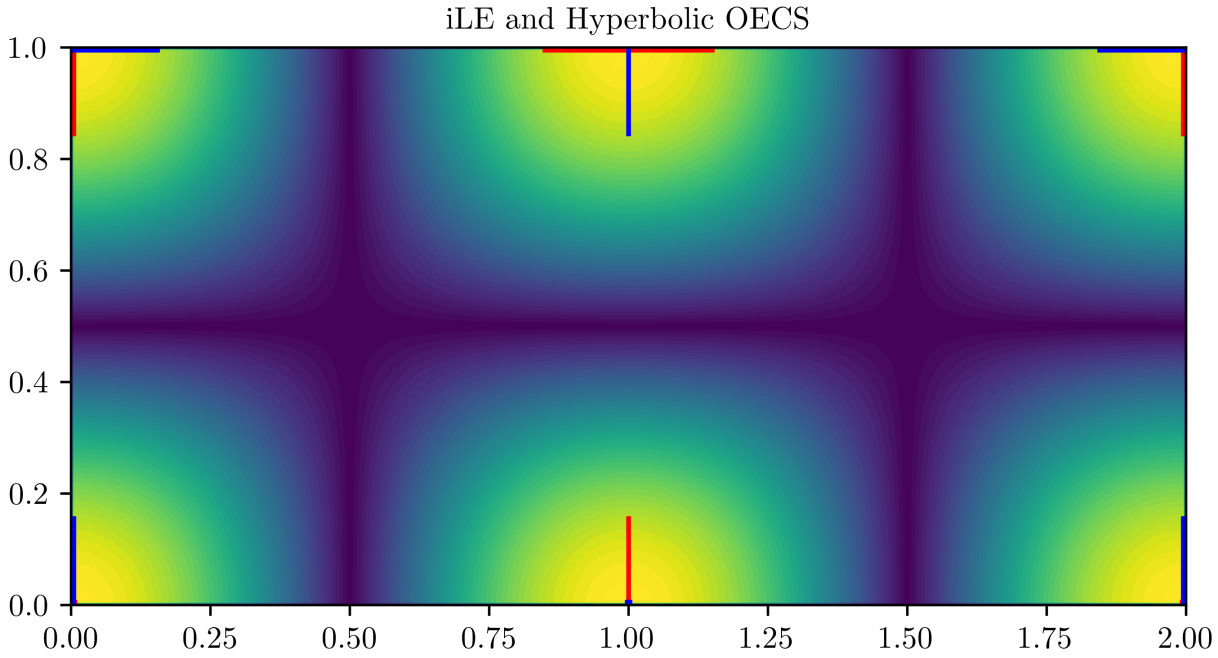

Figure 1.8: Repelling (red) and attracting (blue) OECS overlaid on the iLE field for double gyre at $t = 0$

1.4 Elliptic coherent structures

Elliptic coherent structures are co-dimension 1 surfaces that bound sets which remain coherent under the influence of the flow. These curves are important features of the flow and allow one to identify sets which contain material that resists mixing with the rest of the flows contents. These curves serve as boundaries of persistent vortex structures. There have been two main definitions of these structures (in the purely advective setting). The first defines these curves as solutions to a variational problem in which exceptional curves that extremize

the averaged tangential strain [61] (in the finite-time case, these are material curves) or the averaged tangential stretch rate [141] (in the instantaneous case, these curves are in general not material). The second defines these curves as closed convex level sets of an objective measure of vorticity deviation [64] (from the spatial average of vorticity). We briefly cover each below.

1.4.1 Coherent Lagrangian vortices

Haller and Beron-Vera [61] sought exceptional closed material curves that showed no appreciable variability in their average tangential strain in an infinitesimal belt around the curve (i.e., for a small perturbation of size ϵ to a curve γ, the variability of the average tangential strain to the perturbed curve should be at most $\mathcal{O}(\epsilon^2)$). To define these structures, given our setup (1.1), Let $\gamma \in U$ be a closed material curve. Let $\mathbf{r}(s)$ be a parametrization of the curve where $s \in [0, \sigma]$ and $\mathbf{r}'(s)$ represents the tangent vector to γ at s. Then let

$$q(s) = \frac{\sqrt{\left\langle \mathbf{r}'(s), \mathbf{C}_{t_0}^{t_0+T} \mathbf{r}'(s) \right\rangle}}{\sqrt{\langle \mathbf{r}'(s), \mathbf{r}'(s) \rangle}} \tag{1.31}$$

denote the pointwise tangential material strain from t_0 to $t_0 + T$. Then, the averaged tangential material strain is given by

$$Q(\gamma) = \frac{1}{\sigma} \int_0^\sigma q(s) ds \tag{1.32}$$

Now the proposed definition of coherence (at most $\mathcal{O}(\epsilon^2)$ variability in average tangential strain of an ϵ-close material curve) can be formulated as finding curves γ such that the first variation of Q is zero,

$$\delta Q(\gamma) = 0 \tag{1.33}$$

Solutions to this problem are solutions to a complicated set of differential equations. By invoking results on quotient functionals [23], Haller and Beron-Vera show that any curve γ satisfying Eq. (1.33) will also satisfy,

$$\delta\mathcal{E}_\lambda = 0, \quad \mathcal{E}_\lambda(\gamma) = \int_0^\sigma \langle \mathbf{r}'(s), \mathbf{E}_\lambda(\mathbf{r}(s))\mathbf{r}'(s)\rangle \, ds \tag{1.34}$$

where

$$\mathbf{E}_\lambda(\mathbf{x}_0) = \frac{1}{2}\left(\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0) - \lambda^2\mathbf{I}\right) \tag{1.35}$$

is a generalization of the Green-Lagrange strain tensor ($\lambda = 1$).

After proceeding with the variational calculus, they show that extremizing curves of this functional that coincide with extremizing curves of $Q(\gamma)$ are limit cycles of the following differential equations,

$$\mathbf{r}'(s) = \boldsymbol{\eta}_\lambda^{\pm}(\mathbf{r}(s)), \quad \boldsymbol{\eta}_\lambda^{\pm} = \sqrt{\frac{\lambda^2 - \lambda_2}{\lambda_1 - \lambda_2}}\boldsymbol{\xi}_1 \pm \sqrt{\frac{\lambda_1 - \lambda^2}{\lambda_1 - \lambda_2}}\boldsymbol{\xi}_2 \tag{1.36}$$

defined on the domain

$$U_\lambda = \{\mathbf{x}_0 \in U : \ \lambda_1 \neq \lambda_2, \ \lambda_2 < \lambda^2 < \lambda_1\} \tag{1.37}$$

While extracting these curves is challenging in practice, a number of approaches have been developed which simplify the processes in two dimensions [80, 81, 142]. Using the approach detailed in [142], we show extracted elliptic LCS overlaid on the FTLE field in Figure 1.9.

In Figure 1.10 we show elliptic LCS (blue curves) and the sets they bound (green) advected through the flow. We also show the advection of sets not enclosed by elliptic LCS (orange). The green sets remains coherent while the orange sets do not.

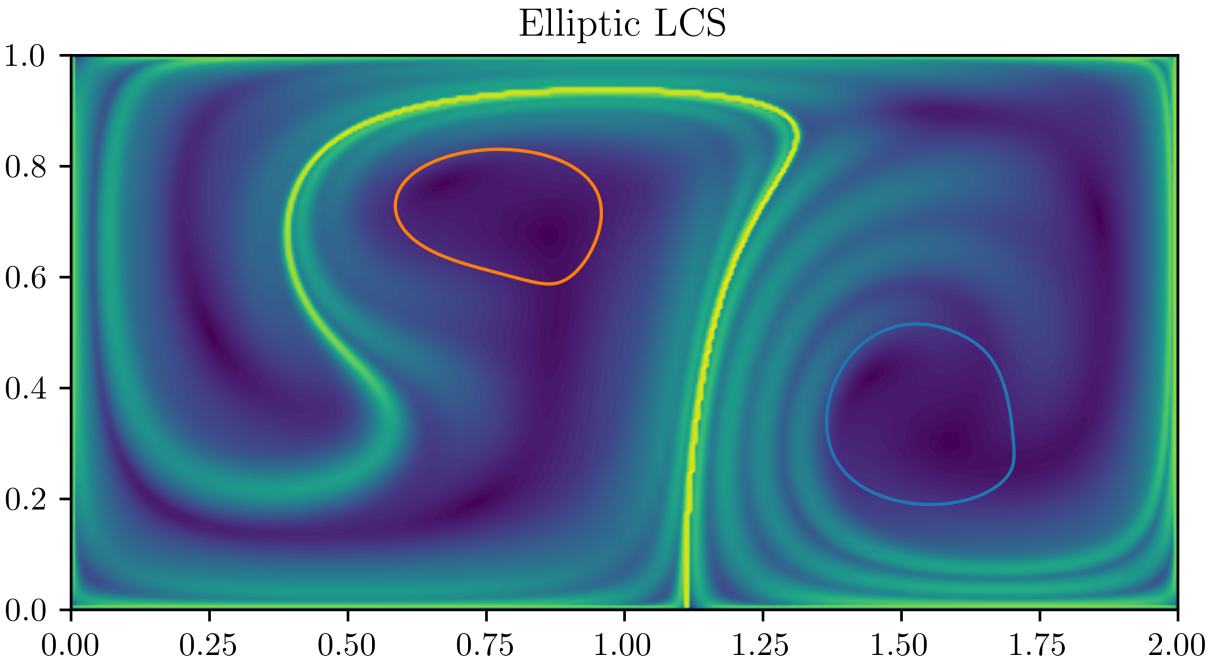

Figure 1.9: Elliptic LCS overlaid on FTLE field for integration time $T = 10$.

1.4.2 Lagrangian-averaged vorticity deviation

In a later paper Haller and coauthors [64] set out to obtain an objective definition of coherent vortices derived from vorticity. The essence of their argument is that coherent vortices should consist of concentric layers of material such that each of these layers undergoes uniform material rotation relative to the spatial mean of vorticity. The most significant of these vortices should highlight regions which remain exceptionally coherent. To this end, they define the *Lagrangian-averaged vorticity deviation (LAVD)*, an objective measure of the integrated vorticity deviation from the spatial mean over some time window of interest.

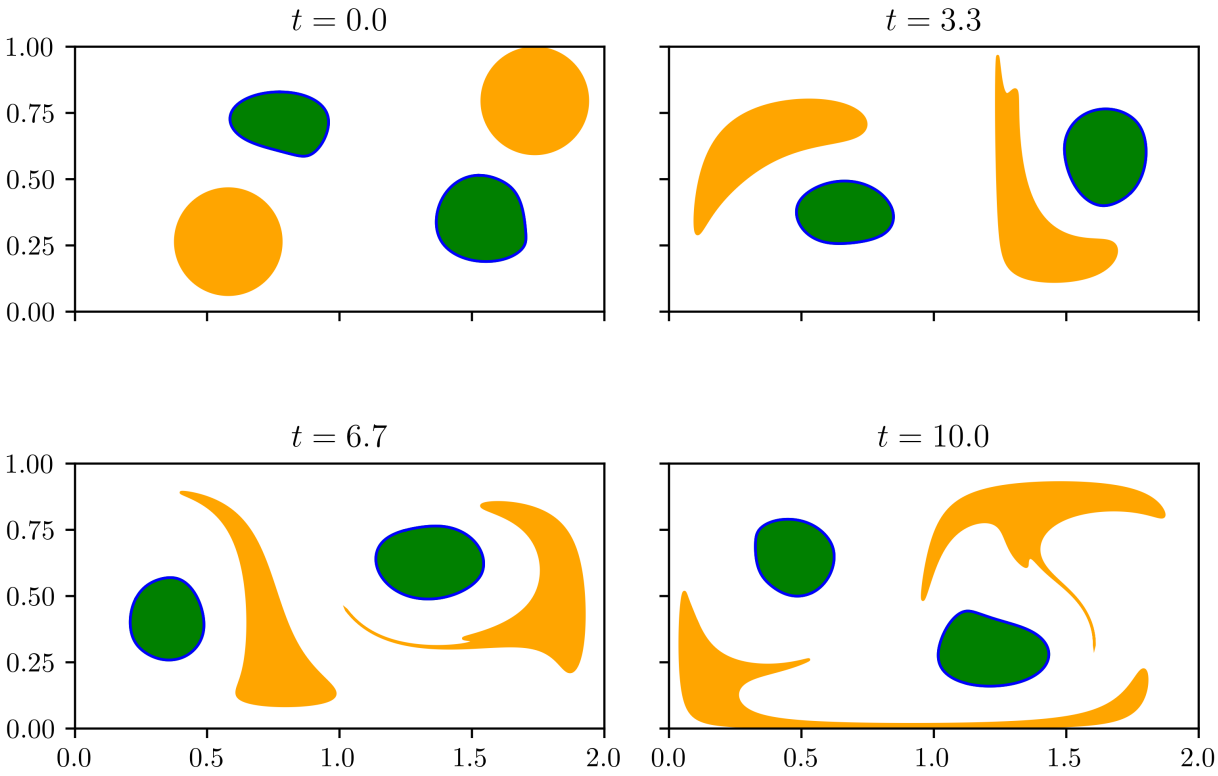

Figure 1.10: Elliptic LCS (blue) and sets they bound (green) along with other sets (orange) advected under flow for $T = 10$. See video.

Given our set up, the vorticity at any point $\mathbf{x} \in U$ is given by $\boldsymbol{\omega}(\mathbf{x}, t) = \nabla \times \mathbf{v}(\mathbf{x}, t)$. Then, the instantaneous spatial mean of vorticity is defined as

$$\overline{\boldsymbol{\omega}}(t) = \frac{\int_U \boldsymbol{\omega}(\mathbf{x}, t) d\mathbf{x}}{\int_U d\mathbf{x}} \tag{1.38}$$

Then the LAVD is defined as,

$$\text{LAVD}^t_{t_0}(\mathbf{x}_0) := \frac{1}{|t - t_0|} \int_{t_0}^{t} |\boldsymbol{\omega}(\mathbf{x}(s; \mathbf{x}_0), s) - \overline{\boldsymbol{\omega}}(s)| ds \tag{1.39}$$

which provides an objective measure of vorticity deviation from the instantaneous spatial mean along a trajectory. Level sets of this field surrounding local maxima of LAVD provide

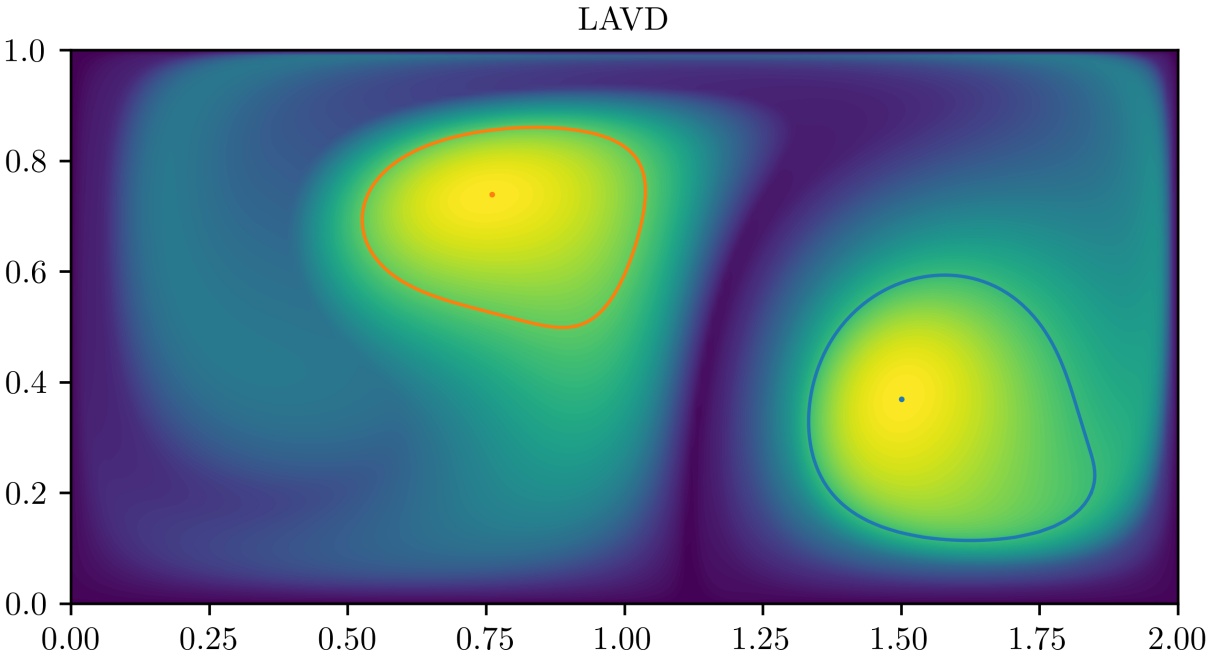

Figure 1.11: LAVD-based vortex centers and LAVD-based elliptic LCS overlaid on LAVD field for integration time $T = 10$.

the sought after concentric closed curves. With this in mind, to extract rotationally coherent Lagrangian vortices, local maxima of the LAVD field are identified and the outermost closed convex level curve around each of these local maxima are extracted. The maxima are referred to as *LAVD-based vortex centers* and the outermost curves are referred to as *LAVD-based elliptic LCS* (or sometimes *rotational LCS*). In Figure 1.11 we show the LAVD-based vortex centers and elliptic LCS overlaid on the LAVD field. While curves extracted via this method and those extracted using the black hole vortices method will in general not perfectly coincide, often they will identify the same structures. In Figure 1.12 we show LAVD-based elliptic LCS (blue curves) and the sets they bound (green) advected through the flow. We also show the advection of sets not enclosed by elliptic LCS (orange). Much like the elliptic LCS from the previous section, the green sets remains coherent while the orange sets do not.

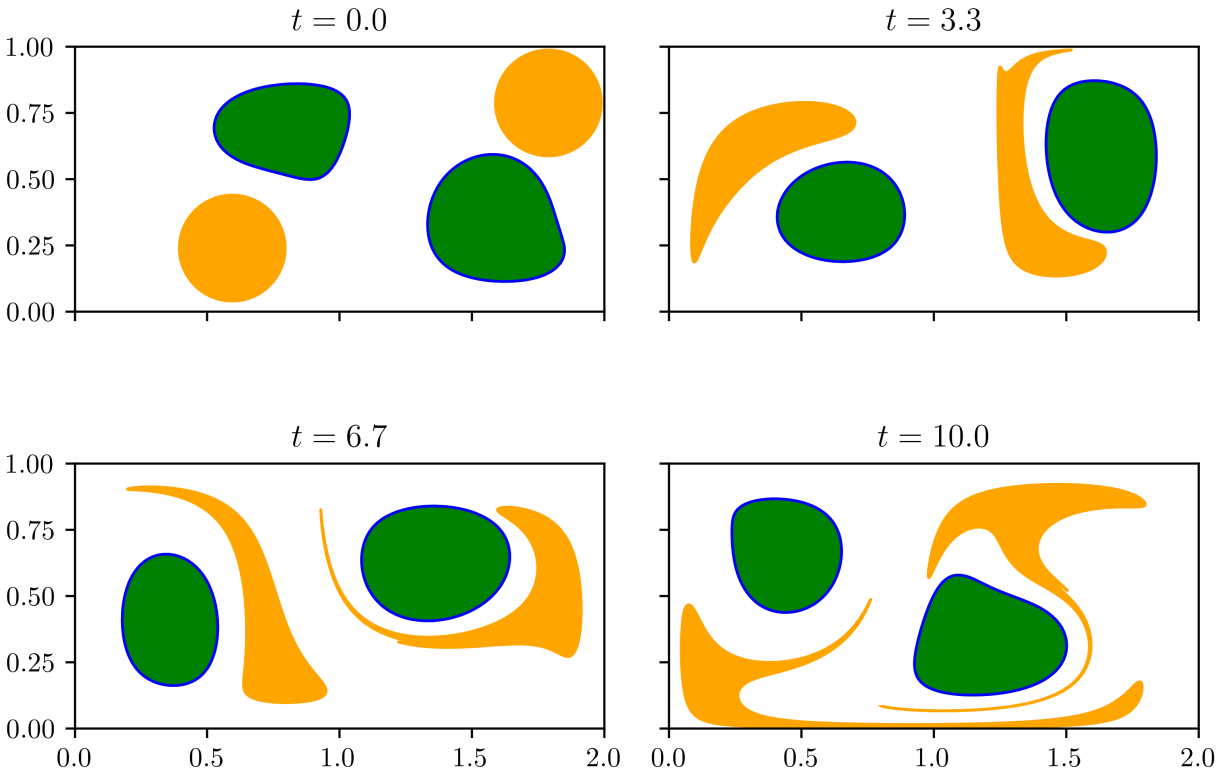

Figure 1.12: LAVD-based elliptic LCS (blue) and sets they bound (green) along with other sets (orange) advected under flow for $T = 10$. See video.

1.4.3 Variational Elliptic OECS

Like the hyperbolic case, Serra and Haller [141] propose Eulerian counterparts to elliptic LCS. These are co-dimension 1 surfaces that, over short times, resist mixing and bound coherent sets. Let a material curve γ be parameterized by $\mathbf{r}(s)$ with $s \in [0, \sigma]$ s.t. $\gamma = \mathbf{r}(s)$ and $\mathbf{r}'(s)$ represents the tangent to γ at $\mathbf{r}(s)$. Then, as previously mentioned, Serra and Haller define the material stretching rate

$$\dot{q}(\mathbf{r}(s), \mathbf{r}'(s), t) = \frac{\langle \mathbf{r}'(s), \mathbf{S}(\mathbf{r}(s), t)\mathbf{r}'(s) \rangle}{\langle \mathbf{r}'(s), \mathbf{r}'(s) \rangle} \tag{1.40}$$

Then, for a given γ the *averaged material stretch-rate* is given by

$$\dot{Q}_t(\gamma) = \frac{1}{\sigma}\int_\gamma \dot{q}(\mathbf{r}(s), \mathbf{r}'(s), t)ds \tag{1.41}$$

They make the same arguments made in [61] (reviewed in Sec. (1.4.1)) about at most $\mathcal{O}(\epsilon^2)$ in a $\mathcal{O}(\epsilon)$ belt. This implies the sought after curves are those on which the first variation of $\dot{Q}_t$ vanishes (i.e., $\delta\dot{Q}_t = 0$). Noting similarities with the variational problem from the Lagrangian case in Sec. (1.4.1), they conclude that elliptic OECS can be extracted as limit cycles of the following differential equations,

$$\mathbf{r}'(s) = \boldsymbol{\chi}_\mu^\pm(\mathbf{r}(s)), \quad \boldsymbol{\chi}_\mu^\pm = \sqrt{\frac{\mu - s_2}{s_1 - s_2}}\mathbf{e}_1 \pm \sqrt{\frac{s_1 - \mu}{s_1 - s_2}}\mathbf{e}_2 \tag{1.42}$$

defined on the domain

$$U_\mu = \{\mathbf{x}_0 \in U : \ s_1 \neq s_2, \ s_2 < \mu < s_1\} \tag{1.43}$$

The numerical developments which simplify the extraction of elliptic LCS mentioned previously apply to the Eulerian structures as well.

1.4.4 Instantaneous vorticity deviation

In the LAVD paper mentioned earlier [64], the authors also define an instantaneous Eulerian counterpart to the LAVD which they call the *instantaneous vorticity deviation (IVD)*. This is simply defined as,

$$\text{IVD}(\mathbf{x}, t) := |\boldsymbol{\omega}(\mathbf{x}, t) - \overline{\boldsymbol{\omega}}(t)| \tag{1.44}$$

providing an objective instantaneous measure of vorticity deviation. They also show that the IVD is the instantaneous limit of the LAVD, much like the link between iLE and FTLE.

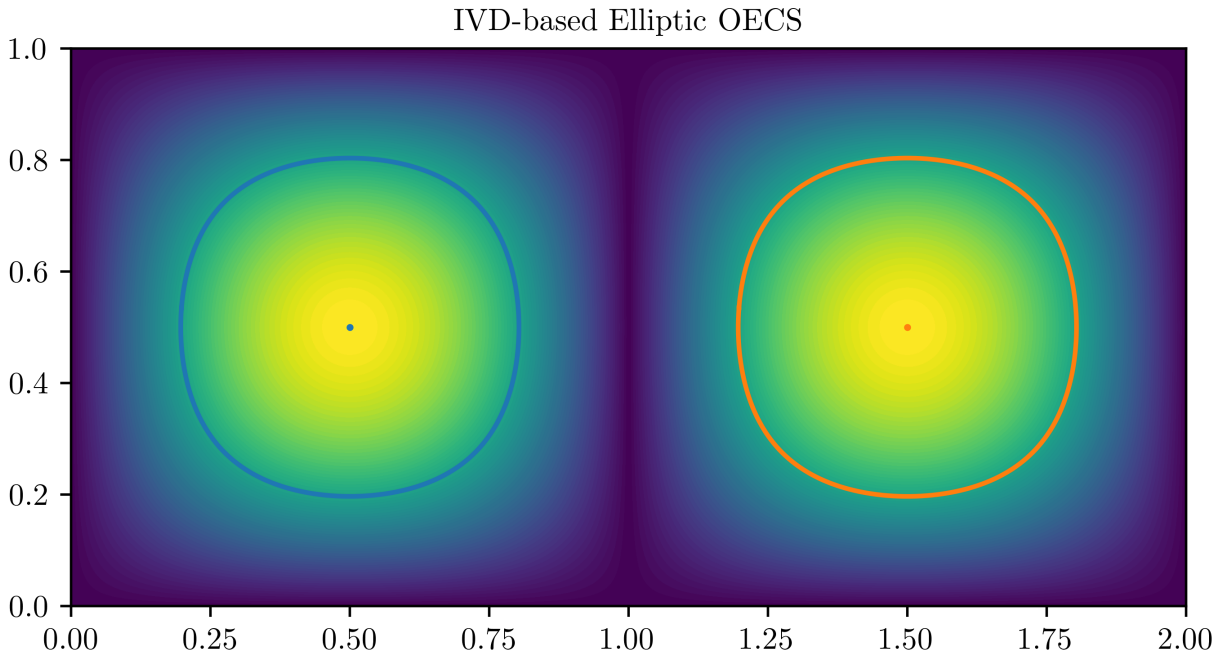

Figure 1.13: IVD-based elliptic OECS for the double gyre at $t_0 = 0.0$.

1.5 Research overview

In the following chapters we cover the main contributions which this dissertation highlights. In Chapter 2, we introduce `NumbaCS`, an efficient Python package that implements many of the methods covered in this chapter. This work is currently under review with the *Journal of Open Source Software*. Due to stringent length requirements from this journal, we present an extension of the submitted manuscript. Chapter 3 is in manuscript form and is the published article *Atmospheric transport structures shaping the 'Godzilla' dust storm*. This chapter retains much of the information from that manuscript. Chapter 4 is a manuscript still in preparation titled *Diffusive flux-rate barriers for general diffusion structure*. Chapter

5 is also a manuscript still in preperation titled *Accurate and Efficient extraction of LCS from their variational theory using Automatic Differentiation*. Finally, Chapter 6 provides a review of the contributions and future directions.

Readers primarily interested in numerical tools and implementations will find relevant material in Chapters 2 and 5. Those looking for a large-scale atmospheric transport application can focus on Chapter 3. Chapter 4 will appeal to readers interested in theoretical developments and extensions.

Chapter 2

NumbaCS: A fast Python package for coherent structure analysis

Attribution

This chapter, a collaborative work with Shane Ross, is an extension of a manuscript currently under review with the *Journal of Open Source Software*.

Author Contributions

Jarvis and Ross conceived this project. Jarvis designed the numerical framework, implemented existing algorithms, designed and implemented original algorithms, optimized code, designed examples, wrote documentation, handled package development and maintenance, and wrote the original manuscript. Jarvis and Ross contributed to revisions.

2.1 Summary

`NumbaCS` (Numba Coherent Structures) is a Python package that implements a variety of coherent structure methods in an efficient and user-friendly manner. "Coherent structure

method" refers to any method which can be used to infer or extract Lagrangian and objective Eulerian coherent structures. The theory behind these methods has been developed over the last few decades with the aim of extending many of the important invariant objects from time-independent dynamical systems theory to the more general setting where a system may have arbitrary time dependence and may only be known or defined for some finite time. These time-dependent systems are ubiquitous in the context of geophysical and engineering flows where the evolution of the velocity field depends on time and velocity data representing these flows is not available for all time. By extending the ideas from the time-independent setting to the more general time-dependent setting, important transient objects (coherent structures) can be identified which govern how material is transported within a flow. Understanding material transport in flows is of great importance for applications ranging from monitoring the transport of a contaminant in the ocean or atmosphere to informing search and rescue strategies for persons lost at sea.

User

2.2 Statement of Need

As theory and implementations of coherent structures have been developed [41, 59, 61, 62, 64, 105, 112, 134, 141, 147] and the utility of these tools has been demonstrated over the last two decades [33, 38, 54, 95, 111, 117, 124, 129, 144], there has been a steadily growing interest in using these methods for real-world applications. Early on, software implementations were largely contained to in-house packages developed by applied mathematicians and engineers advancing the theory. Over the years, there have been a number of software packages developed in an attempt to provide implementations of some of these methods for practitioners outside of the field. Some provide a friendly interface for users (`Dynlab` [110];

`LCS MATLAB Kit` [34]), others aim to provide efficient implementations of specific methods (sometimes in specific circumstances) (`Lagrangian` [16]; `Newman` [37]; `Aquila-LCS` [89]), and a few implement a variety of methods (`TBarrier` [8]; `LCS Tool` [116], `BarrierTool`, [83]). `NumbaCS` intends to unite these aims by providing efficient and user-friendly implementations of a variety of coherent structure methods. By doing this, the hope is to provide a powerful tool for experienced practitioners and a low barrier of entry for newcomers. In addition, as new methods/implementations arise, the framework laid out in `NumbaCS` provides a straightforward environment for contributions and maintenance. Also of note is another package called `CoherentStructures.jl` [78], which is fast, user-friendly, and implements a variety of methods. This package has some overlap with `NumbaCS` but they both implement methods which the other does not. `CoherentStructures.jl` is a powerful tool that should be considered by users who perhaps prefer Julia to Python or are interested in computing some of the methods not implemented in `NumbaCS`. For a more detailed breakdown of how all of the mentioned packages compare with `NumbaCS`, see the documentation.

2.3 Functionality and Implementation

This section provides an overview of the functionality in `NumbaCS` and covers details about the numerical implementation. In the documentation, a User Guide is provided which details the workflow in `NumabCS`, a Theory and Implementation section provides an overview of the theory behind the implemented methods and some details on the numerical implementation, and a number of examples demonstrating the functionality are covered in the Example Gallery. Much of what is covered here will overlap with these sections and we refer the reader to the Introduction (Ch. 1) of this dissertation for a more detailed treatment of the theory.

`NumbaCS` implements the follow features for both analytical and numerical flows:

- Standard flow map computation
- Flow map composition method [17]
- Finite time Lyapunov exponent (FTLE) [147]
- Instantaneous Lyapunov exponent (iLE) [112]
- FTLE ridge extraction [134, 148]
- Lagrangian averaged vorticity deviation (LAVD) [64]
- Instantaneous vorticity deviation (IVD) [64]
- Variational hyperbolic LCS [41, 59]
- Variational hyperbolic OECS [141]
- LAVD-based elliptic LCS [64]
- IVD-based elliptic OECS [64]

For flows defined by numerical velocity data:

- Simple creation of JIT-compiled linear and cubic interpolants

We now briefly cover the implementation of each noteworthy feature and provide some examples. We assume we have a velocity field $\mathbf{v}(\mathbf{x}, t)$ (either given by an analytical expression or by numerical velocity data) as in Eq. (1.1) and we are computing the desired quantity over a n_x by n_y discretized grid $\mathcal{G} = X \times Y \subset U$ where $\times$ refers the the Cartesian product and $X = (x_0, ..., x_{n_x-1})$ and $Y = (y_0, ..., y_{n_y-1})$ refer to the x and y grid-points respectively. We assume constant grid spacing in x given by dx and constant in y given by dy. We can then represent each $\mathbf{x}_0 \in \mathcal{G}$ by the point $(X[i], Y[j])$ for some $i, j \in \{0, ..., n_x - 1\} \times \{0, ..., n_y - 1\}$.

The core of many coherent structure methods relies on approximation of gradients (usually by finite differencing). Assume we have a vector-valued function $\mathbf{f}(\mathbf{x}_0, t)$. Then, given $\mathbf{f}(\mathbf{x}_0, t)$ has x-component $f_0[i, j]$ and y-component $f_1[i, j]$ at $(\mathbf{x}_0, t)$ for some fixed t, the gradient is computed by simple finite differencing,

$$\nabla \mathbf{f}(\mathbf{x}_0, t) \approx \begin{pmatrix} \dfrac{f_0[i+1, j] - f_0[i-1, j]}{2dx} & \dfrac{f_0[i, j+1] - f_0[i, j-1]}{2dy} \\ \dfrac{f_1[i+1, j] - f_1[i-1, j]}{2dx} & \dfrac{f_1[i, j+1] - f_1[i, j-1]}{2dy} \end{pmatrix} \tag{2.1}$$

and $\mathbf{f}(\mathbf{x}_0, t)$ is replaced with the appropriate vector-valued function for the method being applied. Unless otherwise stated, all methods are computed in parallel with the parallelization being applied over the grid and JIT-compiled versions of these functions are used to speed up computations.

2.3.1 Notes on integration

For any Lagrangian method we first compute the flow map for all $\mathbf{x}_0 \in \mathcal{G}$. This is done by looping over all X and Y and applying an ODE solver for each initial condition. For many of these methods, this particle integration step (i.e., solving Eq. (1.1)) is the most costly step. To provide efficient implementations of these methods, the default ODE solver called is a high order explicit Runge-Kutta method (DOP853) with adaptive stepping and optional dense output. This method is invoked as compiled FORTRAN code which bypasses the Python interpreter [162], leading to quite drastic speed ups for this portion of Lagrangian methods. In addition, by default the solver will be invoked in parallel with the parallelization applied over the grid.

2.3.2 FTLE

FTLE is computed via finite differencing of neighboring trajectories and then a subsequent eigenvalue problem. First, we obtain flow maps for all $\mathbf{x}_0 \in \mathcal{G}$ Then, we compute the gradients using neighboring flow maps to obtain the linearized flow map for each $\mathbf{x}_0$. From here, we construct the Cauchy-Green tensor, compute its largest eigenvalue and apply the FTLE formula (1.14).

2.3.3 iLE

The iLE field is computed in essentially the same way as the FTLE field except the finite differencing is done with the velocity field itself (yielding the gradient of the velocity field), rather than the final integrated positions. Using the velocity gradient, we construct the Eulerian rate-of-strain tensor and find its maximum eigenvalue. This is returned for the iLE.

2.3.4 FTLE ridges

Much like FTLE, we begin by obtaining flow maps for all initial conditions, computing gradients using neighboring trajectories, constructing the Cauchy-Green tensor, and then computing eigenvalues and eigenvectors. Following that, we apply a ridge extraction method. There are three main methods to extract FTLE ridges from an FTLE field. All methods identify the same ridge points but differ in how they filter these ridge points and connect ridge points to form ridges (if the method does this). We will cover the main algorithm. Ridge points are extracted as described in Sec. (1.2.2). To review, a point is a ridge point

if,

$$1.)\ \langle \nabla\sigma, \boldsymbol{\xi}_1 \rangle = 0 \tag{2.2}$$

$$2.)\ \langle \boldsymbol{\xi}_1, \mathbf{H}_\sigma \boldsymbol{\xi}_1 \rangle < 0 \tag{2.3}$$

Due to condition (2.2) being strict, applying these conditions directly in a numerical context will lead to a substantial underestimation of ridge points since a grid point will rarely be exactly on a ridge. If instead this conditions is relaxed and we look for points such that the magnitude of the inner product is below some threshold, many points around ridges (and some on ridges) will be identified. Neither of these results are desirable. A contour generating algorithm is sometimes suggested (e.g., marching squares) to find zeros of condition (2.2) but this comes with its own challenges due to the orientation discontinuities in the eigenvector field. To get around these issues, `NumbaCS` follows an approach detailed by Steger [148] to extract ridges from images. This approach obtains ridge points with subpixel accuracy by Taylor expanding the height field (in our case the FTLE field) at a grid point to find if condition (2.2) is satisfied near that grid point and if so, where exactly it is satisfied. This approach yields accurate FTLE ridges that are extremely cheap to compute. `NumbaCS` also follows the approach in Steger to order and link points which belong to the same ridge. In addition, another optional step will connect ridges if they are "close enough" and "in-line". Therefore, the `NumbaCS` implementation can be seen as a hybrid of the Schindler et al. [134] C-ridges and the algorithm put forth by Steger [148] for computing and linking ridge points with an additional (optional) ridge linking step.

2.3.5 Variational Hyperbolic LCS

Computing hyperbolic LCS from their variational theory is far more complex and involved than simply computing FTLE. There are two main reasons for this. The first is has to do with errors arising from approximating derivatives with finite differencing as is done in Eq. (2.1). It turns out that the eigenvalues obtained from $\mathbf{C}_{t_0}^{t_0+T}$ (used to compute FTLE) are quite robust to the spacing used in the finite differencing (the underlying grid spacing) but the eigenvector directions are very sensitive to this spacing. Since candidate hyperbolic LCS are computed as solution curves in one the eigenvector fields of $\mathbf{C}_{t_0}^{t_0+T}$, obtaining accurate eigenvectors is of the utmost importance if one wants to obtain accurate LCS. To solve this issue, Farazmand and Haller [41] propose integrating an auxiliary grid of 4 particles around each $\mathbf{x}_0 \in \mathcal{G}$ where the auxiliary grid spacing is much smaller than the initial grid spacing (i.e., $h_{aux} << \min(dx, dy)$). Lekien and Ross [92] show that using a grid spacing this fine will result in inaccurate computation of the FTLE though, due to the fact that, in general, all of these auxiliary grid points will be on one side of a hyperbolic LCS. To remedy this, it is suggested that both the main grid and auxiliary grid around each point are used and the eigenvalues are computed from the main grid while the eigenvectors are computed from the auxiliary grid.

The second main hurdle has to do with the direction field from which these solution curves are obtained. Since these direction fields are obtained from eigenvectors, the eigenvector direction is not uniquely defined (i.e., if $\boldsymbol{\xi}$ is an eigenvector than so is $-\boldsymbol{\xi}$). For this reason, there will be orientation discontinuities in the direction field and, in general, these orientation discontinuities cannot be globally rectified. The proposed solution is to rectify orientation discontinuities *in-place* as the ODE is being solved. This is done at each stage of the solver and is done by identifying the nearest grid points, checking that the eigenvectors at these surrounding grid points do not have an orientation discontinuity (if they do, rectify this

discontinuity by flipping the eigenvector(s) that cause the discontinuity), then use these surrounding grid points to interpolate the eigenvector at the current point using a linear interpolant, and confirm it is "in-line" with the eigenvector from the previous step (if it is not, flip it). In addition, points where $\lambda_1 = \lambda_2$ are referred to as degenerate points of the tensor field induced by $\mathbf{C}_{t_0}^{t_0+T}$ (since this results in eigenvector directions being ill-defined). So in practice, the differential equation we actually solve is

$$\mathbf{r}'(s) = \mathbf{sign}(\langle \boldsymbol{\xi}_2(\mathbf{r}(s)), \mathbf{r}'(s - \Delta s) \rangle) \alpha(\mathbf{r}(s)) \boldsymbol{\xi}_2(\mathbf{r}(s)) \tag{2.4}$$

where

$$\alpha(\mathbf{x}) = \left(\frac{\lambda_1(\mathbf{x}) - \lambda_2(\mathbf{x})}{\lambda_1(\mathbf{x}) + \lambda_2(\mathbf{x})} \right)^2 \tag{2.5}$$

and Δs is the step size used in the ODE solver. Due to the new differential equation we are solving, which requires an eigenvector orientation check (and possible correction) for each stage of the method, we can no longer employ fancy pre-built adaptive methods since they do not allow us access to the intermediate stages. For this reason, a custom explicit fixed step-size 4th order Runge-Kutta method is used that performs these checks (and possible corrections) at each step. `NumbaCS` uses a JIT-compiled version of this custom solver in an attempt to speed up computations. The `NumbaCS` implementation closely follows the algorithm put forth by Farazmand and Haller [41] with additional options for filtering candidate LCS and a different approach to initial conditions.

2.3.6 Variational Hyperbolic OECS

Computing hyperbolic OECS from their variational theory is similar to the procedure for hyperbolic LCS but due to the instantaneous nature of OECS, some of the challenges associated with finite-time structures do not exist in the instantaneous case. The most important

is that particle integration does not need to be performed so we obviously avoid the need for auxiliary grid particle integration required for the finite-time case. Instead, we proceed by computing the Eulerian rate-of-strain tensor by performing spatial finite differences of the velocity field. If using numerical data, it is suggested that the grid be sufficiently resolved and if it is not, create an interpolant of the velocity field so finite differencing can be computed with sufficiently small spacing, resulting in accurate eigenvectors. Once we have computed the Eulerian rate-of-strain tensor, we compute eigenvalues and eigenvectors. For repelling OECS, we solve the differential equation,

$$\mathbf{r}'(s) = \mathbf{sign}(\langle \mathbf{e}_2(\mathbf{r}(s)), \mathbf{r}'(s - \Delta s) \rangle)(\mathbf{r}(s)) \tag{2.6}$$

with initial conditions given by local maxima of the s_1 field. Integration continues until a boundary is reached or the magnitude of s_1 is no longer monotonically decreasing. Attracting OECS are computed in the same fashion but with $\mathbf{e}_1$ and s_2. `NumbaCS` closely follows the algorithm put forth by Serra and Haller [141].

2.3.7 LAVD-based Elliptic LCS

To compute LAVD-based elliptic LCS we first need to compute LAVD. This begins with computing particle trajectories $\mathbf{F}_{t_0}^{t_0+T}(\mathcal{G})$. Then compute vorticity[1] for the time window of interest and for the domain given by the range of $\mathbf{F}_{t_0}^{t_0+T}(\mathcal{G})$. Then we compute the instantaneous spatial mean of vorticity for the time window $[t_0, t_0 + T]$. Finally, for each trajectory, apply the LAVD formula (1.39). Once the LAVD field is obtained, local maxima are identified. Then, a contour generating algorithm is employed which will extract level

[1] If vorticity data is readily available, as is often the case for geophysical flows, it is recommended to use this data as it will be more accurate and more efficient to use

sets of the LAVD field. For each local maxima, find all level sets that form closed curves around that maxima. Finally, extract the outermost convex closed curve enclosing that local maxima. In practice, we allow for a small convexity deficiency which can be prescribed by the user. The convexity deficiency is computed as the relative difference in area between the candidate curve and its convex hull. The `NumbaCS` implementation closely follows the algorithm put forth by Haller et al. [64].

2.3.8 IVD-based Elliptic OECS

For IVD-based elliptic OECS, one simply needs to compute the vorticity at $t = t_0$, compute the spatial mean of vorticity at this time, and compute the pointwise magnitude of the difference between these two quantities. Once the IVD field is obtained, the extraction of curves is performed in the same way as described in the previous section 2.3.7. The `NumbaCS` implementation closely follows the algorithm put forth by Haller et al. [64].

2.3.9 Flow map Composition

As mentioned previously, the most expensive portion of any finite-time coherent structure method is typically the particle integration step required to solve Eq. (1.1) and obtain flow maps (Eq. (1.2)). If the coherent structure quantity to be computed is only desired at a single instant, there is not much one can do to cheapen this step aside from using more efficient solvers, relaxing error tolerances, or using a coarser grid. Often, the quantity of interest is desired in a time series though, rather than at just a single frame. When this is the case, Brunton and Rowley [17] noted that redundant computation is being performed in the particle integration step that could be circumvented by interpolating the flow map (see Figure 1 in the cited article for a clear picture of the redundancy). This is done by exploiting

the semigroup property of the flow maps,

$$\mathbf{F}_{t_0}^{t_0+T} = \mathbf{F}_{t_{N-1}}^{t_N} \circ \cdots \circ \mathbf{F}_{t_1}^{t_2} \circ \mathbf{F}_{t_0}^{t_1} \tag{2.7}$$

where $t_N = t_0 + T$ and the associative binary operation is the composition operation. Therefore, the flow map over some time window $[t_0, t_0 + T]$ can be obtained by composing a collection of intermediate flow maps of shorter time windows. This is not very useful when only a single flow map is desired but if a time series of flow maps is needed, this can cut down the computational cost quite substantially. Note that if a time series of FTLE was desired for times $t_0, t_1, ..., t_n$, derived from flow maps over the following time windows $[t_0, t_0 + T], [t_1, t_1 + T], ..., [t_n, t_n + T]$, these could be computed as,

$$\begin{aligned}
\mathbf{F}_{t_0}^{t_0+T} &= \mathbf{F}_{t_{N-1}}^{t_N} \circ \cdots \circ \mathbf{F}_{t_1}^{t_2} \circ \mathbf{F}_{t_0}^{t_1} \\
\mathbf{F}_{t_1}^{t_1+T} &= \mathbf{F}_{t_N}^{t_{N+1}} \circ \cdots \circ \mathbf{F}_{t_2}^{t_3} \circ \mathbf{F}_{t_1}^{t_2} \\
&\vdots \\
\mathbf{F}_{t_n}^{t_n+T} &= \mathbf{F}_{t_{N+n-1}}^{t_{N+n}} \circ \cdots \circ \mathbf{F}_{t_{n+1}}^{t_{n+2}} \circ \mathbf{F}_{t_n}^{t_{n+1}}.
\end{aligned} \tag{2.8}$$

Clearly, for each full flow map $\mathbf{F}_{t_k}^{t_k+T}$, all but the first intermediate flow map ($\mathbf{F}_{t_k}^{t_{k+1}}$) used to create this full flow map can be recycled and used to create the successive flow map $\mathbf{F}_{t_{k+1}}^{t_{k+1}+T}$. Given that the flow map is being computed on a grid $\mathcal{G}$, it is necessary to interpolate each intermediate flow map (after the first) since, in general, $\mathbf{F}_{t_k}^{t_{k+1}}(\mathcal{G}) \not\to \mathcal{G}$ and this will be the input to the next intermediate flow map. Therefore, Brunton and Rowley define the interpolation operator $\mathcal{I}$ which acts on a discrete map $\mathbf{F}_{\mathcal{G}_k} := \mathbf{F}_{t_k}^{t_{k+1}}(\mathcal{G})$ and returns the interpolated map, i.e.,

$$\begin{aligned} \mathcal{I} : \mathbf{F}_{\mathcal{G}_k} \mapsto \mathcal{I}\mathbf{F}_k \quad \text{where} \\ \mathcal{I}\mathbf{F}_k : U \to U \end{aligned} \tag{2.9}$$

and $\mathcal{I}\mathbf{F}_k$ is the interpolated flow map $\mathbf{F}_{t_k}^{t_{k+1}}$. Then, for any $t_k \in \{t_0, t_1, ..., t_n\}$

$$\begin{aligned} \widetilde{\mathbf{F}}_{t_k}^{t_k+T}(\mathcal{G}) &= \mathcal{I}\mathbf{F}_{N+k-1} \circ \cdots \circ \mathcal{I}\mathbf{F}_{k+1} \circ \mathbf{F}_{\mathcal{G}_k} \\ &\approx \mathbf{F}_{t_k}^{t_k+T}(\mathcal{G}) \end{aligned} \tag{2.10}$$

They call this the unidirectional method and also define a bidirectional method. They show that the bidirectional method is not as accurate or as fast so we omit any further discussion here. In addition, they suggest an alternative method for storage of smaller composed intermediate flow maps called the multi-tier (as opposed to single-tier) method. `NumbaCS` implements the single-tier unidirectional method which is what was described above. For examples using this method, refer to the Time series section from the Examples Gallery.

2.4 Examples

Here we show the output of a few examples, provide the runtime of each, and breakdown the runtime based on the parts of each method. "Flowmap" refers to the particle integration step, "C-eig" and "S-eig" refer to the eigenvalue/vector step for Lagrangian and Eulerian methods respectively (this time will be roughly equal to the FTLE and iLE times), and the last is the extraction time for a given method. For examples that require particle integration, the default solver (DOP853) was used with the default error tolerances (relative tolerance =

1e-6, absolute tolerance = 1e-8). All runs were performed on an Intel(R) Core™ i7-3770K CPU @ 3.50GHz (which has 4 cores and 8 total threads). Warm-up time[2] is not included in the timings.

Double Gyre (Analytical Flow)

We briefly cover this flow in Sec. 1.1.1.

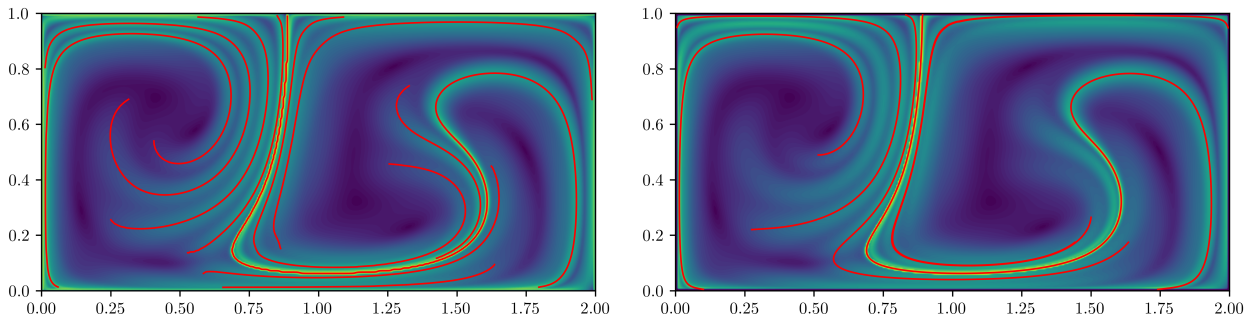

Figure 2.1: Left: DG FTLE ridges at $t_0 = 0$, integration time $T = -10$. Total runtime per iterate: $\sim$ 0.424s (flowmap: $\sim$ 0.390s; C-eig: $\sim$ 0.025s; FTLE ridge extraction: $\sim$ 0.009s). Right: DG hyperbolic LCS at $t_0 = 0$, integration time $T = -10$. Total runtime per iterate: $\sim$ 5.219s (flowmap (aux grid): $\sim$ 1.83s; C eig (aux grid): $\sim$ 0.039s; hyperbolic LCS extraction: $\sim$ 3.350s). Both are computed over a 401x201 grid.

[2]Since many functions in `NumbaCS` are JIT compiled, these functions are optimized and compiled into machine code on the first function call. This initial delay is often referred to as "warm-up time". After the first call, subsequent function calls are much faster.

Bickley jet (Analytical Flow)

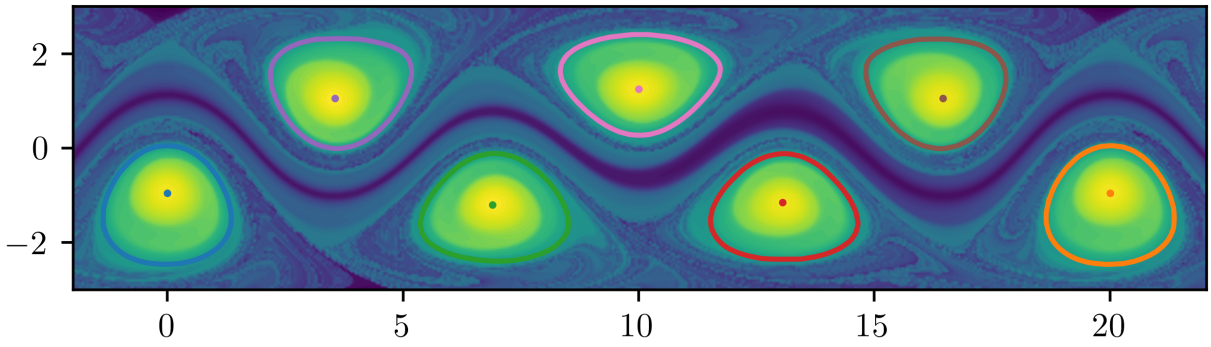

Figure 2.2: Bickley jet elliptic LCS at $t_0 = 0$, integration time $T = 40$ days. Total runtime per iterate: $\sim$ 9.200s (flowmap: $\sim$ 5.050s; LAVD: $\sim$ 4.140s; elliptic LCS extraction: $\sim$ 0.010s). Computed over 482x121 grid.

MERRA-2 (Numerical Flow)

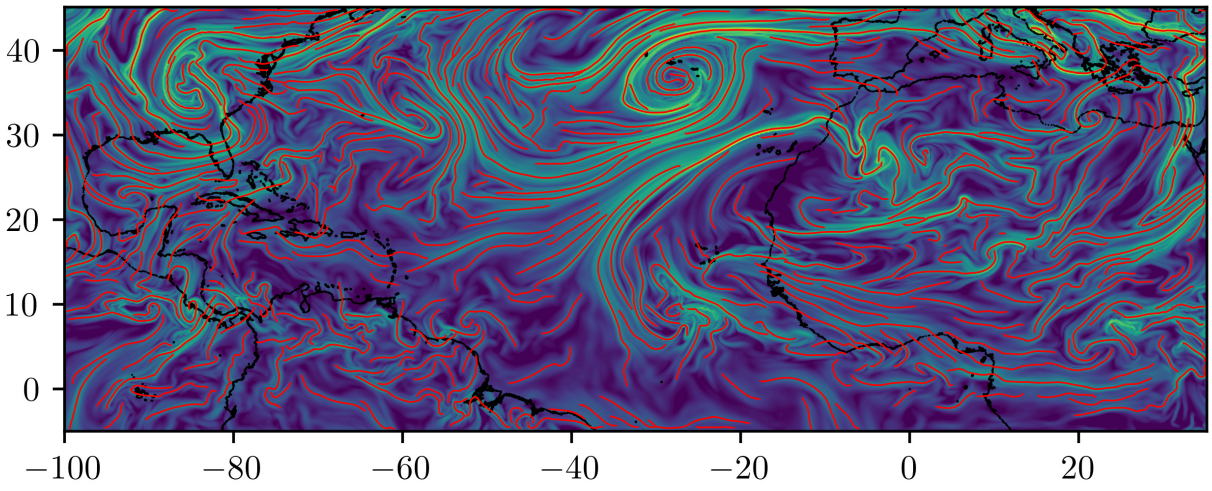

Figure 2.3: MERRA-2 FTLE ridges at $t_0 = 06/16/2020 - 00:00$, integration time $T = -72$hrs. Total runtime per iterate: $\sim$ 7.835s (flowmap: $\sim$ 7.480s; C-eig: $\sim$ 0.085s; FTLE ridge extraction: $\sim$ 0.270s). Computed over a 902x335 grid.

QGE (Numerical Flow)

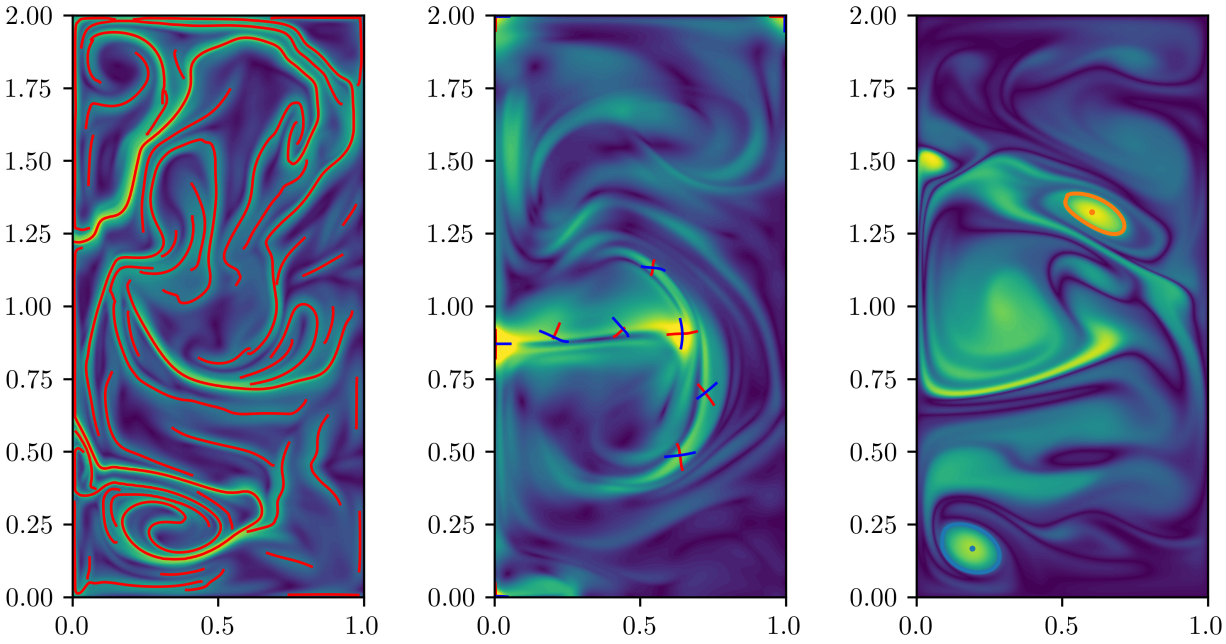

Figure 2.4: Left: QGE FTLE ridges at $t_0 = 0$, integration time $T = 0.1$. Total runtime per iterate: $\sim$ 2.461s (flowmap: $\sim$ 2.400s; C-eig: $\sim$ 0.0.038s; FTLE ridge extraction: $\sim$ 0.023s). Middle: QGE hyperbolic OECS at $t_0 = 0.15$. Total runtime per iterate: $\sim$ 2.238s (S eig: $\sim$ 0.038s; hyperbolic OECS extraction: $\sim$ 2.200s). Right: QGE elliptic OECS at $t_0 = 0.5$. Total runtime per iterate: $\sim$ 0.0452s (IVD: $\sim$ 0.0002s; elliptic OECS extraction: $\sim$ 0.045s). All are computed over a 257x513 grid.

2.5 Key dependencies

All of these implementations are relatively straightforward to use and quite efficient. This is due to three key dependencies `NumbaCS` utilizes to speed up computations. The first is `Numba` [90], a JIT compiler for Python which can drastically speed up numerical operations and provides a simple framework for parallelizing tasks. Next, `numbalsoda` [162] is a Python wrapper to ODE solvers in both C++ (LSODA) and FORTRAN (DOP853) that bypasses the Python interpreter and can be used within `Numba` functions (standard Python ODE solvers

cannot be executed within `Numba` functions). This package is crucial to the efficiency of `NumbaCS` as particle integration is often the most costly part of finite-time coherent structure methods. Finally, the `interpolation` package [161] provides optimized interpolation in Python and is utilized in `NumbaCS` to create JIT-compiled interpolant functions, producing efficient implementations of methods even for flows defined by numerical data. By taking advantage of these packages behind the scenes, `NumbaCS` is able to maintain the simplicity and readability of a dynamically-typed language while achieving runtimes closer to that of a statically-typed language.

2.6 Road map

As of the writing of this dissertation, `NumbaCS` is currently on version 0.1.1. Development continues with the main aims being extending functionality, implementing new features, and further streamlining and optimizing current features, especially those related to large-scale geophysical applications. We will list possible additions below and break them into three categories: features/improvements already under development, features/improvements not currently under development but with plans to develop sometime in the future, and features/improvements which have no current plans for development.

Under development

- Elliptic LCS and OECS via null geodesics (Serra and Haller [142])
- DBS and diffusive transport barriers (Haller et al. [68])
- Direct use of netCDF files for geophysical flows
- Support for masked data

- Rewrite hyperbolic LCS for efficiency

Planned

- FTLE on non-Eucledian manifolds (Lekien and Ross [92])
- Shape coherent sets (Ma and Bollt [101])
- Elliptic LCS and OECS via index theory (Karrasch and Schilling [80])
- Extend diagnostic methods to 3D
- Option to cache JIT-compiled functions

No current plans

- Finite time coherent sets via FEM method (Froyland and Junge [47])
- Parabolic LCS (Farazmand et al. [42])
- Active transport barriers (Haller et al. [69])
- Extend extraction methods to 3D

2.7 Datasets

Two datasets are provided with `NumbaCS` to test the functionality for flows defined by numerical velocity data. One is a numerical simulation of the quasi-geostrophic equations (QGE). We thank the authors of Mou et al. [107] for providing us with this dataset, which was used extensively during during development, and allowing a piece of the dataset to be included in the package. The full dataset was over the time span $[10, 81]$ with $dt = 0.01$. We provide

the velocity fields over the much shorter time span $[10, 11]$ with the same dt. For details on the parameters used in the simulation, refer to the cited paper. The other dataset is a MERRA-2 vertically averaged reanalysis dataset [1, 50], which was used as part of a paper [75] coauthored by the authors of this paper. Wind velocity fields were vertically averaged over pressure surfaces ranging from 500 hPa to 800 hPa. The corresponding latitude, longitude, and date arrays are also provided. All data can be downloaded from the data folder on the GitHub page.

2.8 Usage in ongoing research

As of the writing of this paper, `NumbaCS` has not been public for long but has been utilized in one publication [75] where it was the computational tool for all coherent structure methods. In addition, it is currently being used in an ongoing project focused on airborne invasive species traveling from Australia to New Zealand titled "Protecting Aotearoa from wind-dispersed pests". This is a five year (October 2023 - October 2028) Scion-led and Ministry of Business, Innovation and Employment (MBIE)-supported program

2.9 Acknowledgments

This work was partially supported by the National Science Foundation (NSF) under grant number 1821145 and the National Aeronautics and Space Administration (NASA) under grant number 80NSSC20K1532 issued through the Interdisciplinary Research in Earth Science (IDS) and Biological Diversity & Ecological Conservation programs.

Chapter 3

Atmospheric transport structures shaping the "Godzilla" dust plume

Attribution

This chapter, a collaborative work with Ali Hossein Mardi, Hosein Foroutan, and Shane Ross, was originally published in *Atmospheric Environment* [75].

Author Contributions

All authors contributed to the conception of this project. Jarvis and Hossein Mardi obtained the data and wrote code. Jarvis designed the study, implemented numerical simulations, performed formal analysis, and produced figures. Jarvis wrote most of the original manuscript with help from Hossein Mardi. Jarvis coordinated revisions and all authors contributed to revisions.

Abstract

Saharan dust events, having great ecological and environmental impacts, are the largest producers of the world's dust by far. Identifying the mechanisms by which the dust is transported across the Atlantic is crucial for obtaining a complete understanding of these important events. Of these events, the so-called "Godzilla" dust intrusion of June 2020 was the largest and most impactful in the last two decades and underwent a particularly interesting transport pattern. By uncovering dominant, organizing structures derived from the wind velocity fields, known as Lagrangian coherent structures, we demonstrate the ability to describe and qualitatively predict certain aspects related to the evolution of the dust plume as it traverses the atmosphere over the Atlantic. In addition, we identify regions of high hyperbolicity, leading to drastic changes in the shape of the plume and its eventual splitting. While these tools have been quite readily adopted by the oceanographic community, they have still yet to fully take hold in the atmospheric sciences and we aim to highlight some of the advantages over traditional atmospheric transport methods.

3.1 Introduction

On the third week of June 2020, Aerosol Optical Depth (AOD) measurements of numerous AErosol RObotic NETwork (AERONET) sun photometers located in the Caribbean recorded levels above the long-term background AOD. The observed increase in AOD was caused by a plume of dust, which departed from the western coast of Africa on June 17. The unusual extent and concentration of this African dust plume gave it the "Godzilla" nickname and pushed the 24h average $PM_{2.5}$ concentration far above the U.S. Environmental Protection Agency's (EPA) 35 $\mu g/m^3$ air quality standard in more than 70 air quality

measurement stations located in the southeast U.S. on June 26-27 [166]. The Godzilla dust plume traveled across the Atlantic Ocean, sliding above the marine boundary layer with a maximum plume altitude of 6-8 km and an approximate layer thickness of 3.4 km, showing the characteristics of the Saharan Air Layer (SAL) [21, 82]. Back trajectory analysis of the receptor regions impacted by this plume denote an average travel speed of 15 m/s, which enabled the plume to travel across the Atlantic Ocean in approximately 8-9 days [39]. Outside the African continent, the plume maximum density was reached on June 18 with average AOD value reaching as high as 1 and a maximum AOD value above 1.5 (unitless), over an area between $5°\text{N} - 30°\text{N}$ and $50°\text{W} - 10°\text{W}$ based on NASA Moderate Resolution Imaging Spectroradiometer (MODIS) data. For the mentioned area, such high amounts of AOD were unprecedented during the month of June since 2002 [4].

It would be challenging to locate the exact origin of the emissions leading to the Godzilla dust plume but a deep look into the MODIS satellite imagery of days prior to the incident hints at central and western regions of North Africa as the origin of the plume [127]. Even though the Godzilla dust intrusion into the Caribbean is categorized as a historic event, dust emissions leading to the formation of the plume did not show such extreme characteristics. Yu et al. [166] suggests the modulation of synoptic meteorological conditions as the reason behind the accumulation of dust near the coast of Africa. In their study, the location of a North Atlantic Sub-tropical High (NASH) synoptic system is probed via the NASA Modern-Era Retrospective analysis for Research and Applications, Version 2 (MERRA-2) reanalysis geopotential height at 600 hPa on the days prior to the release of the Godzilla dust plume toward the Caribbean. They note that June 2020 NASH geopotential heights are on average 80 m higher than the 1980-2020 climatology. This study concludes that the specific location of the NASH system, north of the plume and co-occurrence of this geopotential anomaly combined with strong dust emissions from Africa have led to the

historic Godzilla dust intrusion. This example of the synoptic condition role in historic dust intrusions further motivates the analysis of air flow over the Atlantic Ocean to determine the inter-connectedness between the flow regime and dust intrusions.

Organizing Structures in the Atmosphere. Identifying the organizing structures within a geophysical flow is an important part of understanding the transport of a given material under the action of that flow. For example, in the ocean, to know how warm water from the Gulf of Mexico is transported to higher latitudes in the Atlantic ocean, it is important to identify the Gulf Stream and understand its mixing process [95]. Similarly, in the atmosphere, the spread of wildfire smoke over large distances is determined by atmospheric structures, which may not have a specific nomenclature associated with them [32, 124]. To identify these structures and understand how they evolve, an ensemble air parcel trajectory-based—or so-called Lagrangian—point of view must be taken; this incorporates the time evolution of the flow as the material is transported in it. Streamlines, vector fields, and wind barbs come from velocity data at only an instant in time and often provide little insight for material transport over some finite time window in highly time-dependent flows, like wind velocity fields [54, 65]. In addition, these quantities are frame-dependent and conclusions drawn from them can be skewed based on the frame of reference. As pointed out by Bujack and Middel in a review of flow visualization techniques in the environmental sciences [18], these instantaneous (or so-called Eulerian) techniques are commonplace within the atmospheric science community. But they suggest atmospheric flow visualization could benefit from using feature-based extraction techniques and topological methods, like Lagrangian coherent structures.

In fact, transport feature-based methods (LCS and finite-time coherent sets) have been applied to atmospheric applications. The applications have ranged from transport of mi-

crobes and aerosols [49, 111, 119, 124, 135, 137, 154], to hurricane intensification and entrainment [38, 128, 129, 133], and the dynamics of the Arctic and Antarctic polar vortices [77, 92, 132, 144]. However, most of these listed studies were performed by nonlinear dynamicists rather than atmospheric scientists, demonstrating that the atmospheric community has not adopted these methods as readily as the oceanographic community. With this work, we seek to add to the growing body of literature demonstrating the usefulness of these techniques for atmospheric science in the hopes that practitioners will integrate them into their toolboxes. We refer the reader to Günther et al. [54] for a more detailed comparison of some of the common methods in quantifying atmospheric transport and the benefits of using coherent structure methods.

To study the transport of African dust across the Atlantic Ocean, studies have incorporated a trajectory-based approach to either isolate the source region of dust plumes [3, 27, 39] or study the transport pattern and geographic region impacted by transported dust [52]. To establish a connection between source and receptor regions, either backward trajectories are computed from locations downwind along the prevalent dust transport path or forward trajectories are computed from known source regions during an intense dust emission incident. The robustness of the linkage between the source and receptor regions can be further evaluated using retrievals of dust plumes from satellite remote sensing [13, 164, 165]. In addition, some groups take an Eulerian approach in an attempt to discover the underlying mechanisms of trans-Atlantic dust transport. In this approach, the focus of analysis is on the synoptic conditions from a few days prior to the the emission of dust until the dust plume has reached the receptor region. In studies done on the Godzilla storm [44, 166], comparison of anomalies in geopotential height and wind velocity fields relative to the long-term historic climatology (1991-2020) in conjunction with the analysis of the streamlines reveals the underlying atmospheric circulation patterns responsible for emission and transport of dust

incidents. In most cases, these patterns point to an anomaly in seasonal weather patterns as possible indicators of historic dust transport incidents. While not a study specific to dust, Chakraborty et al. [24] identified major aerosol transport pathways across the globe by extending the concept of atmospheric rivers (ARs) for water vapors to aerosol atmospheric rivers (AARs). By doing so, the authors were able to develop a framework to identify extreme aerosol transport events for different major aerosol species and identified AARs specific to dust at the time of the Godzilla storm. Similar to the Lagrangian approach, remote sensing retrievals serve as a real world evaluation for the Eulerian analysis [126].

Regardless of the approach, identifying possible indicators of anomalous dust incidents can facilitate prediction of them in the future. While these studies are all useful and make major contributions towards the goals they set out to achieve, none tackle the problem of understanding these dust events through the lens of Eulerian and Lagrangian coherent structures. In this paper, we look at the Godzilla dust event and focus mainly on the Lagrangian approach, coupled with some Eulerian diagnostics, with the aim of obtaining a large-scale template for transport while the dust plume was present. By identifying key attracting, repelling, and bounding structures (described below), we obtain a qualitative road map for dust transport during this historic event.

3.2 Methods

For this work, we mainly focus on methods which we will refer to as "coherent structure methods" which will be described more thoroughly below. We present these tools and their applications in a high-level manner to familiarize the reader with the simplest implementation of these methods. There is a good deal of nuance relating to how certain methods are implemented, differences between some methods, and when it is appropriate to use one over

another. We omit most of this discussion from the main text but provide more details for the interested reader in supplemental material.

3.2.1 Finite Time Coherent Structures

Lagrangian techniques for uncovering patterns related to material transport have become powerful tools over the last few decades [36, 45, 46, 59, 63, 65, 100, 103, 117, 136, 146, 147, 154]. These techniques were born out of the desire to generalize asymptotic methods from autonomous dynamical systems theory [151, 160] and transfer them to the finite time, nonautonomous setting, appropriate for realistic geophysical flows. Of these methods, the finite-time Lyapunov exponent (FTLE) and closely related Lagrangian coherent structures (LCS) have emerged as among the most widely used. These methods provide objective (i.e., frame-invariant) diagnostics for extracting the most influential material curves (in a 2D flow) or surfaces (in a 3D flow). The hyperbolic[1] structures derived from the mentioned methods can produce a skeleton for transport in unsteady time-dependent flows, identifying material curves or surfaces responsible for attracting, repelling, and bounding regions of the flow over a time window of interest. To define them, first consider the following initial value problem, viewed as a dynamical system over some general n-dimensional smooth manifold $\mathcal{M}$ (e.g., $\mathbb{R}^3$ or $\mathcal{S}^2$),

$$\begin{aligned} \frac{d}{dt}\mathbf{x} &= \mathbf{v}(\mathbf{x},t), \qquad \mathbf{x}\in U\subset\mathcal{M}, \quad t\in I\subset\mathbb{R} \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{aligned} \tag{3.1}$$

[1] 'Hyperbolic' means that the structures exponentially attract or repel nearby material.

Then, there exists a family of diffeomorphisms $\{\mathbf{F}_{t_0}^t\}$ (known as the flow maps) associated with the dynamical system given by,

$$\begin{aligned} \mathbf{F}_{t_0}^t &: I \times I \times U \to U \\ &: \mathbf{x}(t_0; t_0, \mathbf{x}_0) \mapsto \mathbf{x}(t; t_0, \mathbf{x}_0) \end{aligned} \tag{3.2}$$

in which either $t > t_0$ (mapping forward in time) or $t < t_0$ (mapping backwards in time). The flow maps, as depicted in Figure 3.1, take points $\mathbf{x}_0$ (or regions A_0) in the fluid at an initial time t_0 and map them to their locations at some other time $t = t_0 + T$ (where T can be positive or negative). To extract features from the flow map, we define the right

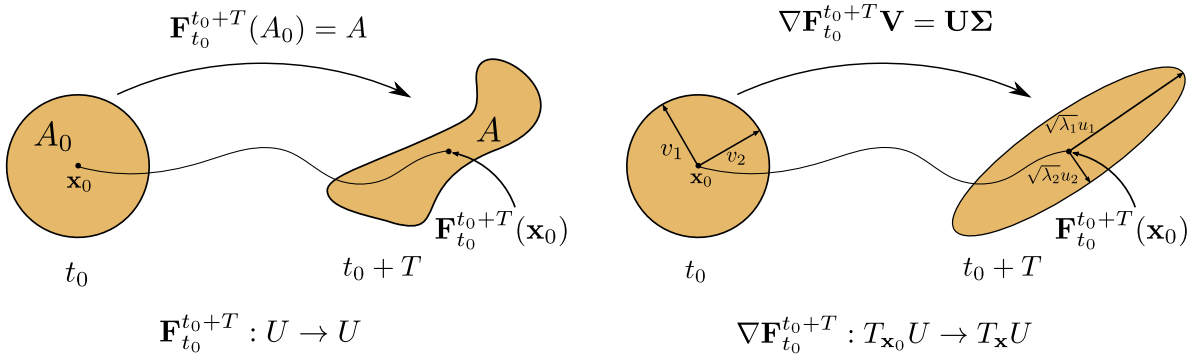

Figure 3.1: Left: Action of the flow map on a point $\mathbf{x}_0$ and enclosing set A_0 over the time window $[t_0, t_0 + T]$. $\mathbf{F}_{t_0}^{t_0+T}$ acts on elements of the domain and maps them to the domain. Its action on a set can be defined in the following manner: $\mathbf{F}_{t_0}^{t_0+T}(A_0) \coloneqq \{\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0) \in U | \mathbf{x}_0 \in A_0\}$. Right: Action of the linear approximation of the flow map acting on an infinitesimal circle defined by vectors v_1, v_2 over the time window $[t_0, t_0 + T]$. The derivative of the flow map, $\nabla\mathbf{F}_{t_0}^{t_0+T}$, acts on elements of the tangent space (i.e., vectors) based at $\mathbf{x}_0$ and maps them to elements of the tangent space downstream at $\mathbf{x} = \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)$. The meaning of the eigenvalues and eigenvectors of $\mathbf{C}_{t_0}^{t_0+T}$ can equivalently be seen through the SVD of $\nabla\mathbf{F}_{t_0}^{t_0+T} = \mathbf{U}\,\mathbf{V}^*$. The singular values ($\mathrm{diag}(\mathbf{U})$) are equal to the square root of the eigenvalues ($\sqrt{\lambda_i}$) and the right singular vectors (v_i) are equal to the eigenvectors ($\boldsymbol{\xi}_i$).

Cauchy-Green deformation tensor in terms of the linearized flow map (again see Figure 3.1),

$$\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0) = (\nabla\mathbf{F}_{t_0}^{t_0+T})^\top \nabla\mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0). \tag{3.3}$$

where ∇ represents the derivative with respect to the initial position $\mathbf{x}_0$ and $T = t - t_0$ is the flow map duration or so-called integration time (which could be positive or negative). The matrix $\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0)$ is symmetric and positive-definite with real eigenvalues λ_i and corresponding orthonormal eigenvectors $\boldsymbol{\xi}_i$ with $i \in \{1, 2, ..., n\}$ such that,

$$\lambda_1 \geq ... \geq \lambda_n > 0 \text{ and,} \tag{3.4}$$

$$\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j \rangle = \delta_{ij}. \tag{3.5}$$

There is a variational theory for hyperbolic LCS [59] wherein the structures are found by identifying solution curves of the maximum and minimum eigenvector fields of $\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0)$ which satisfy certain conditions relating to their influence on material deformation relative to nearby solution curves. The variational LCS method provides a way to find the precise surfaces which dominate fluid-parcel deformation over the interval T of interest, but such surfaces are often quite costly to compute, can be challenging to implement, and have recently been shown to lack robustness to uncertainty in velocity data relative to other coherent structure methods [5]. Alternatively, the FTLE (a finite-time analogue of the classic Lyapunov exponent) field is a scalar field that is related to the average exponential rate of stretch of initially (infinitesimally) nearby particles over the time window of interest. Note that the maximum eigenvector of $\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x})$ gives the direction in which an infinitesimal perturbation will undergo the maximum growth (given by $\sqrt{\lambda_1}$ or equivalently $e^{\sigma|T|}$) over the finite time window $[t_0, t_0 + T]$ where σ is the FTLE, defined as,

$$\sigma_{t_0}^{t_0+T}(\mathbf{x}_0) = \frac{1}{2\,|T|} \log(\lambda_1) \tag{3.6}$$

While the FTLE field still requires a significant amount of particle integration, the numerical implementation can be optimized via parallel computing, significantly reducing the compu-

tation time. We use an in-house python package, that makes use of parallelization and just-in-time compilation, that allows for quick computation even with very large data sets, and in spherical coordinates appropriate for global geophysical flows.

Often, ridges of the FTLE field can be used as a proxy for hyperbolic LCS [147] and the easiest way to identify ridges is to simply use a thresholding method which only looks at FTLE values above a certain threshold, yielding regions of high stretching in forward or backward time. Other methods exist to extract curves (in 2D) that usually coincide with hyperbolic LCS. Care should be taken when implementing these methods and one should be aware of potential pitfalls as they can sometimes result in false positives in regions of high shear if additional criteria is not satisfied along the ridge [59]. Later on, we link to a video of LCS overlaid on FTLE which confirms that all of the important structures we focus on are indeed attracting LCS.

For this work, we start by focusing on backward time FTLE fields, yielding attracting coherent structures (see Figure 3.2). The main reason we do this is to elucidate the predictive capabilities of these methods and demonstrate their usefulness for real-time decision making. These structures are computed over the time window $[t_0 - T, t_0]$ where t_0 is the current time. Attracting structures are computed using only current and past velocity data and therefore can be implemented in (essentially) real-time to inform decision making in time-sensitive applications, since the velocity data is readily available. By contrast, repelling structures (think the inverse behavior of Figure 3.2) are computed over the time window $[t_0, t_0 + T]$ and therefore require future velocity data for their computation. This necessitates forecasting of the velocity fields to make use of them for real-time decision making, introducing further error and uncertainty.

While attracting structures are computed from the backward time system, due to the continuity of the flow and their dominance relative to nearby material lines, they tend to persist

for at least *some* portion of the future time window, preserving their usefulness in the predictive setting, as will be demonstrated. In addition, in recent work [124], attracting structures have been shown to act as "air bridges" (see Figure 3.2, left) in large-scale ($\sim$ 1,000 km) atmospheric flows, behaving as pathways for material to be transported along. We were interested to see if these air bridges persisted on much larger scales (a few thousand kilometers). What we found suggests that the "air bridge" concept may still be relevant, but its role in transport depends on which side of the bridge the material of interest began on (see Figure 3.2).

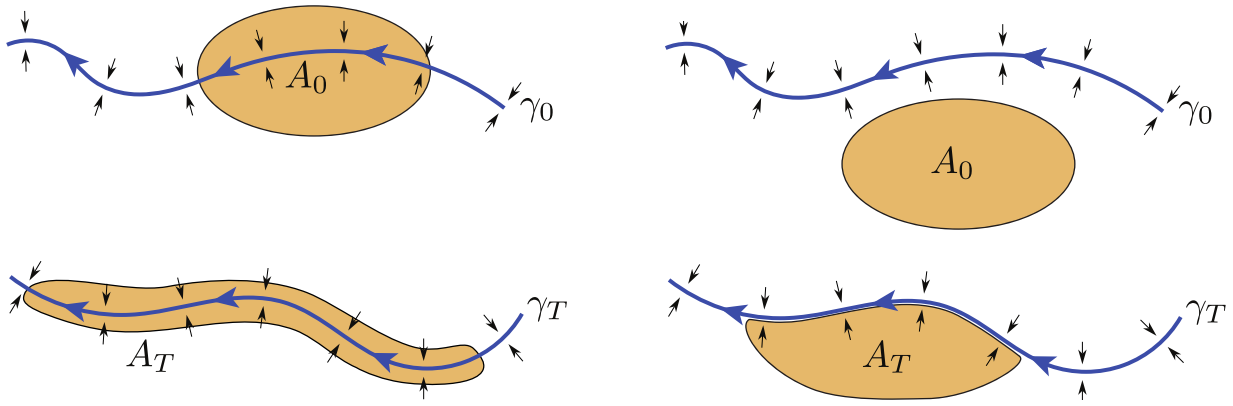

Figure 3.2: Behavior of initial blob A_0 straddling an attracting LCS (left) which acts as an "air bridge" and an initial A_0 below the same attracting LCS (right) after some finite time window of interest.

3.2.2 Instantaneous Coherent Structures

In addition to computing the structures mentioned above which require a certain duration T, one can compute *instantaneous* structures, requiring the velocity only at a single instant t_0. These instantaneous structures are the finite-time coherent structures as the finite-time T goes infinitesimally to zero and like their finite time counterparts, they are also objective, providing an attractive alternative to other common Eulerian quantities. Using only the current data frame, they are simple to compute and often can illuminate important short

time structures/regions within the flow. These structures arise out of the eigenvalues and eigenvectors of the Eulerian rate-of-strain tensor, given by:

$$\mathbf{S}(\mathbf{x}_0, t) = \frac{1}{2}\left(\nabla\mathbf{v}(\mathbf{x}_0, t) + (\nabla\mathbf{v}(\mathbf{x}_0, t))^\top\right) \tag{3.7}$$

where $\mathbf{S}(\mathbf{x}_0, t)$ is a symmetric matrix with real eigenvalues s_i and corresponding orthonormal eigenvectors $\mathbf{e}_i$ with $i \in \{1, 2, ..., n\}$ such that,

$$s_1 \geq ... \geq s_n \text{ and,} \tag{3.8}$$

$$\langle \mathbf{e}_i, \mathbf{e}_j \rangle = \delta_{ij}. \tag{3.9}$$

Following Nolan et al. [112], in a $n = 2$ dimensional flow, denote s_1 (respectively, s_2) by s_+ (resp., s_-), which are instantaneous Lyapunov exponents (iLEs). The iLE field is the limit of the FTLE field as integration time goes to 0,

$$\lim_{T \to 0^\pm} \sigma_{t_0}^{t_0+T}(\mathbf{x}_0) = \pm s_\pm(\mathbf{x}_0, t_0) \tag{3.10}$$

where the superscript $\pm$ on 0 denotes whether the limit is from above $(+)$ or below $(-)$. In this work, we compute the iLE field to identify a region of the plume which undergoes significant instantaneous hyperbolic deformation.

3.2.3 Vortex Identification

Throughout our analysis, we identify vortex structures in the FTLE field that play an important role in the evolution of the dust plume. Identifying vortices by way of FTLE is not standard and we make an attempt to use more common methods to confirm the structures we are focusing on are indeed vortices. In addition, while we make no claims that

the storms we find are cyclones (or anticyclones), we show that they exhibit some common characteristics of cyclones, further demonstrating the strength of these storms. We look at a few basic diagnostics for identifying storms. Mean sea level pressure (MSLP) maps are looked at during the formation of these vortices to see if low pressure systems can be identified. Low level (850 hPa) vorticity is also probed to find highs in these fields, a common characteristic of strong storms and cyclones. Cyclone identification involves considerably more complexity than just these diagnostic methods, and there are many different schemes to detect these storms [9, 20, 25, 159], but we are not concerned with whether or not the structures we identify are properly defined cyclones. Our main focus is on how other structures in the FTLE field interact with these storms. To this end, we simply aim to confirm that these storms behave like vortices. With this in mind, the final diagnostic we look at is the Lagrangian averaged vorticity deviation (LAVD), an objective quantity used to find rotationally coherent vortices in a flow, known as elliptic LCS [64]. Assuming we have a system (3.1) with corresponding flow map (3.2), the vorticity at any point $\mathbf{x}$ will be given by $\omega(\mathbf{x}, t) = \nabla \times \mathbf{v}(\mathbf{x}, t)$. Then, the instantaneous spatial mean of vorticity is given by:

$$\bar{\omega}(t) = \frac{\int_U \omega(\mathbf{x}, t) dV}{\text{vol}(U)}, \tag{3.11}$$

where vol$(\cdot)$ represents the volume (3d) or area (2d) and dV represents either a volume or area element. From there, the LAVD is defined as

$$\text{LAVD}_{t_0}^{t}(\mathbf{x}_0) \coloneqq \frac{1}{|t - t_0|} \int_{t_0}^{t} |\omega(\mathbf{x}(s, \mathbf{x}_0), s) - \bar{\omega}(s)| ds, \tag{3.12}$$

which can be computed either forward or backward in time but, it has been noted that the different time direction calculations will generally result in different elliptic LCS [66]. To identify the boundaries of these coherent vortices, the outermost closed, convex contour

surrounding a local maxima is extracted to serve as the elliptic LCS. In this work, we are less interested in extracting the curves themselves and use the LAVD field as a diagnostic for identifying coherent vortices.

3.2.4 Data and Software

NASA MERRA-2 data (see Figure 3.3) is used to create the wind velocity fields with a $0.5° \times 0.625°$ horizontal resolution, 42 pressure levels of vertical resolution, and a 3-hourly temporal resolution. MERRA-2 is a continuation of the former NASA MERRA reanalysis dataset with an improved meteorology and atmospheric model, generated by the NASA Global Modelling and Assimilation Office [14, 50]. MERRA-2 wind data [1] is chosen due to nominal coverage and widespread application and validation in the literature [22, 84]. MSLP and low level vorticity are obtained from the ERA5 dataset [71, 72], which is the European Centre for Medium-Range Weather Forecasts (ECMWF) fifth-generation reanalysis dataset for the global climate and weather of the past eight decades. Datasets are available hourly in a gridded format with a spatial resolution of 0.25° x 0.25°° and vertical coverage of 1000 hPa to 1 hPa on 37 vertical pressure levels. For streamfunction and vorticity calculations from MERRA-2 data shown later, the windspharm python package [35] is used which utilizes spherical harmonics to compute these quantities. All other diagnostics (FTLE, iLE, and LAVD) and feature extraction (LCS) are performed by an in-house software package available on GitHub.

To isolate the dust plume, the ultraviolet Aerosol Index (AI) values from the Ozone Mapping and Profiling Suite (OMPS), installed on the Suomi-NPP satellite are used, which is the official AI product provided by NASA [156] (see Figure 3.4). Suomi-NPP has a sun-synchronous orbit, meaning that it passes the equator at the same local mean solar time on each pass with

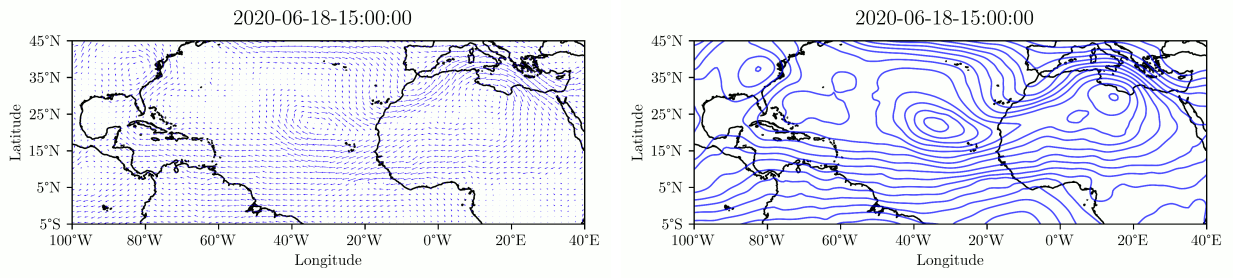

Figure 3.3: From MERRA-2 for 2020-06-18-15:00:00. Left: Velocity field in region of interest. Right: Corresponding streamfunction

multiple swathes covering Earth's surface on each day. Hence, to create daily dust plume images, multiple swathes covering a region between $5°\text{S} - 45°\text{N}$ and $100°\text{W} - 35°\text{E}$ on each day are selected and averaged to a pixel size of $0.5°$. The OMPS AI product is a unitless columnar value representing the atmospheric aerosol concentration from Earth's surface up to the sensor height, with higher AI values representing higher aerosol concentrations. Both dust and smoke aerosols are UV absorbing and therefore contribute to the AI values [163] but considering our latitudes of interest and extent of the dust plume, we believe the impact from other types of aerosols are negligible. As we seek a comparison with 2D vertical column-averaged aerosol index data, we use column-averaged vector fields [111] where the averaged velocity fields come from pressure surfaces ranging from 500 hPa - 800 hPa. These pressure surfaces are where the dust was mostly present, as estimated from a previous study [126].

3.2.5 Caveats with Implementation

The approach mentioned above results in a 2D time-varying column averaged quasi-velocity field. This vector field is no longer a *true* velocity field as it does not describe the velocity of a true fluid parcel anymore. Care should be taken when employing an approach like this. The resulting vector field need not be incompressible anymore and it is perhaps possible,

depending on the velocity fields being averaged, to obtain structures in the FTLE field which are artifacts of the averaging. In our case, we show that the field is indeed incompressible and the main structures we focus on are not artifacts of the averaging by comparing with FTLE obtained from a common pressure surface used in the literature for this event (600 hPa). For more details on why we chose this approach and potential problems this could cause if due diligence is not performed, we refer the reader to section S2 of the Supplementary Material.

The dust concentration we focus on evolves due to the both advective and diffusive processes. FTLE and standard LCS were developed solely in the context of advective transport. More recently, theory has been developed which takes diffusion into account in the weakly diffusive case [68]. Out of this work comes the diffusive barrier strength (or sensitivity) field (DBS), which acts as a diffusive counterpart to FTLE. Although utilizing DBS could potentially offer greater accuracy and relevance, this paper focuses on presenting the simplest and most commonly recognized technique among Lagrangian coherent structure methods: FTLE. It is important to clarify that DBS is not inherently complex; it merely involves additional steps beyond FTLE, along with some increases in memory overhead and computational cost. However, considering our goal to engage practitioners effectively, we suggest that readers might prefer using existing FTLE codebases. There are numerous FTLE implementations available, compared to the relatively limited number for DBS. This availability could be a decisive factor in the choice of methods for practical application. Furthermore, it has been noted in Haller et al. [68] that, unless the diffusion structure tensor is sufficiently anisotropic or the underlying velocity field has significant temporal aperiodicity, the DBS field will not differ significantly from the FTLE field. To be sure we are not missing anything substantial, we compare the FTLE to the DBS over a large domain used in a later section (see video).

In addition to the diffusive considerations, dust particles are inertial (not neutrally buoyant) and their density differs from that of air. For this reason, a more accurate representation

of the structures could be obtained by computing inertial FTLE (iFTLE) or inertial LCS which uses the Maxey-Riley equations in the particle integration step of the FTLE computation to capture the inertial effects of the particles [152, 153]. Along with the additional computational expense, doing this would require estimating the Stokes number and size of the dust particles. However, we show that using the simplified[2] method captures dominant structures related to transport and evolution of the dust plume.

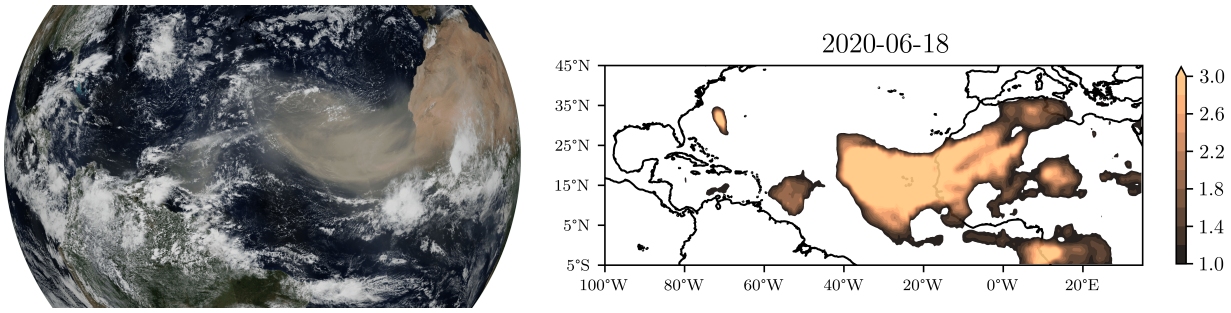

Figure 3.4: Left: Godzilla dust plume, June 18, 2020 (Credit: NASA). Right: Aerosol Index data (unitless) from OMPS (see video). These values are used throughout the paper.

3.3 Results

Using data retrieved from MERRA-2, we begin by computing backward FTLE for an integration time of 96 hours (4 days) from June 5, 2020 to June 30, 2020. The integration time chosen should generally be tied to some characteristic time scale of the flow or the material being transported by the flow. If too short a time is chosen, there is a risk of missing important structures, and if too long of a time, often one ends up with an overly complicated set of LCSs from which it is hard to derive meaning. The plume takes ∼8 days to traverse

[2]We note that we are in the density regime in which particles will more strongly be attracted to passive attracting ridges (dust is an "aerosol" or a heavier particle than the carrier fluid and the Stokes number is not small) as opposed to the case when the inertial particles are lighter than the carrier fluid (known as "bubbles"), in which case passive attractors act more like repellers and the inertial particles aggregate away from these structures. See the mentioned references for more details. In the regime we are in, the simplification becomes more justified.

the Atlantic Ocean. We tested a variety of integration times ranging from 1 day to 8 days. We settled on 4 days since this was roughly half the time it took for the dust to traverse the Atlantic and produced satisfactory FTLE fields. See S3 of the Supplemental Material for more details. Here, we demonstrate a simplified implementation of the coherent structure methods for the problem at hand to highlight the robustness of these methods.

We focus on the affected area where latitude runs from $5°\text{S} - 45°\text{N}$ and longitude runs from $100°\text{W} - 35°\text{E}$ but use the global velocity field in our FTLE calculations (see Figure 3.5). The aerosol index data provided by OMPS representing the actual extent of the dust plume was overlaid on the region of interest. For the coherent structures we present, a simple thresholding method is used for both the fields of interest (to obtain ridges corresponding to dominant attracting/repelling structures) and the aerosol index data (in an attempt to focus on the dust plume itself) with the aim of demonstrating the effectiveness of a comparatively simple-to-use and computationally inexpensive approach for real-time decision making. If one is interested in obtaining actual curves instead of visible ridges of a scalar field, which is useful when quantities normal and tangent to the ridge are desired, a ridge detection algorithm can be deployed [105, 140, 147, 148] to obtain these structures from the FTLE field. Alternatively, LCS be found with the variational method. We show both approaches in Figure 3.5. Note that the relatively high threshold value we have adopted may result in the omission of numerous LCS, but these are of lower strength (as the FTLE, and thus ridge height inherently measures). These omitted structures are generally of lesser importance for the analysis of large-scale transport phenomena that we aim to investigate. This thresholding approach was deliberately chosen to prioritize the identification and analysis of the most influential and large-scale structures, aligning with our research objectives. If one was interested in obtaining more of the LCS, ridge detection could be performed without any thresholding.

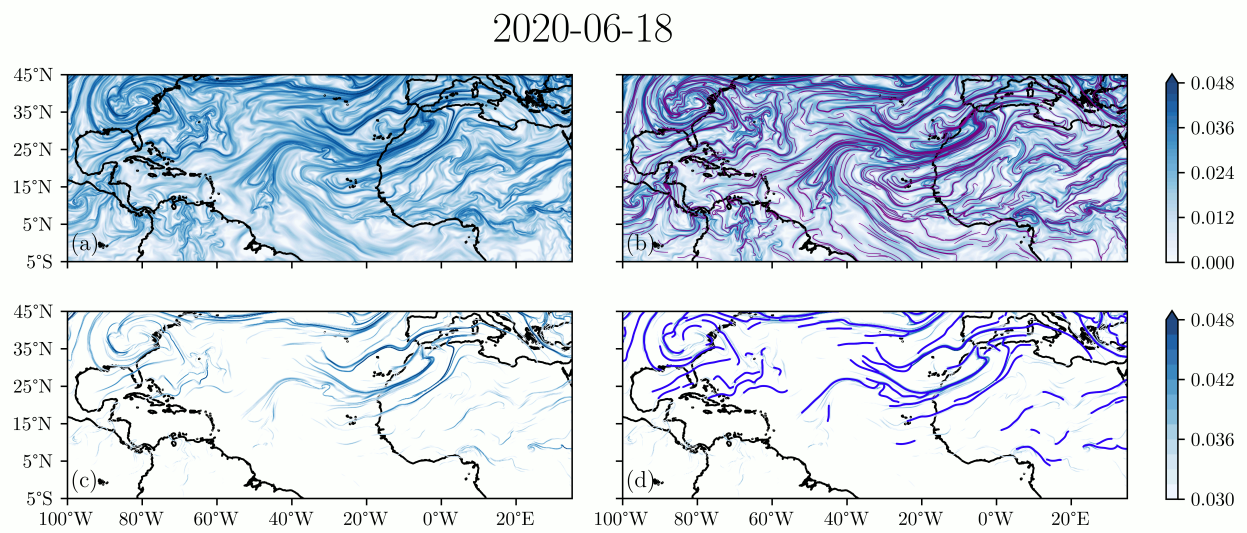

Figure 3.5: (a): Backward FTLE field (hr^{-1}) at same time as composite image via satellite (see FTLE video). (b): Attracting LCS (purple) found via variational method overlaid on backward FTLE field. (c): Simple thresholding method used to obtain dominant regions. (d): Ridge detection algorithm used to obtain ridges of interest. These values in the colorbar are used for the remainder of the paper.

The results are organized as follows: we discuss the backward FTLE field overlaid with OMPS aerosol index data in the region of interest, making note of significant features, some of which we will analyze in greater detail in later sections. This subsection represents what could be done with FTLE in essentially real-time with enough computing power (only current and past data is used) and access to relatively accurate velocity data. Following this, we include the forward FTLE field to see what is gained by incorporating forecast wind data[3]. Then, we make comparisons with Eulerian quantities by highlighting areas where they provide insight and demonstrate that the FTLE field helps us understand, and in some cases predict, phenomena which the Eulerian quantities simply cannot. In subsequent figures we will highlight certain features we are describing with arrows or boxes if we feel they are not obvious from their description alone. In S4 of the Supplemental Material, we produce

[3]Here we still use MERRA-2 reanalysis data so this would be assuming the forecast data is as accurate as reanalysis data. Results would be effected if true forecast data was used.

the same figures that will be covered in sections 3.1 and parts of 3.2 but using velocity fields from the 600 hPa pressure to make sure we are not missing any important structures or observing any artifacts due to the averaging.

3.3.1 FTLE

On June 1, 2020, a vortex starts to form to the north-northwest of the plume, off the northwest coast of Africa. We refer to this as the early June vortex in later sections. We omit the figures showing its formation as this is less important. A video of the backward FTLE ridges overlaid on OMPS aerosol index data can be seen here. In addition, we provide a video of attracting LCS, computed using the variational method, overlaid on the backward FTLE field to confirm the structures we are focused on are true hyperbolic structures and not FTLE false positives due to shear. The vortex hovers in this area until about June 5 (Figure 3.6a), when a FTLE ridge forming the northern piece of this vortex (green arrow in 3.6a)) forms another vortex (green box Figure 3.6b) and they are both shepherded to the east while the original early June vortex dissipates and has little effect on the plume. As this happens, the secondary vortex breaks off a piece of the plume over northern Africa and pushes it towards southeast Asia (green arrow in Figure 3.6c). These type of vortex structures, made up of regions of high deformation, will be important in describing the evolution of the plume. In addition, there is a strong attracting ridge acting as the northern boundary for the plume. This ridge persists throughout a large portion of the plume's lifetime and prevents transport to the north.

For the next two days, the plume does not change in shape much and hovers over northwest Africa, moving north slightly as it is attracted to the strong ridge acting as its northerly boundary. On June 12, a new vortex begins to form in roughly the same place as the early

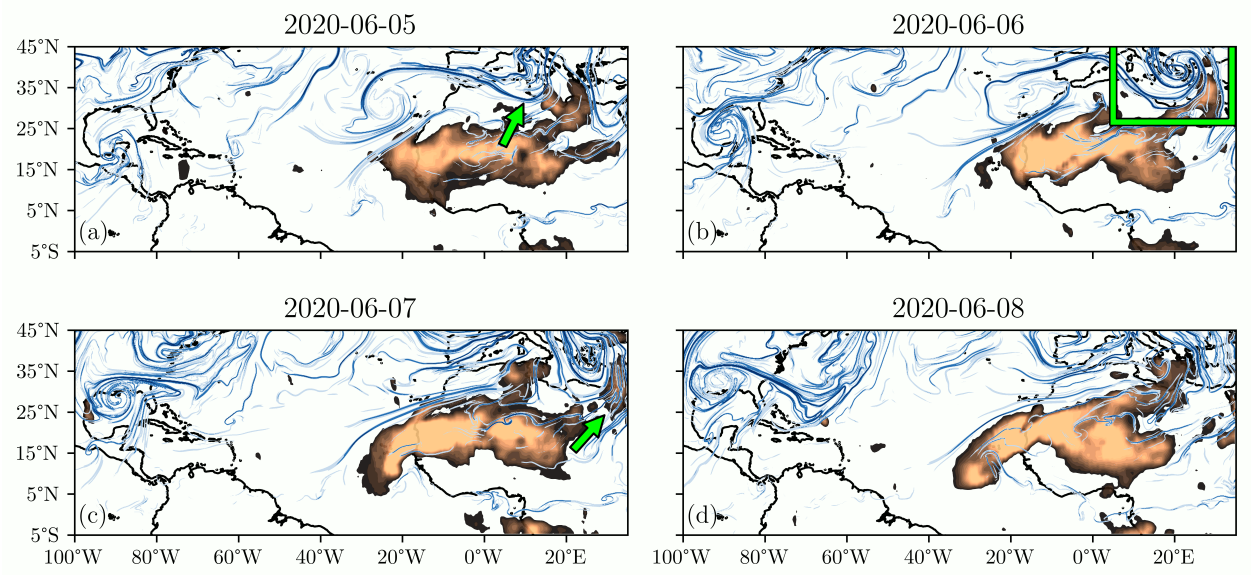

Figure 3.6: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 5-8, 2020. See text for details.

June vortex (off the coast of northwest Africa). We will refer to this vortex as the mid June vortex and first show it on June 15 (Figure 3.7a). At first, this vortex seems quite similar to the early June vortex and one might expect a similar minor effect on the plume. But this time, as the northern attracting ridge is attracted towards the vortex, the vortex is propelled towards the plume and collides with it directly. As it does, the vortex dissipates and spreads out but its effect on the plume is substantial. The vortex pushes down on the plume and drastically deforms its northern boundary, flattening and elongating it as it propels the western portion further west at a more rapid rate than it was originally traveling. We see the results of this collision in Figure 3.8. A mushrooming effect is taking place on June 19 (Figure 3.8a) as the western portion grows and begins to separate from the rest of the plume. This becomes more pronounced on June 20 (Figure 3.8b) and then a split happens on June 21 (Figure 3.8c); all the while a strong attracting ridge bounds the main plume to the north, its shape changed by the earlier collision. By June 23 (Figure 3.9a), the plume has fully split into separate pieces. As this happens a strong ridge runs between the two. Starting on June

24 (Figure 3.9b) a new feature becomes evident. We see a ridge running through the middle of the plume, acting in a similar manner to the air bridges seen in other transoceanic smoke transport [124]. Furthermore, a "FTLE front" becomes apparent (Figure 3.9c,d) that ushers the new mushrooming portion of the plume westward towards the Americas.

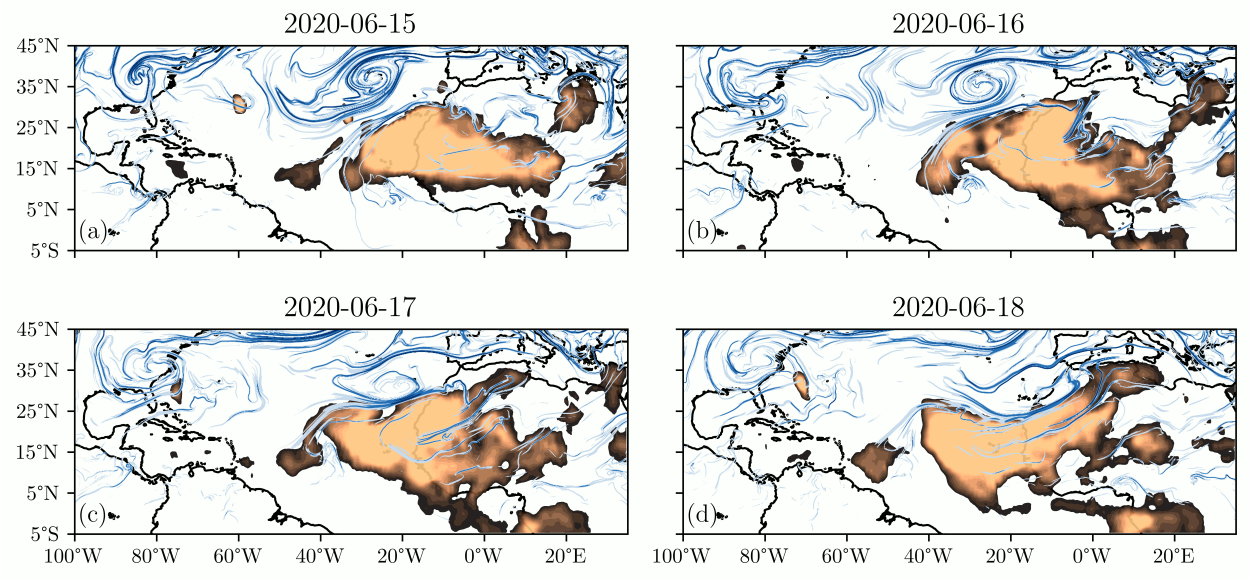

Figure 3.7: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 15-18, 2020.

In Figures 3.10, 3.11 and 3.12, both backward *and* forward FTLE ridges are overlaid with dust data (see video) to see what is gained from incorporating future (originally, forecast) data. On June 11 (Figure 3.10a) we notice some repelling ridges near the southwestern portion of the plume. On June 12 (Figure 3.10b), these ridges connect and, on the 13th (Figure 3.10c), we notice there are some repelling ridges intertwined with the strong mid June vortex structure mentioned previously. This behavior is usually indicative of cyclonic storm behavior (e.g., hurricanes) but storms in this region are not classified by NOAA so there is no mention of it in the literature. In addition, we can see the repelling ridge near the southern portion of the plume being drawn to and eventually connecting with the repelling ridges in the vortex. This connection creates "turnstile lobes" [31, 91, 102, 106] that could

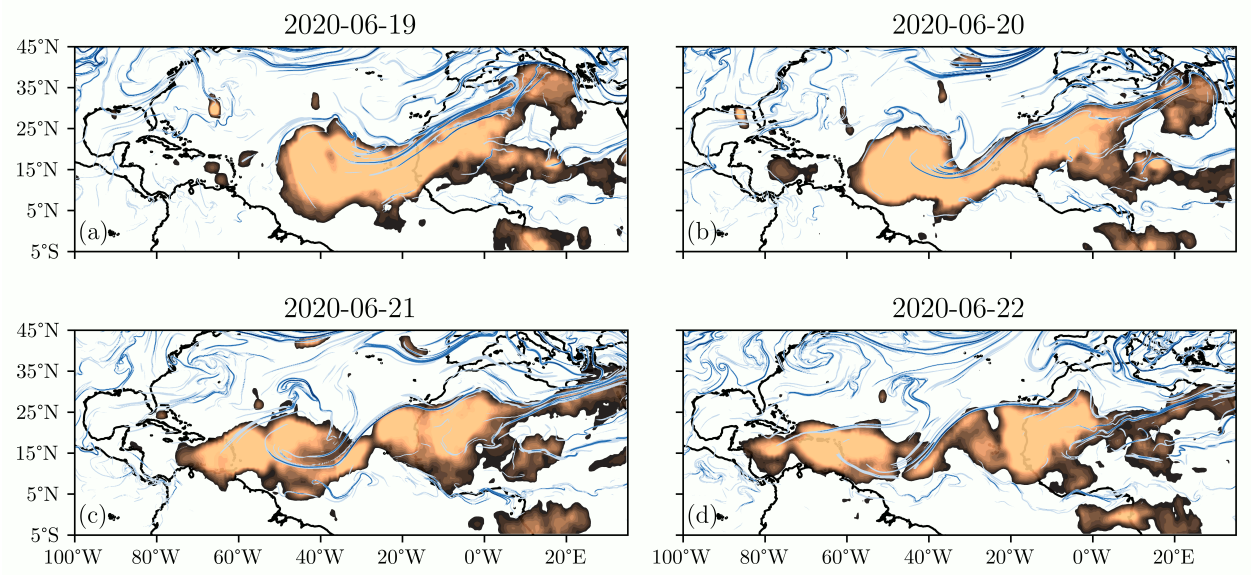

Figure 3.8: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 19-22, 2020.

be useful for identifying sets of particles that will remain separate from the vortex and those which will be entrained into it; see [38, 128, 129, 133]. For example, had this storm been over the plume and one had information about the locations of dust associated microbes present, predictions could be made about which of these would be ejected from the plume and subsequently entrained into the vortex as it traversed the Atlantic [135, 154].

On June 15 (Figure 3.11a), repelling ridges can be seen in between the vortex structure and the northern attracting ridge. On June 16 (Figure 3.11b), we can see one of these repelling ridges essentially being sandwiched between the two dominant attracting structures and being ejected towards the east at a rapid rate. This is behavior that is indicative of strong stretching (in dynamical systems terms, hyperbolic) events and we see this play out in subsequent days as the plume stretches and undergoes strong hyperbolic deformation, especially at the northern portion. We see similar behavior on June 18 (Figure 3.11d) with a few ridges intersecting the northerly attracting ridge and being pulled to the east, producing further hyperbolic deformation. Finally, during June 19-21 (first shown as green arrow in

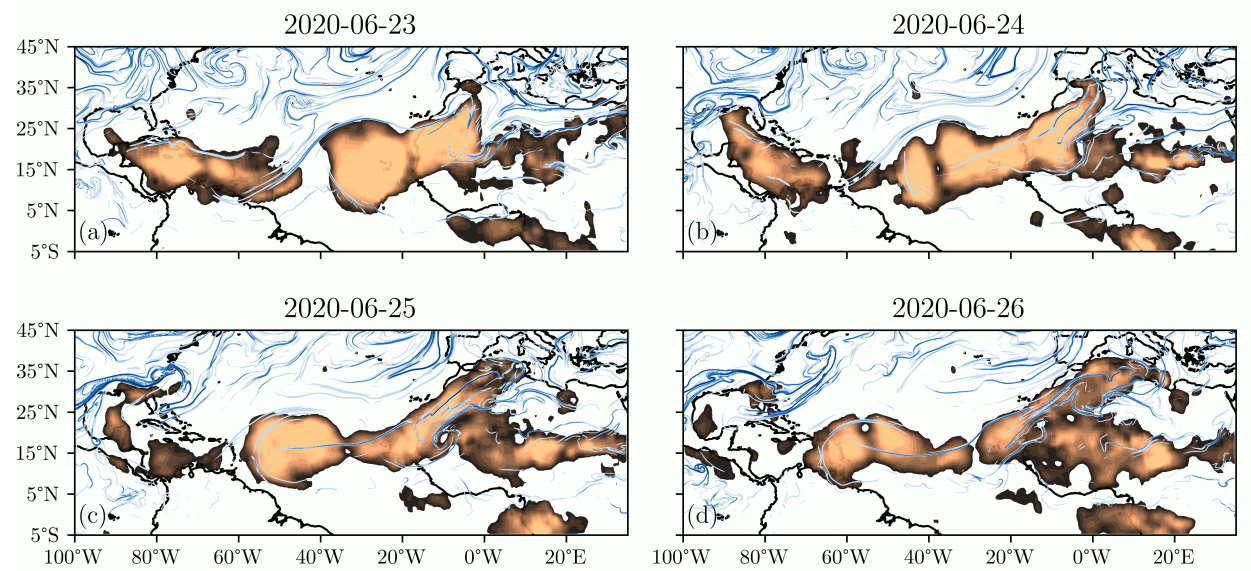

Figure 3.9: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 23-26, 2020.

Figure 3.12a), there is a repelling ridge intersecting the main attracting ridge. It is hard to say for sure with the temporal resolution for the dust data limited to 24 hour averages, but it seems this intersection point may lead to more hyperbolic behavior and assist in the eventual splitting of the plume.

3.3.2 Eulerian Combined with Lagrangian Analysis

In this subsection, we focus on some of the phenomena mentioned above and observe where Eulerian information is useful and where it falls short when compared to the Lagrangian diagnostics mentioned above. There are three main phenomena to discuss. First, we will compare the early June and mid June vortices and attempt to uncover why two seemingly similar structures had drastically different effects on the fate of the plume. Then, we will see if the strong northern ridge, which bounded the plume for most of its lifetime, could have been identified using Eulerian information. Finally, we will focus on the splitting of the

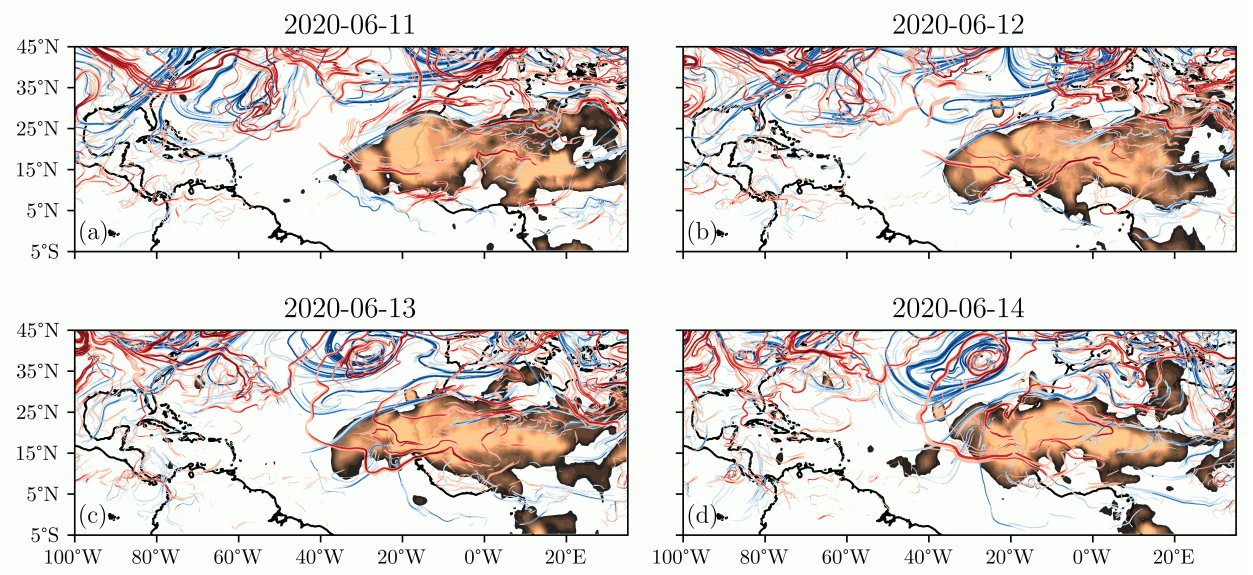

Figure 3.10: Forward and Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 11-14, 2020.

plume in late June and again see if any Eulerian quantities could have indicated that this would happen.

Early June and Mid June Vortex Comparison. In the following figures, we take a wider view and focus on the area where latitude runs from $5°\text{S} - 75°\text{N}$ and longitude runs from $120°\text{W} - 80°\text{E}$ to better see the effects of other FTLE features on the plume. We begin by identifying the vortices by looking at the mean sea level pressure (MSLP), low level vorticity (at 850 hPa), and backward Lagrangian averaged vorticity deviation (LAVD), which is computed for an integration time of 1 day using the averaged velocity fields. For these figures, the focus is on the region with center around 25°W, 35°N, off the northwest coast of Africa. On June 3, we notice a low in the MSLP (Figure 3.13a). On June 14, the MSLP (Figure 3.13b) in the region of the vortex is not quite as low, but is a low relative to the surrounding high. For the low level vorticity, we notice highs for both June 3 (Figure 3.13c) and June 14 (Figure 3.13a). In addition, we see local maxima in the LAVD field

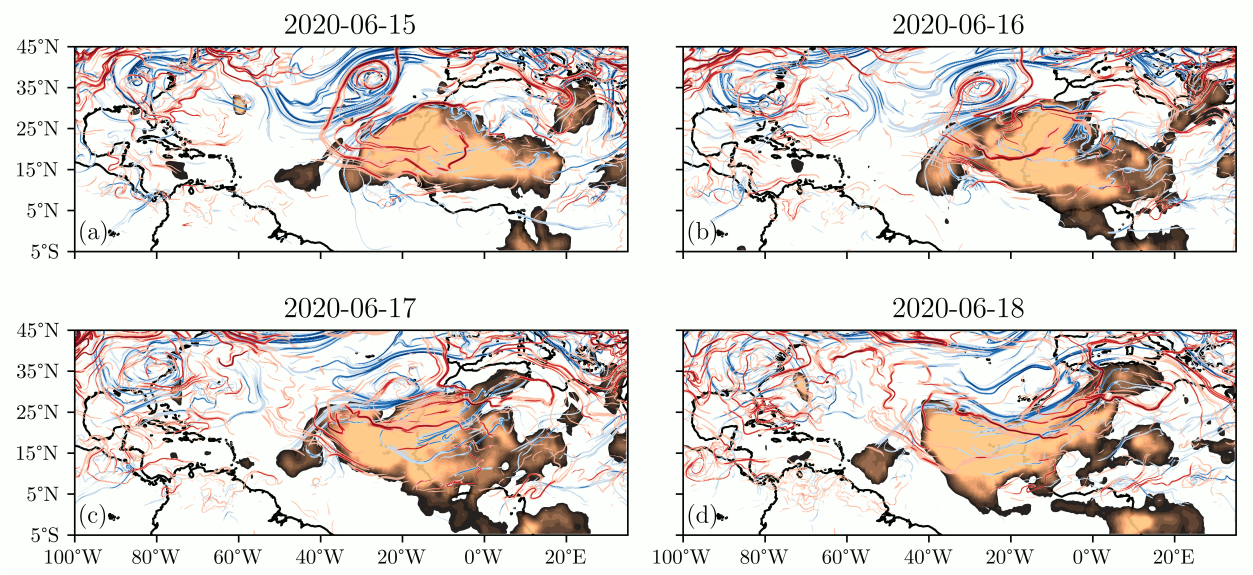

Figure 3.11: Forward and Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 15-18, 2020.

on both dates (Figure 3.13e,f), indicating a coherent vortex in the center of the storm that has persisted from at least one day ago. These figures seem to indicate that the early June vortex is the stronger of the two with a lower low and more coherent/strong center as seen in the vorticity and LAVD fields. Much of the same is drawn from the following day. Again the early June vortex shows as a low in the MSLP (Figure 3.14a) and the mid June vortex shows as a relative low while surrounded by a strong high (Figure 3.14b). Highs are again seen in both low level vorticity fields (Figure 3.14c,d) and local maxima can be seen in the LAVD fields (Figure 3.14e,f). The mid June vortex seems to be strengthening as can be seen by its more coherent vortex center. Through these figures we can confirm that we are indeed looking at true vortex structures. In addition, these storms can be seen in the satellite imagery as apparent on the NASA page for this dust event. Now, we move on to focusing on other features in the FTLE field which interact with these structures.

In the subsequent figures, the early and mid June vortex patterns are in different columns (early June left, mid June right), while we show the velocity field and FTLE field in the

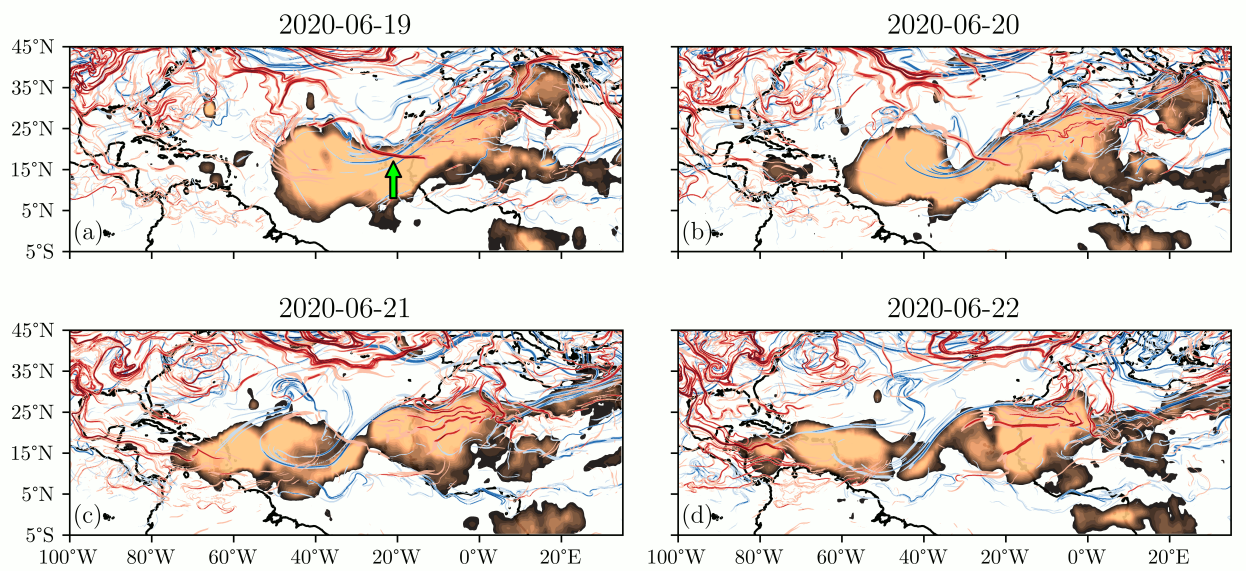

Figure 3.12: Forward and Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 19-22, 2020. See text for details.

rows (velocity top, FTLE bottom). A full video of the comparison of these two fields was also produced. The reader is encouraged to watch the comparison video as much of what is described in this section is much easier seen in the video. In addition, we again provide a video of attracting LCS overlaid on the backward FTLE field to confirm the structures we are focused on are truly hyperbolic. Looking at the velocity fields (Figure 3.15a,b), a counter-rotating vortex pattern is apparent [109], both in the early and mid June vortices identified earlier, but also in a larger vortex over northern Africa that kept the plume in place before the collision with the mid June vortex. Interestingly, one might argue the early June vortex is the "stronger" of the two and therefore would be the one having a greater effect on the plume. Observations could be made about the velocity field above the vortex structures, over Europe, and to the northwest, over North America. In early June there is a strong vortex to the north while this does not exist in mid June. In early June there are more vortex patterns to the northwest and more of a jet feature in mid June. In both of these cases it is hard to say with any confidence how these patterns will effect the early and

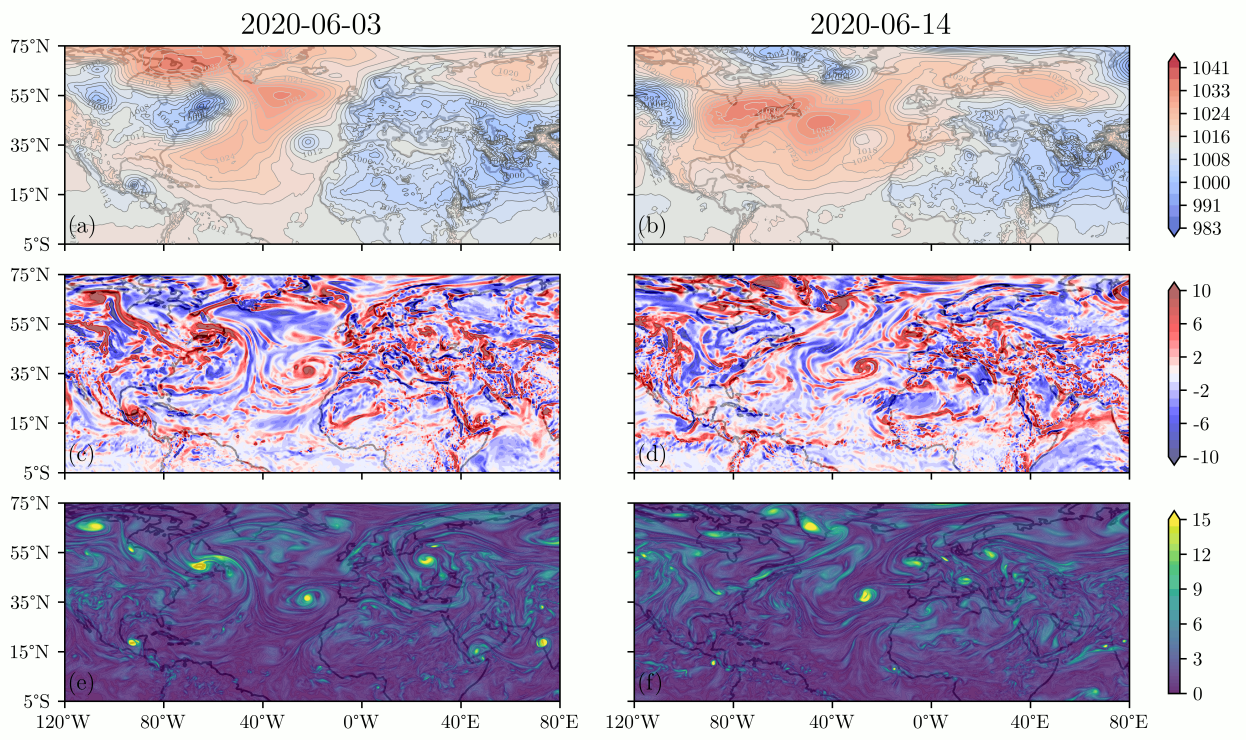

Figure 3.13: MSLP (hPa, top), low level vorticity at 850 hPa ($10^{-5}\times s^{-1}$, middle), and LAVD ($10^{-5}\times s^{-1}$, bottom) on June 3, 2020 (left) and June 14, 2020 (right).

mid June vortices. Conversely, the FTLE ridges yield influential material lines that will be advected with the flow and affect nearby particles as they come in contact. In early June, the ridge making up the northern portion of the vortex is long and is connected to other regions of the flow running north through eastern Europe (green box in Figure 3.15c) while the mid June vortex is less "connected" as its northern ridge remains separate from the strong ridge off the western coast of Europe and the one running through eastern Europe (green arrow Figure 3.15d). While not of immediate interest, we note the FTLE activity over North America. There is quite a bit more going on in mid June compared to early June which can be most easily seen in the video, where a jet feature is present in the FTLE field.

In Figure 3.16, we begin to fully see the distinction between the effects of the early and mid June vortices on the fate of the plume. There is a similar story in the velocity fields; one

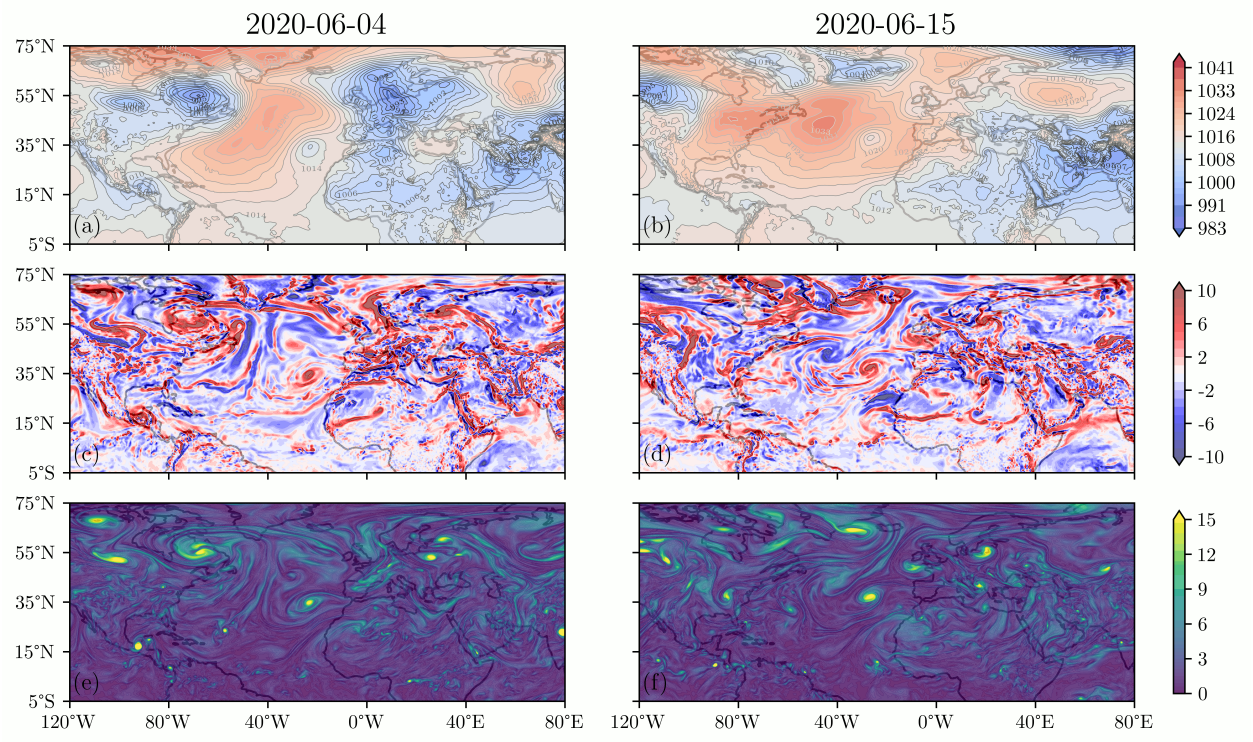

Figure 3.14: MSLP (hPa, top), low level vorticity at 850 hPa ($10^{-5}\times s^{-1}$, middle), and LAVD ($10^{-5}\times s^{-1}$, bottom) on June 4, 2020 (left) and June 15, 2020 (right).

notices differences between the early and mid June features. In early June (Figure 3.16a) it is difficult to see why the vortex was simply pulled towards the east in comparison to the trajectory of the mid June one (Figure 3.16b). There is a jet feature present in mid June to the northwest that is somewhat apparent in the Eulerian data but one cannot conclude from this data alone that it would have as great of an effect on the mid June vortex as it did. When we look at the FTLE fields, things become more clear. In early June, the vortex is being pulled along by the secondary vortex mentioned earlier, to which it is connected. In addition, it is being influenced and swept along by a large FTLE pattern over Europe (green box in Figure 3.16c). In mid June, the jet feature is now more apparent (magenta box in Figure 3.16d) with a small vortex being created at the end of the jet. This leads to a mushrooming effect with a FTLE ridge leading the way (green box in Figure 3.16d), which

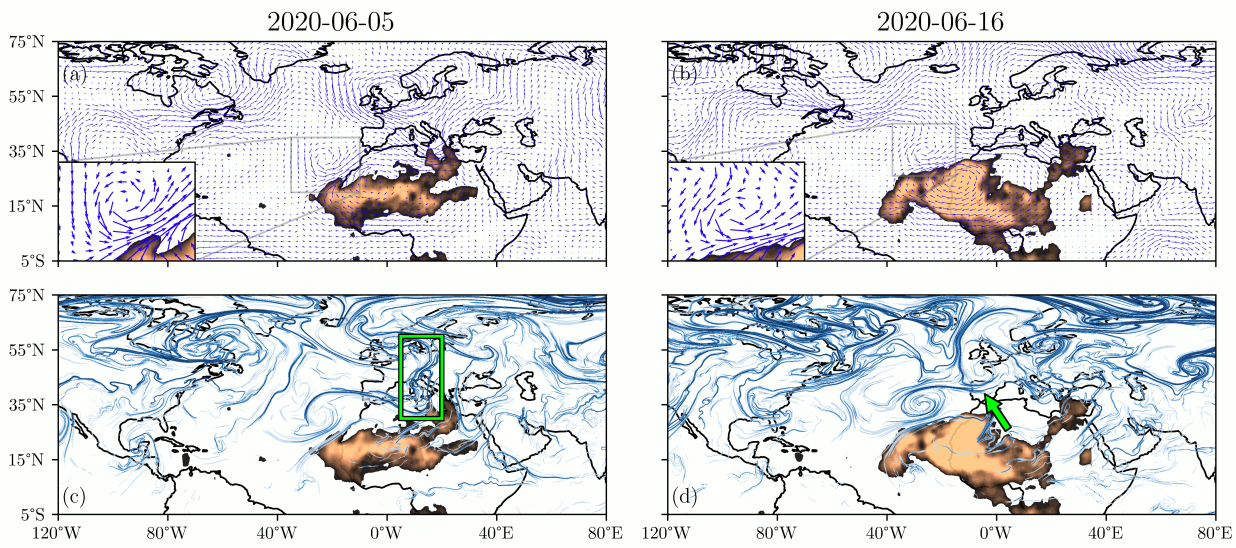

Figure 3.15: Velocity field (top) and backward FTLE ridges (bottom) overlaid on OMPS aerosol index data on June 5, 2020 (left) and June 16, 2020 (right). See text for details.

pushes on the vortex and propels it into the plume. In the final figure (Figure 3.17) we see the effects of these interactions and the drastic difference in the shape of the plume after the early and mid June vortices run their courses. The key difference here is that the velocity field shows the direction and instantaneous speed of a fixed point in space, allowing one to loosely infer where/how particles in the flow are actually moving while the FTLE field actually shows us how material in the flow is moving (by identifying the most influential material lines and *seeing* them move through space), allowing for much more detail and insight to be extracted when focusing on transport events.

Dust Plume Northern Boundary. There are other Eulerian quantities which can be used to compare to FTLE or help better analyze this transport event. We go through them briefly here, providing only a frame for each but with an accompanying video if the reader is interested. Consider the video performing the same comparison as was done between FTLE and velocity fields, but with the corresponding *streamfunction* instead. Many of the same

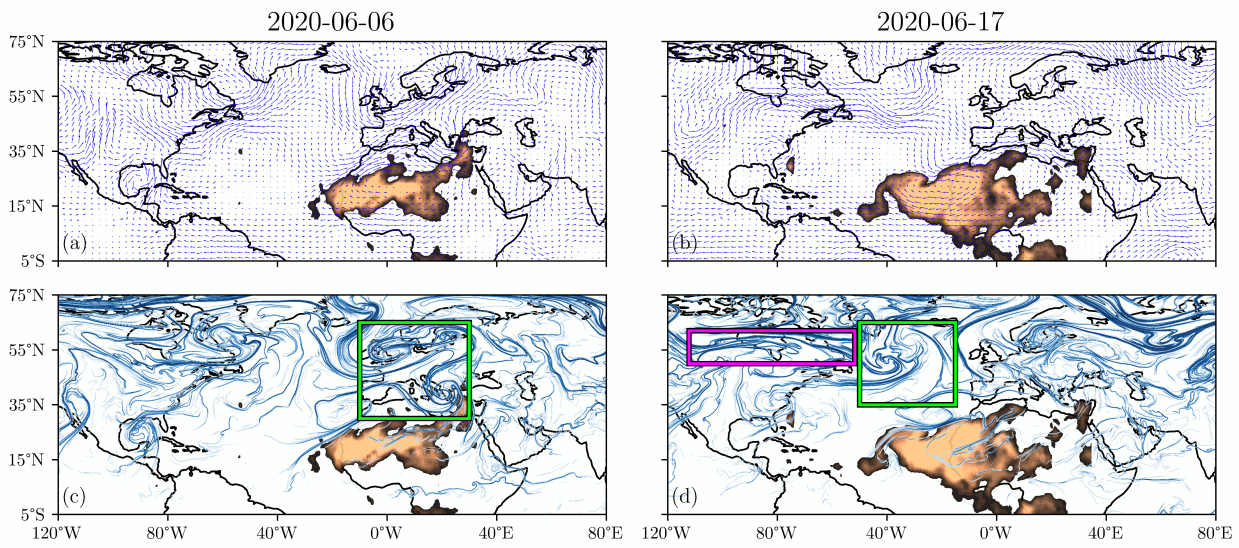

Figure 3.16: Velocity field (top) and backward FTLE ridges (bottom) overlaid on OMPS aerosol index data on June 6, 2020 (left) and June 17, 2020 (right). See text for details.

conclusions are drawn as with the previous comparison. Note that the streamfunction does not identify the northern boundary (see Figure 3.18 and video mentioned previously).

On the other hand, the *vorticity* field provides significant insight, capturing the strong northerly boundary FTLE ridge as a curve of vanishing vorticity; see Figure 3.19 (left) and the full video. Note that there are many other zero vorticity contours not associated with FTLE ridges. The FTLE ridge as the northern plume boundary instills confidence that it will indeed be a significant transport barrier, as it is objective (independent of the frame of reference) whereas vorticity is not objective and the FTLE takes into account information from the entire past time window.

Finally, we turn to an objective Eulerian diagnostic, the instantaneous Lyapunov exponent (iLE) field mentioned earlier. The iLE field quantifies instantaneous deformation and gives credence to the large effect of the mid June vortex collision with the plume. As seen in Figure 3.19 (right), as the mid June vortex collides with the plume we see very large iLE values along the plume's northern portion. Recall the iLE is the limit of the FTLE field as

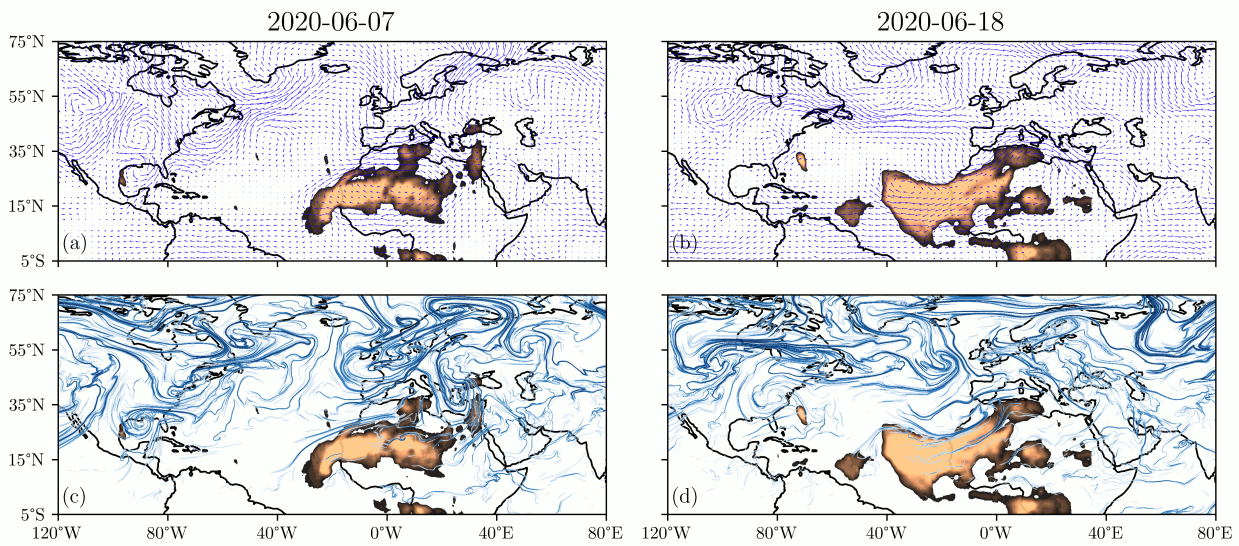

Figure 3.17: Velocity field (top) and backward FTLE ridges (bottom) overlaid on OMPS aerosol index data on June 7, 2020 (left) and June 18, 2020 (right).

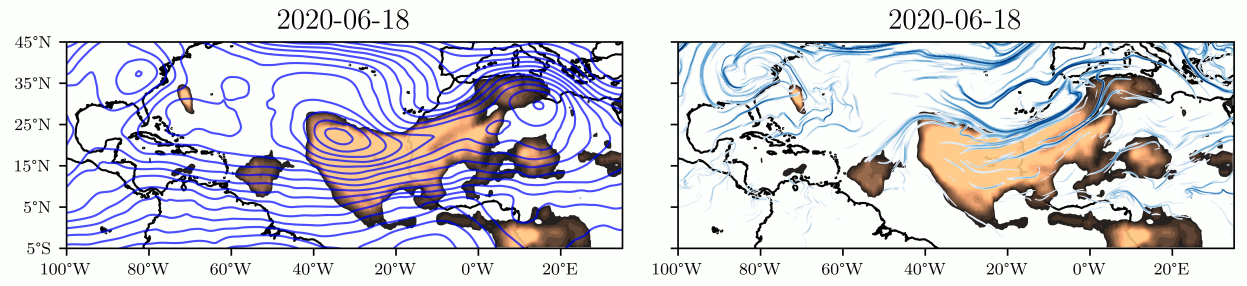

Figure 3.18: Left: Streamfunction overlaid on OMPS aerosol index data. Right: Backward FTLE overlaid on aerosol index data.

the integration time goes to zero, providing an objective diagnostic for identifying regions of high attraction and repulsion at an instant in time. The high values along the northern portion confirm that the plume underwent large scale deformation at the time of impact with the vortex. A full video of the iLE field overlaid on the dust data and backward FTLE ridges is provided here.

Plume Splitting Event. We end our analysis by looking at the splitting event of the plume around June 21. Recalling Figure 3.12a, we see an intersection point (green arrow)

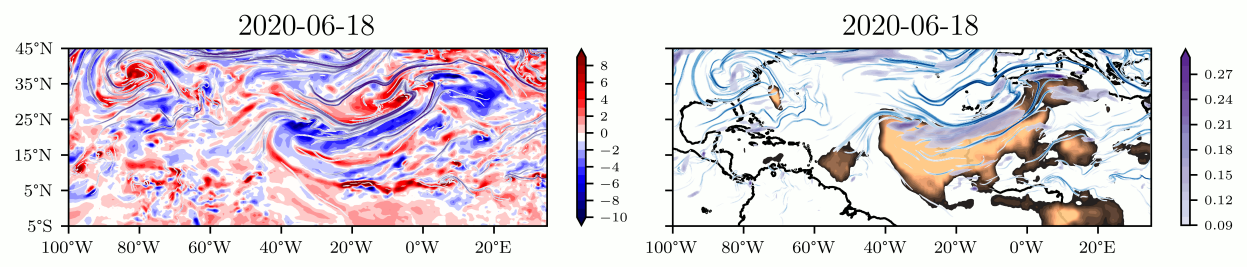

Figure 3.19: Left: Backward FTLE ridges (purple) overlaid on vorticity ($10^{-5}\times s^{-1}$, blue and red). Right: iLE ($10^{-6}\times s^{-1}$, purple) field overlaid on backward FTLE ridges (blue) and OMPS aerosol index data (copper) on June 18, 2020.

between the forward and backward FTLE ridges near where the splitting takes place. Forward and backward FTLE ridges can essentially be considered as time-dependent analogues of stable and unstable manifolds [19, 131, 147] and therefore, their intersections can be thought of as time-dependent analogues of saddle points[4]. These points can be important as they sometimes behave as moving, transient saddle points, having a similar effect, locally, on the flow nearby. Clearly they can be identified by looking for intersections between the forward and backward FTLE ridges (attracting and repelling LCS intersections are sometimes referred to as "generalized saddles") but we were curious if we could find these points, or regions near these points, by using only current and past data. We focus on the specific point we refer to at the beginning of this section as we believe it played a role in the splitting of the plume. We did not find any Eulerian data on its own that identified this point or captured its effect on nearby parcels. That being said, when we look at velocity vectors along the backward FTLE ridges, we can notice an inflection point in the direction of velocity vectors along the main northern boundary ridge, right around the intersection point (see Figure 3.20, left). While looking at velocity vectors along a moving Lagrangian ridge can sometimes be misleading, it is clear here that this point identifies an area of high stretching

[4]In addition to saddle-like points, these points can also behave like homoclinic orbits, homoclinic tangles, and primary intersection points (PIPs). Further investigation is needed to decipher when an intersection point behaves as one of the mentioned candidate points.

as seen by the large magnitude velocity vectors on either side of this point farther down the ridge. This point can be found by looking at the zero point of the tangential velocity along a ridge as seen is Figure 3.20, right. Note in this figure that the black dots represent the intersection point between the forward and backward ridges. The zero tangential velocity point lags slightly behind due to general westward motion in the flow. This is an example of when using Lagrangian and Eulerian data in conjunction can be beneficial.

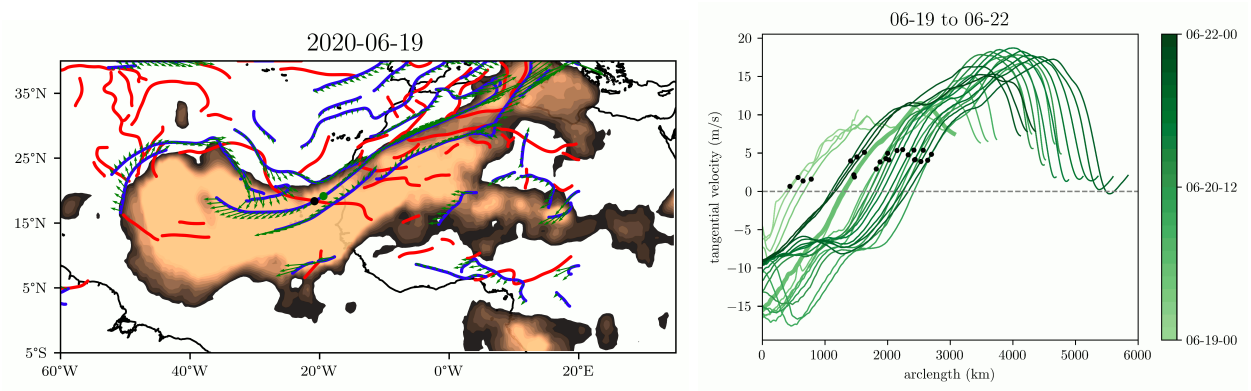

Figure 3.20: Left: Backward FTLE ridges (blue) with velocity vectors along ridge (green arrows) overlaid on forward FTLE ridges (red) and OMPS dust data (copper) on June 19, 2020. Black dot represents the intersection point while green dot represents zero tangential velocity point along ridge. Right: Tangential velocity value along portion of main northern ridge of the plume intersecting repelling ridge. The black dots represent the intersection point between the backward and forward ridge which led to the splitting of the plume with the dotted line representing the zero tangential velocity point. Shades of green represent date-time in 3 hour increments. Bold line corresponds to same date-time as left figure.

3.4 Discussion

Throughout the lifespan of the dust plume, a dominant ridge of the backward FTLE field acts as a northern boundary for the plume and plays a role in its evolution and eventual splitting. In addition, a vortex structure visible on June 15-16 moves south, intersecting the plume. This structure has a major effect on the plume, drastically changing its shape by flattening

and elongating it, propelling the western half towards the Americas at a more rapid rate, and leading to the eventual splitting of the plume. Overall, insight is gained from computing backward time Lagrangian coherent structures, which rely on only current and past velocity data. These structures can be of use in making qualitative, time-sensitive decisions. Had one been able to compute these structures as the event was occurring, one could identify the strong northern ridge and surmise it would act as a barrier, preventing any dust transport farther north. In addition, the identification of the strong vortex structure on June 15 and the jet-core to its northwest suggests an eventual drastic plume shape change, the speed-up of the western portion, and eventual plume splitting. After the fact, the backward FTLE field assists in understanding the drivers of transport during this massive dust event by identifying the key structures responsible for the fate of the plume.

What additional information do we gain from the utilizing the forward time FTLE structures, which would depend on wind forecasts? Interestingly, the main structures acting as boundaries and those acting as catalysts for deformation are captured within the backward FTLE field itself. The forward FTLE does offer some useful insight however, further demonstrating the strength of the mid June vortex and helping understand why it collided with the plume in the manner it did. The movement of air masses due to the vortical motion can be related to intersection points of attracting and repelling ridges and the resulting "lobe dynamics" [38, 108, 128].

As demonstrated by the velocity (similarly, streamfunction) and FTLE comparisons, it is clear that the FTLE field, being derived from current *and past* data, can provide more insight into transport events like this one. The velocity fields give simply a snapshot of data with no information about its past evolution. The FTLE ridges identify the material curves within the flow that are most influential on plumes of transported material, its deformation, and fate. As the most significant material lines, in terms of repulsion or attraction of nearby

material lines, they have a strong influence on material-laden fluid parcels that come near. This is apparent in mid June as a cascading effect takes place. A strong jet-core propels influential material lines and air mass towards the east, creating a small vortex which leads to a mushrooming effect that eventually collides with the mid June vortex, driving it into the plume and drastically effecting the evolution of the dust storm (see Figure 3.16). With this info in real time, a qualitative prediction could have been made about the effect the mid June vortex had on the plume. Using only the velocity fields, no such prediction was evident. For instance, there was not a streamline which acted as a northern plume boundary nor were insights about the fate of the plume captured in the velocity fields.

This is not to say the velocity fields are not useful; they highlight the counter-rotating vortex pattern and made it clear why the plume lingered for as long as it did. In addition, the vorticity field provided insight related to the strength of the counter-rotating vortex pair pattern and the potential alignment of zero vorticity curves with the plume northern boundary attracting ridge.

A number of different approaches could be used to study and perhaps attempt to predict transport in an event like this. Synoptic maps have been used to uncover patterns related to transport. Anomalies relative to the climatology in fields shown in these maps can point to potential exceptional dust events and highlight significant circulation features which play a role in the evolution of the plume [44, 126, 166]. These works mainly focus on anomalies in pressure conditions (geopotential heights, NASH) and resulting wind features (strong African eastetly jet (AEJ) and anomalous streamfunction patterns). These methods are very useful and highlight indicators of an extraordinary event while providing a coarser view of the transport. By contrast, FTLE provides finer detail by identifying the structures responsible for transport in a more precise manner. For example, these works point to the anomalous strength and position of the NASH and the associated anticyclonic circulation causing an

exceptionally strong AEJ which favored transport to the Americas [166]. Using FTLE, we identified the strong jet above North America which forced the vortex structure into the plume, causing it to split and propel each portion to the west and east respectively. Indeed, the anticyclonic circulation did strengthen the AEJ but our results suggest that the true culprit to its exceptional strength was the jet we identified that caused the storm to impact the plume. To the authors' knowledge, no work identified this interaction and the significance of these two features beyond noting the role of the anticyclonic circulation in strengthening the AEJ. In addition, we are not aware of any work that uncovered the northern boundary of the plume as the FTLE field did.

For another Lagrangian approach, trajectory models can be used which can calculate particle transport and dispersal in the atmosphere. In most of the literature, tools like these, e.g., the Hybrid Single-Particle Lagrangian Integrated Trajectory (HYSPLIT) model [149], are often used to compute backward trajectories and identify possible source regions for these exceptional events [39, 94]. In addition, forward trajectory models can be used to verify conclusions drawn from remote-sensing data and modeling [11], and to investigate long-term seasonal trends of dust plumes [104]. Using trajectory models like this can identify persistent source regions and transport pathways after an event which can be useful information for future events.

The HYSPLIT model has the ability to forecast particle trajectories but the authors are not aware of any work being done utilizing this propagator to forecast long-range dust transport, though it would be interesting to see the accuracy of this tool in a real-time prediction context. Since it would rely on forecast data, it can only be as accurate as the forecast data. One could argue that computing backward FTLE on forecast data would be more advantageous due to the information FTLE provides while relying on less forecast data (FTLE only needs 2D wind velocity while HYSPLIT will need 3D wind velocity and

additional meteorological conditions) and being relatively robust to uncertainty in velocity data [5]. As noted, we are not aware of this being done but would be interested to see the comparison.

Though we do not use FTLE in a true prediction context, applying FTLE in the manner we did (using only current and past data) in real-time could lead to qualitative inferences in a situation like this (e.g., the "northern boundary ridge" would act as a boundary and impede transport to the north, the unfolding of the jet-vortex interaction could imply the splitting of the plume and speed up transport to the Americas, etc.). Using it in this manner would be quick to implement as well as it would only necessitate the computation of the FTLE field at the current time (given we have computed backward FTLE up to the current time). We do not see how using HYSPLIT on only current and past data could provide similar insight in real-time.

We suggest that the FTLE field (and other coherent structure methods) be integrated as part of the atmospheric scientists toolbox when studying transport events like this and that Eulerian and Lagrangian diagnostics should be used in tandem to get the most out of the information available. This will provide the most complete picture of the transport after the fact and, more importantly, could provide the best information for making predictions in real-time.

3.5 Conclusion

Using NASA data for velocity fields (MERRA-2) and aerosol index values (OMPS) to represent the "Godzilla" dust plume of 2020, we demonstrated a simple implementation of the FTLE method for transport in geophysical flows. Even with significant simplification, these tools can be employed, essentially in real-time to assist in prediction of the transport of

dust or any other contaminant in the atmosphere (e.g., wildfire smoke [124]). Additionally, these tools can be used after the fact to identify key features responsible for the evolution of this massive dust plume and assist in better understanding the mechanisms by which the transport occurred.

We have shown the advantages of Lagrangian coherent structure (LCS) methods over traditional flow visualization techniques in the atmospheric sciences. By computing quantities which incorporate data outside of the current frame, a more complete picture of transport can be obtained. We do not suggest an abandonment of traditional techniques—far from it. Using Lagrangian tools such as the finite-time Lyapunov exponent (FTLE) field in conjunction with traditional methods and other Eulerian quantities mentioned previously provides the most comprehensive assessment of transport. We suggest researchers use all the tools and techniques available in an optimal manner to get the most out of the available data.

In future work, a more detailed analysis can be performed to obtain a more accurate portrait of the most influential structures of interest. As a caution, the FTLE does not provide information on parabolic and elliptic structures which may be important. Thus, performing parabolic and elliptic LCS computation could yield other important structures that were not captured in the purely hyperbolic investigation. We were able to infer the presence of a jet-core using the attracting structures by its effect on nearby material lines but parabolic LCS would have identified it without the need for inference. In addition, we identified coherent vortices using the LAVD for just a few frames. Employing this tool or other methods to find elliptic LCS for the entire time window could yield more insight. One could also compute FTLE at a number of different pressure surfaces to see how much the structures differ from the ones we obtain from the averaged velocity field and 600 hPa surface. In addition, these FTLE fields could be "stiched" together to create an approximation of the 3D structures. For a higher fidelity analysis, the FTLE could be computed from the 3D velocity field itself

and inertial particle effects of dust could be incorporated. However, comparison with ground truth data would be difficult as the dust data available here via satellite was only a 2D vertical column-averaged data, and only once every 24 hours, with little to no vertical resolution. Though not mentioned here, there is a complementary viewpoint which seeks to identify the coherent material sets themselves rather than the material curves that often bound them [48, 53, 150, 155] which has seen improvements in computationally efficiency in recent years [47]. A study performing a comparison of these methods in the atmospheric setting would be useful. Like the the work done for this dust event, we could perform similar computations for other large scale dust events and see if similar structures persist or different phenomena is observed. Coupling this transport modeling with microbiological analysis could assist in understanding similar past events and mitigating possible negative effects of future ones as well.

We will aim to pursue some of these avenues in future work with most of the focus on computing FTLE over the entire globe for some significant portion of the past to see if we can identify recurrent transport patterns to assist in prediction and obtain a historical record of global atmospheric FTLE. In addition, we have hopes that this data may shed some light on storm formation and intensification. By doing this and other studies, we intend to further demonstrate the usefulness of coherent structure methods to the atmospheric community in hopes of adding to the researchers toolkits when talking problems of this kind.

Ethics

This work does not include studies on human subjects, human data or tissue, or animals

Declaration of competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Albert Jarvis: Conceptualization, Formal analysis, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing - original, Writing - review & editing. **Ali Hossein Mardi**: Conceptualization, Data curation, Investigation, Methodology, Resources, Software, Validation, Writing - original, Writing - review & editing. **Hosein Foroutan**: Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Writing - review & editing. **Shane Ross**: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Writing - review & editing.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the NASA MITAD project NASA MITAD: Microbes In Trans-Atlantic Dust grant no. 80NSSC20K1532 issued through the NASA Interdisciplinary Research in Earth Science (IDS) program. The authors thank an anonymous reviewer for insightful comments and suggestions which have significantly enhanced the quality of the manuscript.

Chapter 4

Diffusive Flux-Rate Barriers for General Diffusion Structure

Attribution

This chapter, a collaborative work with Shane Ross, is a manuscript currently in preparation.

Author Contributions

Jarvis and Ross conceived this project. Jarvis developed the theory, designed the implementation, implemented algorithms, designed examples, and wrote the original manuscript. Jarvis and Ross contributed to revisions.

Abstract

The identification of transport barriers in fluid flows has long focused on Lagrangian coherent structures (LCS), which describe barriers to purely advective transport. More recently, the notion of coherence has been revisited to account for diffusive effects, leading to a theory of elliptic transport barriers that extremize diffusive transport in advection-dominated

regions. While this theory has been developed in the finite-time setting for general diffusion structures, an analogous instantaneous (Eulerian) formulation has remained lacking. In the special case of steady, homogeneous, isotropic diffusion, it has been shown that barriers extremizing the diffusive flux-rate coincide with purely advective, objective Eulerian coherent structures. In this work, we extend the framework to define diffusive flux-rate barriers in the instantaneous case for general, potentially anisotropic, unsteady, and inhomogeneous, diffusion structures. We illustrate how these barriers can differ significantly from their isotropic counterparts through a toy example designed to highlight the impact of complex diffusion structure on transport barrier geometry.

4.1 Introduction

Understanding how a nonautonomous flow's contents are organized over some finite time window can be very useful for determining where a substance under the action of the flow may or may not go over that time window. Applications range from analyzing some geophysical phenomena [38, 49, 95, 129, 130, 132, 144] to better understanding transport of some contaminant in a geophysical flow [32, 75, 111, 117, 124, 135]. In these examples, and many others in geophysical and engineering applications, the material being transported through the flow is influenced by both advective and diffusive contributions, though the diffusive contributions are orders of magnitudes smaller than the advective ones. The theory of LCS was developed in the purely advective setting and did not account for diffusion. With this in mind, standard LCS were defined via a variational formulation which proposed them as material surfaces which extremized the normal repulsion or attraction rate [59] in the hyperbolic case and as closed material surfaces which stretch by a uniform factor (to leading order) over the finite time window in the elliptic case [61]. More recently, theory was developed for

the instantaneous case which looks at these structures as the integration time $\tau \to 0$. Out of this work, objective Eulerian coherent structures were born [141] which characterize important short-term transport structures. Both the Lagrangian and Eulerian structures have received a great deal of attention and been used extensively, to great success, in physical applications like those mentioned above. More recently, Haller et al. [68] extended the concept of transport barriers by incorporating diffusion in the weakly diffusive regime—a regime relevant to many physical transport processes. They introduced a new tensorial framework (based on the transport tensor and the averaged diffusion-weighted Cauchy-Green strain tensor), which yields material surfaces that extremize diffusive transport. This formulation overcomes a limitation of the purely advective case, in which material flux is always zero and thus ill-defined as a measure of coherence. In their work, Haller and coauthors developed a theory of material barriers to diffusive and stochastic transport for general diffusion structure in the finite-time setting, and for steady, homogeneous, isotropic diffusion in the instantaneous (Eulerian) case. While this latter setting will be the more common case, we extend their framework to define instantaneous barriers for general diffusion structure. This effort provides a step toward developing general Eulerian counterparts to Lagrangian diffusive barriers. This paper is organized as follows: we begin with a brief overview of coherent structures in the purely advective setting. We then review the recent developments in the diffusive and stochastic transport framework. Following this, we present our extension of the instantaneous theory to general diffusion structure. We illustrate the key differences through an example comparing barriers arising from isotropic and general diffusive theories. We conclude with remarks and directions for future work.

4.2 Background

We will start by (very briefly) reviewing elliptic advective coherent structures in 2 dimensions. We refer the reader to the cited articles [61, 141] for more details. Assume we have a time-varying velocity field $\mathbf{v}(\mathbf{x}, t)$ that is C^2 in space and C^1 in time. Consider the initial value problem,

$$\frac{d}{dt}\mathbf{x} = \mathbf{v}(\mathbf{x}, t), \quad x \in U \subset \mathbb{R}^n, \quad t \in I \subset \mathbb{R} \tag{4.1}$$

Then there exists a family of diffeomorphisms $\{\mathbf{F}_{t_0}^t\}$ associated with velocity field $\mathbf{v}(\mathbf{x}, t)$ known as the flow maps which advect particles through the flow,

$$\begin{aligned} \mathbf{F}_{t_0}^t &: U \to U \\ &: \mathbf{x}_0 \mapsto \mathbf{x}(t; t_0, \mathbf{x}_0) \end{aligned} \tag{4.2}$$

We can define the linearized flow map as the gradient of the flow map with respect to initial conditions,

$$\nabla_0 \mathbf{F}_{t_0}^t(\mathbf{x}_0) = \frac{\partial \mathbf{F}_{t_0}^t(\mathbf{x}_0)}{\partial \mathbf{x}_0}. \tag{4.3}$$

The right Cauchy-Green deformation tensor is defined as,

$$\mathbf{C}_{t_0}^t(\mathbf{x}_0) = \left(\nabla_0 \mathbf{F}_{t_0}^t(\mathbf{x}_0)\right)^\top \nabla_0 \mathbf{F}_{t_0}^t(\mathbf{x}_0) \tag{4.4}$$

which is symmetric and positive-definite. This implies that $\mathbf{C}_{t_0}^t(\mathbf{x}_0)$ has real eigenvalues λ_i

and corresponding orthonormal eigenvectors $\boldsymbol{\xi}_i$ with $i \in \{1, 2, ..., n\}$ such that,

$$\lambda_1 \geq ... \geq \lambda_n > 0 \text{ and,} \tag{4.5}$$

$$\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j \rangle = \delta_{ij}. \tag{4.6}$$

The Eulerian rate-of-strain tensor is defined as the symmetric part of the spatial gradient of velocity,

$$\mathbf{S}(\mathbf{x}, t) = \frac{1}{2} \left(\nabla \mathbf{v}(\mathbf{x}, t) + (\nabla \mathbf{v}(\mathbf{x}, t))^\top \right) \tag{4.7}$$

which is symmetric, implying it has real eigenvalues s_i and corresponding orthonormal eigenvectors $\mathbf{e}_i$ with $i \in \{1, 2, ..., n\}$ such that,

$$s_1 \geq ... \geq s_n \text{ and,} \tag{4.8}$$

$$\langle \mathbf{e}_i, \mathbf{e}_j \rangle = \delta_{ij}. \tag{4.9}$$

For short times, $\mathbf{S}$ quantifies the deviation of $\mathbf{C}$ from the identity, i.e.,

$$\mathbf{C}_{t_0}^t(\mathbf{x}_0) = \mathbf{I} + 2\mathbf{S}(\mathbf{x}_0, t_0)(t - t_0) + \mathcal{O}(|t - t_0|^2).\mathfrak{z} \tag{4.10}$$

4.2.1 Lagrangian

Lagrangian elliptic coherent structures are defined in terms of the averaged tangential material strain for a given closed curve γ [61]. To start, parameterize a closed curve γ such that $\gamma = \mathbf{r}(s)$ for $s \in [0, \sigma]$ and $\gamma(\sigma) = \gamma(0)$. Let $\mathbf{r}'(s)$ represent the tangent vector to γ at $\mathbf{r}(s)$ and let $\frac{d}{ds}\mathbf{F}_{t_0}^t(\mathbf{r}(s))$ denote the corresponding tangent vector along the advected curve

$\mathbf{F}_{t_0}^{t}(\gamma)$. Then, the lengths of these vectors can be calculated by,

$$\|\mathbf{r}'(s)\| = \sqrt{\langle \mathbf{r}'(s), \mathbf{r}'(s) \rangle}, \quad \left\| \frac{d}{ds} \mathbf{F}_{t_0}^{t}(\mathbf{r}(s)) \right\| = \sqrt{\left\langle \mathbf{r}'(s), \mathbf{C}_{t_0}^{t} \mathbf{r}'(s) \right\rangle}. \tag{4.11}$$

Let,

$$q(\mathbf{r}(s), \mathbf{r}'(s), t) = \frac{\sqrt{\left\langle \mathbf{r}'(s), \mathbf{C}_{t_0}^{t}(\mathbf{r}(s)) \mathbf{r}'(s) \right\rangle}}{\sqrt{\langle \mathbf{r}'(s), \mathbf{r}'(s) \rangle}}, \tag{4.12}$$

represent the pointwise tangential strain. Then, the *averaged tangential strain* along γ is given by,

$$Q(\gamma) = \frac{1}{\sigma} \int_{\gamma} q(\mathbf{r}(s), \mathbf{r}'(s), t) ds. \tag{4.13}$$

Proceeding with the variational calculus, Haller and Beron-Vera [61] show that extremizers of this functional are limit cycles of the following differential equations,

$$\mathbf{r}'(s) = \boldsymbol{\eta}_{\lambda}^{\pm}(\mathbf{r}(s)), \quad \boldsymbol{\eta}_{\lambda}^{\pm}(\mathbf{r}(s)) = \sqrt{\frac{\lambda^2 - \lambda_2}{\lambda_1 - \lambda_2}} \boldsymbol{\xi}_1(\mathbf{r}(s)) \pm \sqrt{\frac{\lambda_1 - \lambda^2}{\lambda_1 - \lambda_2}} \boldsymbol{\xi}_2(\mathbf{r}(s)), \tag{4.14}$$

defined on the domain,

$$U_{\lambda} = \{ \mathbf{x}_0 \in U : \lambda_1 \neq \lambda_2, \ \lambda_2 < \lambda^2 < \lambda_1 \}. \tag{4.15}$$

.

4.2.2 Eulerian

The Eulerian formulation follows a similar setup, but the objective shifts to identifying closed curves that extremize the *averaged tangential material stretch-rate* [141],

$$\dot{Q}(\gamma) = \frac{1}{\sigma}\int_{\gamma} \dot{q}(\mathbf{r}(s), \mathbf{r}'(s), t)ds, \tag{4.16}$$

with,

$$\dot{q}(\mathbf{r}(s), \mathbf{r}'(s), t) = \frac{\langle \mathbf{r}'(s), \mathbf{S}(\mathbf{r}(s), t)\mathbf{r}'(s)\rangle}{\langle \mathbf{r}'(s), \mathbf{r}'(s)\rangle}, \tag{4.17}$$

Noting similarities to the variational problem in the Lagrangian case from the previous section, the authors conclude extremizers of $\dot{Q}(\gamma)$ will coincide with limit cycles of the following differential equations,

$$\mathbf{r}'(s) = \boldsymbol{\chi}_{\mu}^{\pm}(\mathbf{r}(s)), \quad \boldsymbol{\chi}_{\mu}^{\pm}(\mathbf{r}(s)) = \sqrt{\frac{\mu - s_2}{s_1 - s_2}}\mathbf{e}_1(\mathbf{r}(s)) \pm \sqrt{\frac{s_1 - \mu}{s_1 - s_2}}\mathbf{e}_2(\mathbf{r}(s)) \tag{4.18}$$

defined on the domain,

$$U_{\mu} = \{\mathbf{x}_0 \in U : s_1 \neq s_2,\ s_2 < \mu < s_1\}. \tag{4.19}$$

4.3 Advection-diffusion transport

Next we present what was done for diffusive barriers in the Lagrangian case by Haller et al. [68]. Then, we apply similar arguments as in [141] to define instantaneous Eulerian diffusive barriers.

4.3.1 Lagrangian

For a given time-varying velocity field $\mathbf{v}(\mathbf{x},t)$, the tracer field $c(\mathbf{x},t)$ evolves according to the classic advection-diffusion equation,

$$\frac{\partial c}{\partial t} + \nabla \cdot (c\mathbf{v}) = \nu \nabla \cdot (\mathbf{D}\nabla c), \quad c(\mathbf{x},t_0) = c_0(\mathbf{x}), \tag{4.20}$$

where $\mathbf{x} \in U \subset \mathbb{R}^n$ and $\mathbf{D}(\mathbf{x},t) \in \mathrm{M}_n(\mathbb{R})$ is the symmetric, positive definite diffusion-structure tensor. The parameter $\nu > 0$ represents the diffusivity and is assumed to be small so that the transport of $c(\mathbf{x},t)$ is dominated by advection. For our purposes, we assume both the concentration $c(\mathbf{x},t)$ and diffusion structure tensor $\mathbf{D}(\mathbf{x},t)$ are C^2. Then we have the flow maps $\{\mathbf{F}_{t_0}^t\}$ as defined above (Eq. (4.2)), for an arbitrary smooth material surface $\mathcal{M}_t = \mathbf{F}_{t_0}^t(\mathcal{M}_0)$ with initial condition given by $\mathcal{M}_0 = \mathcal{M}_{t_0} \subset U$, the total transport of $c(\mathbf{x},t)$ through this surface, over the time window $[t_0,t_1]$, is given by,

$$\Sigma_{t_0}^{t_1}(\mathcal{M}_t) = \int_{t_0}^{t_1} \int_{\mathcal{M}_t} \nu \mathbf{D}\nabla c \cdot \mathbf{n}\, dA\, dt \tag{4.21}$$

where $\mathbf{n}(\mathbf{x},t)$ is the normal to the surface $\mathcal{M}_t$ for all $\mathbf{x} \in \mathcal{M}_t$ with $\mathbf{n}_0(\mathbf{x}_0)$ the normal to $\mathcal{M}_0$ for all $\mathbf{x}_0 \in \mathcal{M}_0$. Then, we can reformulate the total transport in terms of Lagrangian coordinates by noting $\mathbf{n}\, dA = \det(\nabla_0 \mathbf{F}_{t_0}^t)[\nabla_0 \mathbf{F}_{t_0}^t]^\top \mathbf{n}_0 dA_0$ [55] and applying the chain rule to $\nabla_0 c$ to obtain ∇c in terms of the material coordinates. Note that $\nabla = \frac{\partial}{\partial \mathbf{x}}$ represents the spatial gradient and $\nabla_0 = \frac{\partial}{\partial \mathbf{x}_0}$ represents the material gradient. The material and spatial gradients are related in the following manner [55],

$$\nabla_0 c = (\nabla_0 \mathbf{F}_{t_0}^t)^\top \nabla c \implies \nabla c = (\nabla_0 \mathbf{F}_{t_0}^t)^{-\top} \nabla_0 c \tag{4.22}$$

Then, the total transport is given by,

$$\begin{aligned}
\Sigma_{t_0}^{t_1}(\mathcal{M}_0) &= \nu \int_{t_0}^{t_1} \int_{\mathcal{M}_0} \mathbf{D}(\mathbf{F}_{t_0}^t, t)[\nabla_0 \mathbf{F}_{t_0}^t]^{-\top} \nabla_0 c(\mathbf{F}_{t_0}^t, t) \cdot [\nabla_0 \mathbf{F}_{t_0}^t]^{-\top} \mathbf{n}_0 \, dA_0 \, dt \\
&= \nu \int_{t_0}^{t_1} \int_{\mathcal{M}_0} \left\langle \mathbf{D}(\mathbf{F}_{t_0}^t, t)[\nabla_0 \mathbf{F}_{t_0}^t]^{-\top} \nabla_0 c(\mathbf{F}_{t_0}^t, t),\, [\nabla_0 \mathbf{F}_{t_0}^t]^{-\top} \mathbf{n}_0 \right\rangle \, dA_0 \, dt \\
&= \nu \int_{t_0}^{t_1} \int_{\mathcal{M}_0} \left\langle \nabla_0 c(\mathbf{F}_{t_0}^t, t),\, [\nabla_0 \mathbf{F}_{t_0}^t]^{-1} \mathbf{D}(\mathbf{F}_{t_0}^t, t)[\nabla_0 \mathbf{F}_{t_0}^t]^{-\top} \mathbf{n}_0 \right\rangle \, dA_0 \, dt \\
&= \nu \int_{t_0}^{t_1} \int_{\mathcal{M}_0} [\nabla_0 c(\mathbf{F}_{t_0}^t, t)]^\top \mathbf{T}_{t_0}^t \mathbf{n}_0 \, dA_0 \, dt,
\end{aligned} \tag{4.23}$$

where $\mathbf{T}_{t_0}^t \coloneqq [\nabla_0 \mathbf{F}_{t_0}^t]^{-1} \mathbf{D}(\mathbf{F}_{t_0}^t, t)[\nabla_0 \mathbf{F}_{t_0}^t]^{-\top}$ is a symmetric positive-definite tensor which we will refer to as the diffusive Lagrangian flux tensor.

As shown by Haller et al. [68], this can be rewritten as,

$$\Sigma_{t_0}^{t_1}(\mathcal{M}_0) = \nu \int_{t_0}^{t_1} \int_{\mathcal{M}_0} [\nabla_0 c_0]^\top \mathbf{T}_{t_0}^t \mathbf{n}_0 \, dA_0 \, dt + o(\nu) \tag{4.24}$$

To find material surfaces that extremize diffusive transport across them, the authors propose an initial concentration such that $\mathcal{M}_0$ is a level set of the initial concentration (this implies $\nabla_0 c_0(\mathbf{x}_0) = K \mathbf{n}_0(\mathbf{x}_0)$ for $K > 0$ and $\mathbf{n}_0$ defined previously) and therefore, the concentration will diffuse directly through $\mathcal{M}_0$ (due to Fick's law) and will identify barriers that will impede (or enhance) diffusion maximally in the most diffusive prone context. The total transport then becomes,

$$\Sigma_{t_0}^{t_1}(\mathcal{M}_0) = K\nu \int_{t_0}^{t_1} \int_{\mathcal{M}_0} \mathbf{n}_0^\top \mathbf{T}_{t_0}^t \mathbf{n}_0 \, dA_0 \, dt + o(\nu) \tag{4.25}$$

From here, Haller rewrites this in the following manner,

$$\Sigma_{t_0}^{t_1}(\mathcal{M}_0) = \nu K (t_1 - t_0) \int_{\mathcal{M}_0} \left\langle \mathbf{n}_0, \bar{\mathbf{T}}_{t_0}^{t_1} \mathbf{n}_0 \right\rangle dA_0 + o(\nu) \tag{4.26}$$

where the diffusive transport tensor $\bar{\mathbf{T}}_{t_0}^{t_1}$ is the the time average of $\mathbf{T}_{t_0}^{t_1}$,

$$\bar{\mathbf{T}}_{t_0}^{t_1} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} \mathbf{T}_{t_0}^{t} dt \tag{4.27}$$

The total transport is normalized by dividing by the diffusivity parameter ν, the integration time $(t_1 - t_0)$, the magnitude of the initial concentration gradient K, and the area of the material surface $A(\mathcal{M}_0)$. This yields a dimensionless characterization of transport,

$$\tilde{\Sigma}_{t_0}^{t_1}(\mathcal{M}_0) \coloneqq \frac{\Sigma_{t_0}^{t_1}(\mathcal{M}_0)}{\nu K (t_1 - t_0) A(\mathcal{M}_0)} = \mathcal{T}_{t_0}^{t_1}(\mathcal{M}_0) + \mathcal{O}(\nu^{\alpha}), \tag{4.28}$$

for some $\alpha \in (0, 1)$. The *normalized transport functional* is defined as,

$$\mathcal{T}_{t_0}^{t_1}(\mathcal{M}_0) \coloneqq \frac{\int_{\mathcal{M}_0} \left\langle \mathbf{n}_0, \bar{\mathbf{T}}_{t_0}^{t_1} \mathbf{n}_0 \right\rangle dA_0}{\int_{\mathcal{M}_0} dA_0}, \tag{4.29}$$

which provides a leading-order estimate of diffusive transport through $\mathcal{M}_t$ over the time interval $[t_0, t_1]$.

Solving the corresponding variational problem leads to identifying limit cycles of a modified tensor field defined by $\bar{\mathbf{T}}_{t_0}^{t_1}$, analogous to the advective case, but with the transport tensor replacing the Cauchy-Green tensor.

Importantly, this approach detailed above yields a predictive measure of diffusive transport barriers in advection-dominated flows, without requiring explicit simulation of the full advection-diffusion dynamics. That is, the barrier structures can be computed by solving low-dimensional ordinary differential equations (ODEs) derived from the variational formulation, bypassing the need to numerically solve the advection-diffusion partial differential equations (PDEs) directly.

While not a focus of this paper, based on necessary conditions for transport barriers, the authors also defined the diffuisve barrier strentgh (DBS) which can be viewed as the diffusive counterpart to the finite-time Lyapunov exponent (FTLE) and is defined as the trace of the transport tensor,

$$\mathrm{DBS}(\mathbf{x}_0) \coloneqq \mathrm{trace}\left(\bar{\mathbf{T}}_{t_0}^{t_1}(\mathbf{x}_0)\right). \tag{4.30}$$

Much like the FTLE field, the DBS can highlight regions of strong hyperbolic behavior.

4.3.2 Eulerian

We now seek structures that instantaneously extremize the normalized diffusive transport as integration time approaches zero. We note that the normalized transport functional can also be viewed as the time-averaged diffusive Lagrangian flux functional. Therefore, identifying structures that extremize instantaneous diffusive transport is equivalent to seeking those that extremize the time-averaged diffusive Lagrangian flux $\bar{\mathbf{T}}_{t_0}^{t_1}$ and, in the limit as $t_1 \to t_0$, this reduces to extremizing the diffusive flux rate at time t_0.

We highlight two important results that will be used in what follows. First, the limit of the time derivative of the diffusive transport tensor as the integration time approaches zero is given by,

$$\lim_{t_1 \to t_0} \frac{d}{dt_1} \bar{\mathbf{T}}_{t_0}^{t_1} = \dot{\mathbf{T}}_{t_0}, \tag{4.31}$$

and the limit of the time derivative of the Lagrangian flux tensor as the integration time approaches zero is given by,

$$\dot{\mathbf{T}}_{t_0}(\mathbf{x}_0) \coloneqq \lim_{t_1 \to t_0} \frac{d}{dt_1} \mathbf{T}_{t_0}^{t_1} = -2\mathbf{S}_{\mathbf{D}}(\mathbf{x}_0, t_0), \tag{4.32}$$

where $\mathbf{S_D}$ is the diffusion-weighted Eulerian rate-of-strain tensor, which we define as,

$$\mathbf{S_D} \coloneqq \frac{1}{2}\left([\nabla \mathbf{v}]\mathbf{D} - \dot{\mathbf{D}} + \mathbf{D}[\nabla \mathbf{v}]^\top\right). \tag{4.33}$$

Details of these derivations can be found in Appendix B.1 and Appendix B.2, respectively. The tensor $\mathbf{S_D}$ is symmetric and objective (see Appendix B.3), and encodes information analogous to the standard Eulerian rate-of-strain tensor $\mathbf{S}$, but with weighting by the diffusion structure tensor. As such, it captures both temporal and spatial variations in $\mathbf{D}$.

Since $\mathbf{S_D}$ is real and symmetric, it has only real eigenvalues $\mathscr{s}_i$ and corresponding orthonormal eigenvectors $\boldsymbol{e}_i$ with $i \in \{1, 2, ..., n\}$, satisfying,

$$\mathscr{s}_1 \geq ... \geq \mathscr{s}_n \text{ and,} \tag{4.34}$$

$$\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle = \delta_{ij} \tag{4.35}$$

We note that, for an incompressible flow, the standard Eulerian rate-of-strain tensor has zero trace as a consequence of the zero divergence condition—implying, for example in 2D, that its eigenvalues are equal in magnitude and opposite in sign. This property does not generally hold for $\mathbf{S_D}$.

We now take a time derivative of the normalized transport functional and examine the limit as the integration time approaches zero,

$$\lim_{t_1 \to t_0} \frac{d}{dt_1} \mathcal{T}_{t_0}^{t_1}(\mathcal{M}_0) = \frac{\int_{\mathcal{M}_0} \left\langle \mathbf{n}_0, \frac{d}{dt_1}\left(\bar{\mathbf{T}}_{t_0}^{t_1}\right)\big|_{t_1=t_0} \mathbf{n}_0 \right\rangle dA_0}{\int_{\mathcal{M}_0} dA_0} = \frac{-2\int_{\mathcal{M}_0} \langle \mathbf{n}_0, \mathbf{S_D}\mathbf{n}_0 \rangle \, dA_0}{\int_{\mathcal{M}_0} dA_0}, \tag{4.36}$$

which leads to the definition of the *diffusive flux-rate functional* (or *diffusive transport-rate*

functional),

$$\dot{\mathcal{T}}_{t_0}(\mathcal{M}_0) \coloneqq \frac{\int_{\mathcal{M}_0} \langle \mathbf{n}_0, -\mathbf{S_D}\mathbf{n}_0 \rangle \, dA_0}{\int_{\mathcal{M}_0} dA_0}, \tag{4.37}$$

where we have used the previously stated results (4.31) and (4.32), and divided out the factor of 2, which has no effect on the stationary curves of the functional. This functional provides a universal, objective measure of the *averaged diffusive flux rate* over a surface $\mathcal{M}_0$ at time t_0. Objectivity is inherited from the tensor $\mathbf{S_D}$.

This variational problem closely parallels those considered in [61, 68, 141], and we adopt a similar approach to solving it. However, the general-dimensional proof strategy in [68] is not directly applicable here, as it relies on the symmetry and positive definiteness of the transport tensor $\mathbf{T}^t_{t_0}$, which ensures the existence of a unique, real-valued symmetric matrix square root. In contrast, $\mathbf{S_D}$ is real and symmetric, but not necessarily positive definite.

We therefore focus on the two-dimensional case and exploit the reparameterization invariance of the functional, allowing us to proceed with a normal-based formulation. We also draw on the results from [61, 68], which show that stationary curves of similar functionals coincide with null-geodesics of an associated Lorentzian metric. Our structures are defined based on this correspondence, and we refer the reader to the those works for additional detail.

To find extremizers of $\dot{\mathcal{T}}_{t_0}(\mathcal{M}_0)$, we seek surfaces for which the first variation vanishes,

$$\delta \dot{\mathcal{T}}_{t_0}(\mathcal{M}_0) = 0. \tag{4.38}$$

Since the diffusive flux-rate functional is a quotient functional, this extremum problem is equivalent to that of extremizing the following energy functional [23],

$$\delta \dot{\mathcal{E}}_{\dot{\mathcal{T}}_0} = 0, \quad \dot{\mathcal{E}}_{\dot{\mathcal{T}}_0} \coloneqq \int_{\mathcal{M}_0} \Big[\langle \mathbf{n}_0, -\mathbf{S_D}\mathbf{n}_0 \rangle + \dot{\mathcal{T}}_0 \Big] \, dA_0, \tag{4.39}$$

where $\dot{\mathcal{T}}_0$ is the (constant) value of the functional (see Eq. (4.37)) for the minimizing solution $\mathcal{M}_0^*$,

$$\dot{\mathcal{T}}_0 = -\left(\frac{\int_{\mathcal{M}_0^*} \langle \mathbf{n}_0, -\mathbf{S_D}\mathbf{n}_0 \rangle \, dA_0}{\int_{\mathcal{M}_0^*} dA_0}\right). \tag{4.40}$$

The functional $\dot{\mathcal{E}}_{\dot{\mathcal{T}}_0}$ can be interpreted as a *diffusive flux-rate energy functional*, and its stationary curves coincide with those of $\dot{\mathcal{T}}_{t_0}(\mathcal{M})$. We may rewrite this as,

$$\dot{\mathcal{E}}_{\dot{\mathcal{T}}_0} = \int_{\mathcal{M}_0} \left\langle \mathbf{n}_0, \mathbf{S}_{\dot{\mathcal{T}}_0}\mathbf{n}_0 \right\rangle dA_0, \tag{4.41}$$

where $\mathbf{S}_{\dot{\mathcal{T}}_0} = -\mathbf{S_D} + \dot{\mathcal{T}}_0\mathbf{I}$. The scalar quantity $\left\langle \mathbf{n}_0, \mathbf{S}_{\dot{\mathcal{T}}_0}\mathbf{n}_0 \right\rangle$ defines a Lorentzian metric, and the stationary curves of the diffusive flux-rate functional correspond to null-surfaces of this metric.

In two dimensions, these curves are closed orbits (limit cycles) of the following differential equations,

$$\mathbf{r}'(s) = \boldsymbol{\chi}^{\pm}_{\dot{\mathcal{T}}_0}(\mathbf{r}(s)), \quad \boldsymbol{\chi}^{\pm}_{\dot{\mathcal{T}}_0}(\mathbf{r}(s)) = \sqrt{\frac{\mathcal{J}_1 - \dot{\mathcal{T}}_0}{\mathcal{J}_1 - \mathcal{J}_2}}\boldsymbol{e}_1(\mathbf{r}(s)) \pm \sqrt{\frac{\dot{\mathcal{T}}_0 - \mathcal{J}_2}{\mathcal{J}_1 - \mathcal{J}_2}}\boldsymbol{e}_2(\mathbf{r}(s)), \tag{4.42}$$

defined on the domain,

$$U_{\dot{\mathcal{T}}_0} = \{\mathbf{x}_0 \in U : \mathcal{J}_1 \neq \mathcal{J}_2, \ \mathcal{J}_2 < \dot{\mathcal{T}}_0 < \mathcal{J}_1\}. \tag{4.43}$$

See Appendix B.4 for more details.

4.3.3 Compatibility and parallels with Elliptic OECS

Haller and coauthors observed that, when $\mathbf{D} = \mathbf{I}$, elliptic OECS coincide with diffusive flux-rate barriers. Our results are consistent with this observation: in the case $\mathbf{D} = \mathbf{I}$, we recover $\mathbf{S_D} = \mathbf{S}$, the standard rate-of-strain tensor. Furthermore, many results and computational techniques developed for elliptic OECS apply equally well to diffusive flux-rate barriers.

In particular, the computational methods proposed by Karrasch et al. [81], refined by Karrasch and Schilling [80], and the null-geodesic approach of Serra and Haller [142], are all applicable here, with $-\mathbf{S_D}$ replacing the tensor of interest in those formulations.

Shortly after introducing OECS, Serra and Haller [143] proposed a non-dimensional metric to quantify the expected *persistence* of elliptic OECS as they evolve with the flow. Since elliptic OECS (and by extension, diffusive flux-rate barriers) are not material curves or surfaces, their advection under the flow does not generally coincide with the elliptic OECS computed at a later time. That is, advecting the material curve identified as the elliptic OECS at t_0 for a time Δt typically yields a different curve than the elliptic OECS computed at $t = t_0 + \Delta t$.

By comparing the advected image of the OECS at $t = t_0$ with the one computed at $t = t_0 + \Delta t$, a measure of relative "material leakage" can be obtained. Serra and Haller approximate this quantity instantaneously by computing the *pointwise instantaneous material flux density* through an elliptic OECS $\gamma(t)$. This is derived by comparing the velocity of the material curve with the (apparent) "velocity" of the OECS induced by the eigenvector strain-rate fields.

We do not review the full derivation here, but refer the reader to [143], where they define

the *relative material leakage* as,

$$\Gamma_\gamma(t) \coloneqq \frac{\oint_{\gamma(t)} |\varphi(\mathbf{x}(s,t),t)|ds}{A_{\gamma(t)}} \tag{4.44}$$

where φ represents the velocity difference, $\mathbf{x}(s,t)$ is a parametrization of $\gamma(t)$, and $A_{\gamma(t)}$ is the area enclosed by $\gamma(t)$.

In addition, inspired by objective measures of vorticity [64], they define the *rotational coherence* of an elliptic OECS $\gamma(t)$ as,

$$\omega_\gamma(t) \coloneqq \frac{\left|\int_{A_{\gamma(t)}} \omega(\mathbf{x},t) - \overline{\omega}(t)dA\right|}{A_{\gamma(t)}}, \tag{4.45}$$

where ω is the scalar vorticity field and $\overline{\omega}(t)$ is its spatial average over $A_{\gamma(t)}$.

Using these two quantities, they define the *persistence metric* of an elliptic OECS $\gamma(t)$,

$$\Theta_\gamma(t) \coloneqq \frac{\omega_\gamma(t)}{\Gamma_\gamma(t)} = \frac{\left|\int_{A_{\gamma(t)}} \omega(\mathbf{x},t) - \overline{\omega}(t)dA\right|}{\oint_{\gamma(t)} |\varphi(\mathbf{x}(s,t),t)|ds}. \tag{4.46}$$

Selecting the candidate elliptic OECS that maximizes $\Theta_\gamma(t)$ yields the optimal structure for identifying barriers surrounding long-lived Lagrangian vortices.

We note that this metric is also appropriate for diffusive flux-rate barriers, since their derivation assumes the advection-dominated regime. In future work, one could define diffusion-weighted analogues of rotational coherence and material leakage to construct a similar persistence metric for the diffusive case. However, we leave this as an open direction beyond the scope of the present paper.

4.4 Numerical Example

To highlight these structures we look at an ocean flow example and create a contrived diffusion structure tensor to observe a clear difference between these diffusion weighted structures and those that are obtained from the isotropic diffusion case (these are equivalent to advective elliptic OECS). In most fluid flow problems, diffusion will be isotropic and therefore the advective elliptic OECS will be the correct structures to look at. It is only in atypical cases that the diffusion structure will take on a more complicated form (flow through porous media, diffusion through tissue structure, etc.). We simply demonstrate an application of the theory here without a particular real-word diffusion structure example in mind.

4.4.1 Agulhas Leakage

As is commonly done in the coherent structure literature focusing on elliptic structures we look at the Agulhas Rings – anticyclonic eddies generated by the Agulhas current, a current which flows along the southeastern coast of Africa carrying very warm salt water. When the current flows past the tip of Africa, it approaches the roaring forties and most of the contents carried by this current double back to the east (this process is called "retroflection"). Some of the warm salt water "leaks" out in the form of long-lived vortices generated by retroflection which carry this warm salt water north and west through the Atlantic. These eddies can persist for months as they travel through the Atlantic, all the while keeping the warm salt water they carry in their cores isolated from the surrounding flow. We use the geostrophic assumption and derive velocities from the sea surface height (SSH) by treating the SSH h as the streamfunction,

$$\dot{\phi}(\phi,\theta,t) = -\frac{g}{R_e^2 f_c(\theta)\cos(\theta)}\frac{\partial h(\phi,\theta,t)}{\partial \theta}, \quad \dot{\theta}(\phi,\theta,t) = \frac{g}{R_e^2 f_c(\theta)\cos(\theta)}\frac{\partial h(\phi,\theta,t)}{\partial \phi} \tag{4.47}$$

where $f_c(\theta) = 2\Omega\sin(\theta)$ represents the Coriolis force with Ω representing the angular velocity of the Earth. R_e represents the average radius of the Earth.

We use AVISO data provided by the `TBarrier` package [8] for a direct comparison to the original diffusive barrier paper [68]. This dataset comes from an earlier AVISO product (no longer available). For up-to-date versions of these datasets, see https://www.aviso.altimetry.fr.

4.4.2 Diffusion Structure

Due to the construction of these structures (being defined in terms of diffusive transport while being in the advection dominated regime), it will only be in exceptional cases that complicated diffusion structure actually creates transport barriers. The more common scenario would be that the diffusion structure destroys transport barriers. We focus on this case and define a diffusion structure tensor that will destroy transport barriers. We do this by first identifying these barriers and then creating extreme diffusion structure where one of these barriers is initially present. The extreme diffusion structure we use is constructed out of a radial super Gaussian. This a generalization of a standard 2D Gaussian. Recall the 2D gaussian is given by,

$$f(x,y) = A\exp\left(-\left(\frac{(x-x_0)^2}{2\sigma_X^2} + \frac{(y-y_0)^2}{2\sigma_Y^2}\right)\right) \tag{4.48}$$

where A denotes the maximum magnitude, (x_0, y_0) denotes the center and σ_X and σ_Y de-

termine the x and y spread of the Gaussian respectively. Assuming $\sigma = \sigma_X = \sigma_Y$, the super Gaussian is a generalization given by,

$$f_S(x, y; n) = A \exp\left(-\frac{|x - x_0|^n + |y - y_0|^n}{\sigma^n}\right) \tag{4.49}$$

As n gets larger, the peak spreads out more and the drop off becomes steeper, though the shape tends towards a square rather than a circle. To remedy this, we use a radial super-Gaussian to maintain the behavior but keep the circular shape. This is given by,

$$f_{S_r}(x, y; n) = A \exp\left(-\frac{((x - x_0)^2 + (y - y_0)^2)^{n/2}}{\sigma^n}\right) \tag{4.50}$$

As mentioned, we identify a strong transport barrier at the initial time and then we apply this radial super-Gaussian in region which the transport barrier encapsulates to create highly anisotropic diffusion structure in this region. For our case, $\mathbf{x}_0^* = (x_0, y_0) = (2.8, -32.0)$, $A = 100$, $\sigma = 1.5$, and $n = 8$, resulting in the following function,

$$f_{S_r}(\mathbf{x}) = 100 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0^*\|^4}{1.5^8}\right) \tag{4.51}$$

where $\|\cdot\|$ represents the standard 2-norm. Let

$$C_r(\mathbf{x}) = 1 + f_{S_r}(\mathbf{x}) = 1 + 100 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0^*\|^4}{1.5^8}\right) \tag{4.52}$$

We then construct the diffusion tensor in the following manner,

$$\mathbf{D}(\mathbf{x}) = \begin{bmatrix} \frac{1}{C_r(\mathbf{x})} & 0 \\ 0 & C_r(\mathbf{x}) \end{bmatrix} \tag{4.53}$$

This leads to diffusion structure that is highly anisotropic (with extreme y-direction preference) within the transport barrier, and isotropic everywhere else in the domain.

4.4.3 Transport and Transport-Rate Barriers Simulation

We now compute transport barriers and transport-rate barriers for both homogeneous isotropic diffusion and for the diffusion structure tensor we created above. We use the null geodesic approach developed by Serra and Haller [142] to compute all coherent structures and finite-time structures are computed using an integration time of $T = 25$ days. Following this, we initialize a concentration of a passive diffusive scalar field that evolves according to the advection diffusion equation. Then, we use FEM to simulate the transport of the concentration on a simple 541×401 rectangular grid with longitude running from $[-7.5°, 6°]$, latitude running from $[-34°, -24°]$, $dt = 0.2$ days, and $Pe = \mathcal{O}(10^4)$. We solve the advection diffusion equation using an implicit midpoint Crank-Nicolson method and, considering we are in the advection dominated regime, we add streamline upwind Petrov-Galerkin (SUPG) stabilization to minimize spurious oscillations. All FEM simulations are performed using the `FEniCS` Python package [2, 86, 87, 88, 96, 97, 98, 99, 115], an open-source package which is essentially a wrapper for the C++ backend `DOLFIN`[1]. Initial concentrations are defined by using an indicator function which is 1 inside a coherent set and 0 outside, followed by the application of Gaussian filter which replaces infinite gradients at coherent set boundaries with steep gradients, again with the aim of minimizing spurious oscillations. Three additional circular sets are initialized in the same way to highlight the difference in behavior between a concentration inside a coherent set and one outside it. We overlay the elliptic barriers and advect them under the flow map. In addition, we provide the DBS field for

[1]We note that there are new versions of both `FEniCS` and `DOLFIN` called `FEniCSx` and `DOLFINx` that improve on the previous packages [2, 6, 138, 139].

completeness. All advective particle integration is performed by the current public version of `NumbaCS` [74] and all coherent structure methods are performed by a development version that will be included in the next public release. For all figures, a link to the accompanying video is provided in the caption.

Isotropic diffusion barriers

We begin by computing elliptic diffusive transport barriers at the initial time when diffusion is isotropic ($\mathbf{D} = \mathbf{I}$) and initialize the concentrations we describe above (see Figure 4.1, left). Notice all the diffusive barriers act as they should and largely contain the diffusive substance in a coherent manner while the other initialized concentrations show more complicated behavior and mix with the rest of the flow. We note that there is a strong diffusive barrier with center at roughly $\mathbf{x}_0^* = (2.8, -32)$.

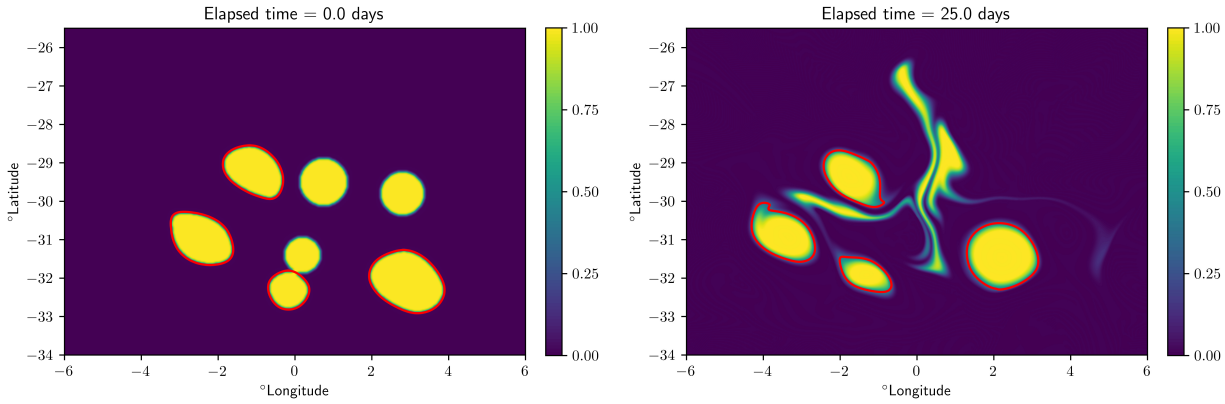

Figure 4.1: Left: Lagrangian elliptic diffusion barriers overlaid on initial concentration at $t_0 = 11/24/2006$. Right: Advected image of elliptic diffusion barriers and evolved concentration at $t = t_0 + 25$ days. See video.

The corresponding DBS fields can be seen in Figure 4.2. To see the DBS field overlaid on the diffusive simulation, see the following video.

In Figure 4.3, we see the initial Eulerian elliptic diffusion barriers at the initial time and the

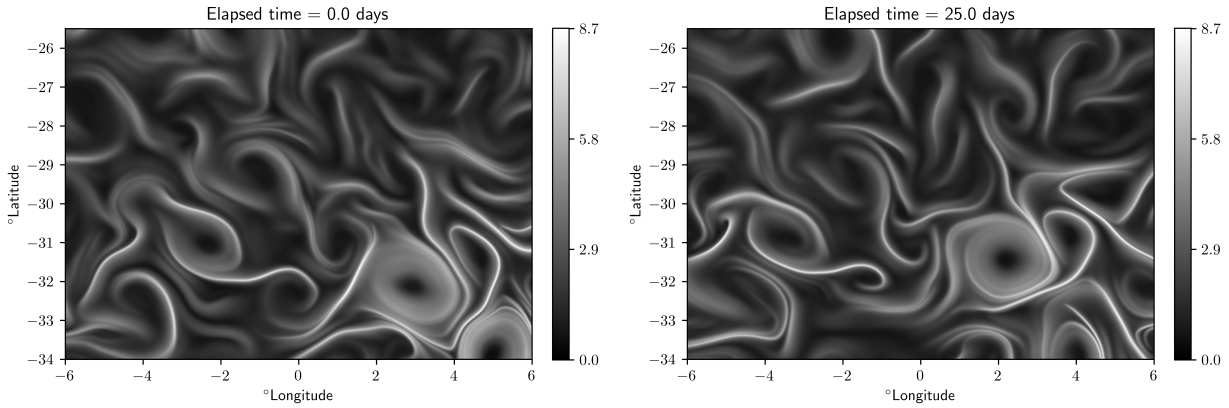

Figure 4.2: Left: DBS field at $t_0 = 11/24/2006$. Right: DBS field at $t = t_0 + 25$ days. See video.

images of under the flow for a period of 10 days. We note that these are barriers derived from a single velocity snapshot so the fact that they appear less coherent than their Lagrangian counterparts is not surprising and it is actually quite impressive that they at least identify similar structures and remain somewhat coherent for as long as they do. If we employed the persistence metric put forth by Serra and Haller [143] and briefly described in Sect. 4.3.3, we would identify structures that remained even more coherent for longer and, if we used a threshold of this metric, structures that quickly lose their coherence could be filtered out (like the one centered at $(2.5, -29.3)$). The strong Lagrangian barrier centered at $\mathbf{x}_0^*$ is also roughly identified.

Inhomogeneous, anisotropic diffusion barriers

Next, both Lagrangian and Eulerian diffusive barriers are computed for the diffusion structure tensor from Eq. (4.53). We first show how the Lagrangian and Eulerian barriers computed for isotropic diffusion will incorrectly identify a diffusive barrier with center at $\mathbf{x}_0^*$. While the Lagrangian theory can account for this by simply using the correct diffusion structure tensor in the calculation, there is no equivalent Eulerian calculation that accounts

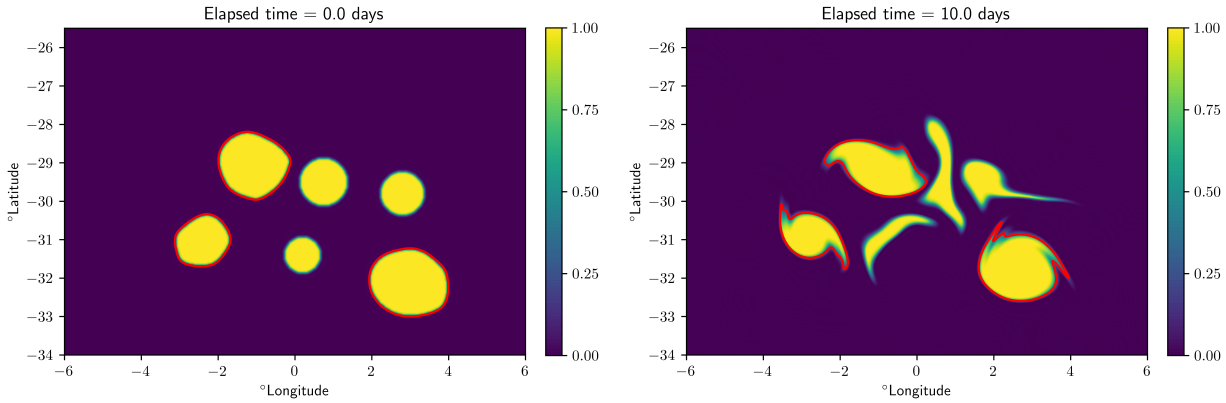

Figure 4.3: Left: Eulerian elliptic diffusion barriers overlaid on initial concentration at $t_0 = 11/24/2006$. Right: Advected image of elliptic diffusion barriers and evolved concentration at $t = t_0 + 10$ days. See video.

for this complicated diffusion structure. The diffusive flux-rate barriers for general diffusion structure we have presented in this work fill this need. In Figure 4.4, we again see the same initialization on the left. In the same figure on the right, we see that the strong coherence of the barrier with center at $\mathbf{x}_0^*$ no longer persists, and the concentration within that barrier diffuses quite dramatically through the barrier.

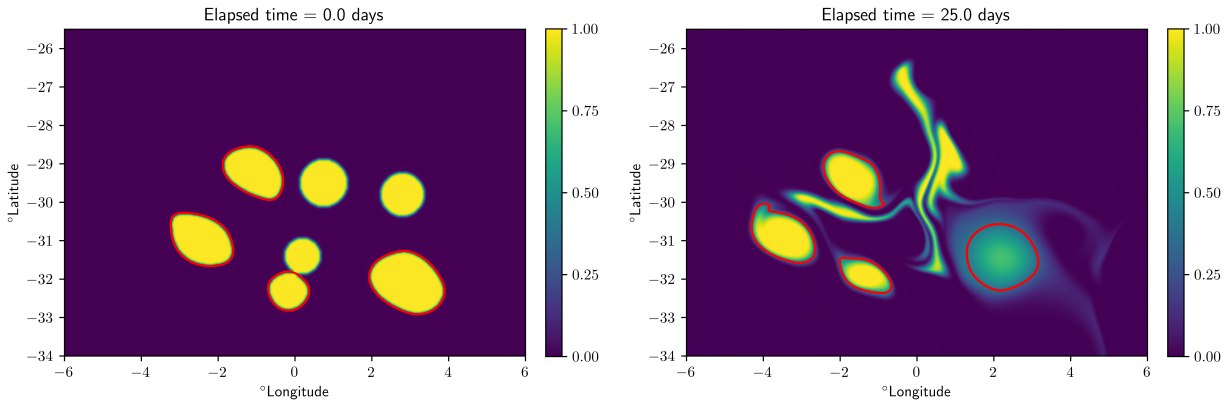

Figure 4.4: Left: Lagrangian elliptic diffusion barriers computed from isotropic diffusion overlaid on initial concentration at $t_0 = 11/24/2006$. Right: Advected image of elliptic diffusion barriers and evolved concentration using the more complicated diffusion structure tensor at $t = t_0 + 25$ days. See video.

The corresponding DBS fields can be seen in Figure 4.5. Notice the high DBS values near the

highly anisotropic diffusion region. To see the DBS field overlaid on the diffusive simulation, see the following video.

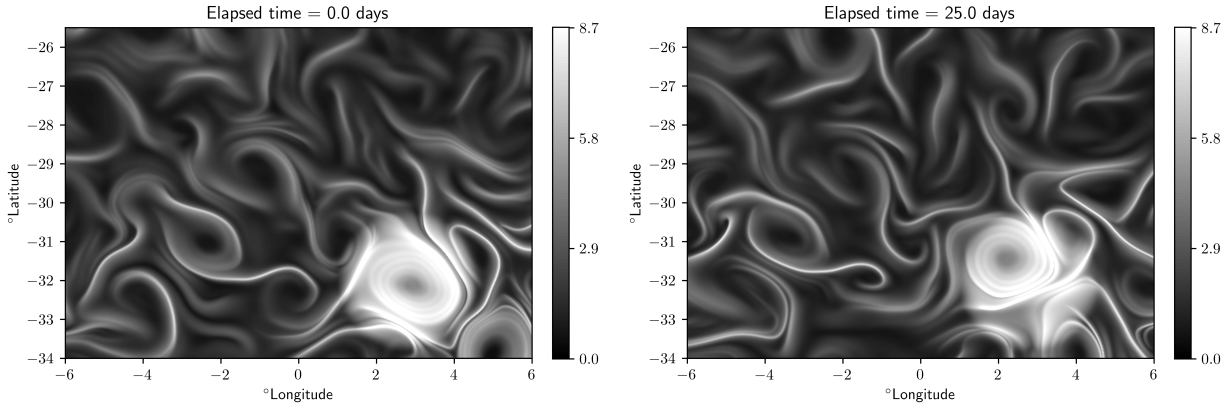

Figure 4.5: Left: DBS field at $t_0 = 11/24/2006$. Right: DBS field at $t = t_0 + 25$ days. See video.

We perform the same simulation with the Eulerian barriers and the same issue persists, a structure is incorrectly identified with center at $\mathbf{x}_0^*$. This can be seen in Figure 4.6.

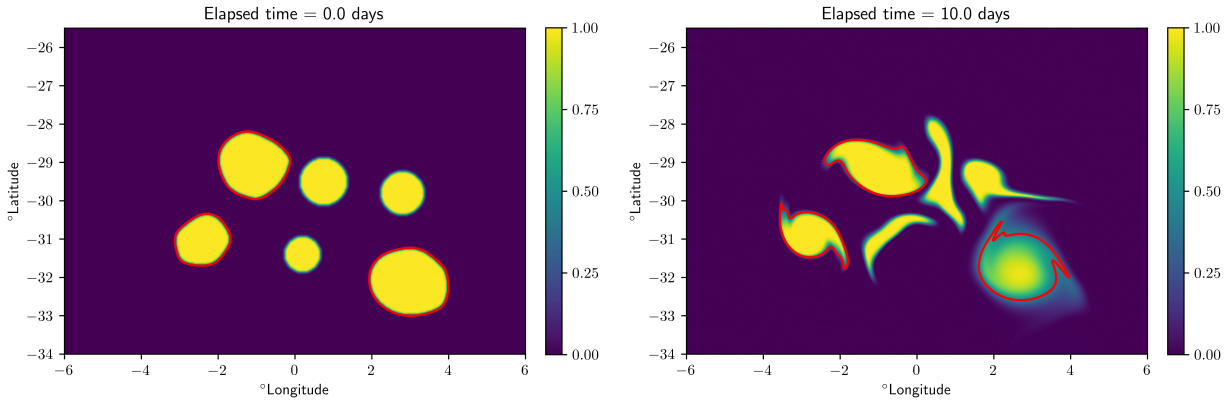

Figure 4.6: Left: Eulerian elliptic diffusion barriers computed from isotropic diffusion overlaid on initial concentration at $t_0 = 11/24/2006$. Right: Advected image of elliptic diffusion barriers and evolved concentration using the more complicated diffusion structure tensor at $t = t_0 + 10$ days. See video.

Finally, we show the diffusive barriers computed using the complicated diffusion structure tensor. The Lagrangian barriers can be seen in Figure 4.7 and the Eulerian barriers in Figure

4.8.

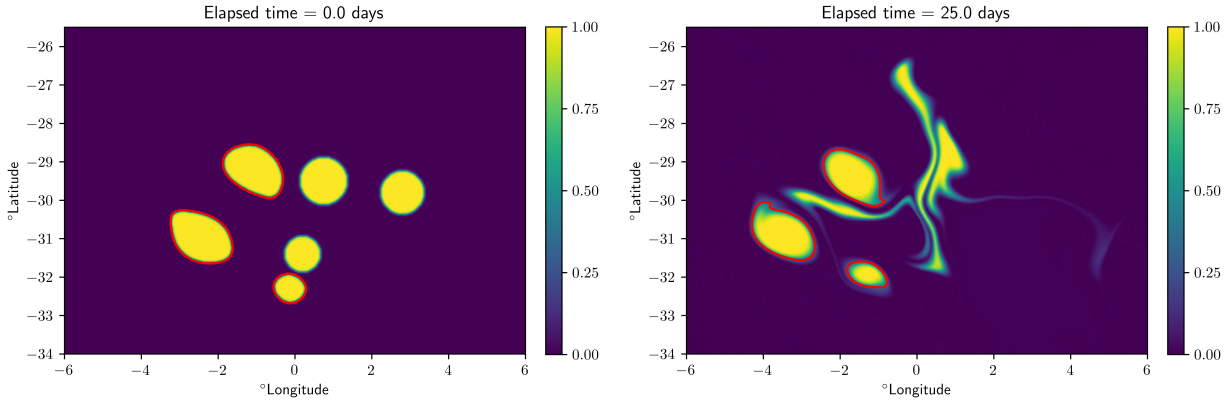

Figure 4.7: Left: Lagrangian elliptic diffusion barriers computed from the complicated diffusion structure overlaid on initial concentration at $t_0 = 11/24/2006$. Right: Advected image of elliptic diffusion barriers and evolved concentration using the more complicated diffusion structure tensor at $t = t_0 + 25$ days. See video.

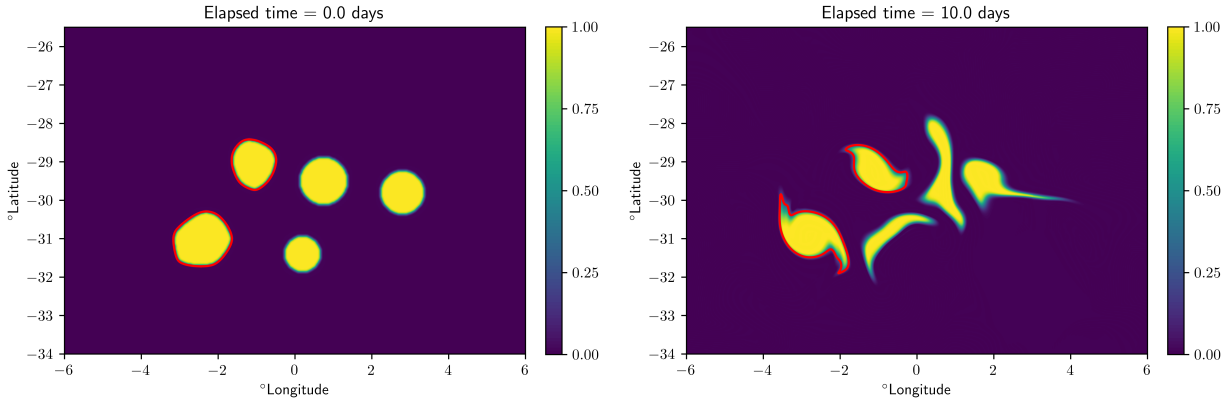

Figure 4.8: Left: Eulerian elliptic diffusion barriers computed from the complicated diffusion structure overlaid on initial concentration at $t_0 = 11/24/2006$. Right: Advected image of elliptic diffusion barriers and evolved concentration using the more complicated diffusion structure tensor at $t = t_0 + 10$ days. See video.

In both of these cases, when the appropriate diffusion structure tensor is taken into account, both the Lagrangian and Eulerian structures correctly do *not* identify a transport barrier centered at $\mathbf{x}_0^*$ and the sets whose boundaries these methods identify stay largely coherent, as expected.

4.5 Conclusion

We have developed an instantaneous approximation of material diffusive transport barriers by differentiating the appropriate functional and proceeding with the variational problem, much like what was done in the purely advective case. By defining diffusive flux-rate barriers in terms of a diffusive weighted version of the classic Eulerian rate-of-strain tensor, we show that these Eulerian structures can capture complicated spatial and temporal dependencies in the diffusion structure tensor. While true physical application of this theory will most likely be in exceptional circumstances, we have appropriately defined the instantaneous counterparts to finite-time diffusive barriers.

We see a few main directions of work moving forward. First, defining a persistence metric similar to what was done for elliptic OECS (or showing that the elliptic OECS persistence metric is the appropriate measure of persistence for these structures as well) is a worthwhile endeavor. While not explored here, there seems to be a range of Péclet numbers for which this theory (and the finite-time case) identify what would physically be seen as the "correct" structures. In the original derivation of material diffusion barriers, when non-dimensonalizing the functional the diffusivity parameter is divided out and is only assumed to be small such that transport is happening in the advection dominated regime. One could construct a case where the diffusion structure tensor destroys many coherent sets (in both the instantaneous and finite-time case) but the Péclet number is so large that this diffusion structure has little practical effect on the transport of a concentration. From the few numerical experiments we performed, this seems to be more pronounced for the Eulerian structures but does affect the Lagrangian structures as well. Exploring the appropriate range of couplings between diffusion structure and Péclet numbers for both instantaneous and finite-time structures would be a useful line of work. Finally, though we did not delve into it here, this theory

applies equally well to instantaneous barriers to stochastic transport. Rigorously stating this and demonstrating the theory with applications will be a main focus for future work, and we expect this context may give rise to more applications for the general diffusion structure case.

Chapter 5

Accurate and Efficient extraction of LCS from their variational theory using Automatic Differentiation

Attribution

This chapter, a collaborative work with Shane Ross, is a manuscript currently in preparation.

Author Contributions

Jarvis and Ross conceived this project. Jarvis developed the theory, designed the numerical framework, implemented algorithms, optimized code, designed examples, and wrote the original manuscript. Jarvis and Ross contributed to revisions.

Abstract

Lagrangian coherent structures (LCS) computed from their variational theory provide precise material surfaces that are extremizers of the associated functional from which they were

defined, though their computational implementations are challenging, expensive, and prone to errors due to how derivatives of the flow map are computed with respect to initial conditions. An area of research in the overlap of mathematics and computer science called Automatic Differentiation (AD) has seen an explosion in popularity in recent years due its use in training deep neural networks (backpropagation can be viewed as a special case of AD) by providing an accurate (up to machine precision) and efficient way to compute derivatives of functions with respect to function parameters. The applicability of this technique to variational LCS has only very recently been explored for hyperbolic structures in 3 dimensions, though this work has largely gone unnoticed. Here we attempt to highlight the utility of this technique for coherent structures and explore the application to hyperbolic and elliptic coherent structures in 2 dimensions. We also present an easy-to-use Python framework which implements these methods.

5.1 Introduction

The identification and extraction of influential material curves has emerged as a powerful tool for understanding how a flow's contents are organized through the development of Lagrangian coherent structures. These structures were defined as extremizers of functionals relating to the strain experienced by material curves over a given time window. Hyperbolic structures are defined as material curves that extremize the averaged normal repulsion rate and elliptic structures are those which extremize the averaged tangential strain. These structures can be obtained as solution curves in the appropriate direction fields made up of the eigenvectors (scaled in the elliptic case) of the Cauchy Green strain tensor.

While this may seem straightforward at first glance, there are a number of nontrivial challenges associated with this algorithm. We will focus on the one that incurs the highest

computational cost. To compute the Cauchy Green tensor, gradients of the flow map must be computed and this is typically done with finite difference approximations by using neighboring grid points in the divided differences. This is simple and somewhat efficient but is highly prone to error. It was shown that the eigenvalues are robust to grid spacing and in fact, one can argue, benefit from using a larger grid spacing to capture dominant regions in the FTLE field on a discretized grid [92]. On the other hand, the eigenvectors of $\mathbf{C}$ are extremely sensitive to this grid spacing and given that LCS are computed as solution curves in the eigenvector fields of $\mathbf{C}$, it is of paramount importance that these eigenvector directions are accurate [41].

The two options to remedy this are to use an extremely fine grid, which would be prohibitively expensive, or use an auxiliary grid around each grid point to compute the finite differences. The latter is clearly the better approach, but results in a $5\times$ increase (in 2D) in computation time as now, for each grid point, we need to compute 5 particle trajectories instead of 1. In addition, while we are getting a more accurate approximation to the gradient of the flow map, we are still performing a finite difference, with an extremely small spacing (due to the small auxiliary grid spacing) which is prone to round-off errors. An ideal solution would avoid the need for additional particle integration while still providing accurate gradients with respect to initial conditions. At first glance, this might seem like an unrealistic expectation – a case of wanting to have our cake and eat it too.

However, automatic differentiation (also known as algorithmic differentiation) offers an elegant solution to this problem and, thanks to recent advancements in its implementation, can do so in a highly efficient manner. AD is an approach that sits somewhere in between numerical and symbolic differentiation and provides a way to obtain derivatives of functions that are accurate up to machine precision. A specific version of this method (backpropagation) serves as the backbone for training deep neural networks and is essentially the mechanism

by which they learn. While this method has become more mainstream due to the explosion in popularity of deep learning models, the utility of this tool for Lagrangian coherent structure methods has gone largely unnoticed. We note Gilpin presented on this topic in 2021 [51] with a focus on the finite-time Lyapunov exponents, and more recently, Tyler and Wittig [158] developed an improved numerical method for 3D hyperbolic LCS which utilize similar underlying ideas. In this work, we propose these ideas as a foundation to improve all Lagrangian coherent structure methods and, by presenting this work in terms of automatic differentiation and neural ODEs, open up the possibility for a number of off-the-shelf numerical implementations as these tools have seen huge amounts of development in recent years. We also note that this approach can be extended to a number of other coherent structure type methods which rely on accurate linearized flow maps.

5.2 Background

We will begin covering the concept of automatic differentiation by following [10] and [40]. We refer the readers to the many great surveys on the topic (e.g., [7, 10, 28, 40]) for a more detailed treatment. Then, we briefly cover the the recent emergence of neural differential equations. Finally, we review the standard approach for obtaining linearized flow maps used for LCS extraction methods in 2 dimensions and refer the reader to earlier sections in this dissertation for more details (see Ch. 1 and Ch. 2).

5.2.1 Computational Differentiation

A crucial step in many computational methods involves finding derivatives of some function with respect to function parameters or initial conditions. Certain applications require

extreme accuracy while others benefit more from efficient algorithms.

For deep neural networks, the basic idea is to train a model, consisting of many layers, each with many nodes, and with unique weights on each of these nodes, to approximate some complicated underlying process which the data is derived from. This training happens via gradient descent, typically by minimizing some loss (usually just the error between the model output and known target output data). The "learning" happens by finding which combinations of weights minimize the loss. How is this done? By differentiating the loss with respect to all weights of the network (leading to the gradient of the loss with respect to weights), and moving the weights in the opposite direction of the gradient (the gradient will tell us which direction a function is growing the most), and repeating this process until the loss is below some set threshold. In essence, this boils down to computing derivatives of deeply composed functions with respect to possibly millions or billions of weights. In this application, efficiency is key.

The three main methods to compute derivatives in a computational setting would be to use either *symbolic differentiation*, *numerical differentiation*, or *automatic differentiation*. In symbolic differentiation, derivative rules (like power rule, transcendental derivatives, chain rule, product rule, etc.) are essentially hard-coded into a program and these rules are used to compute derivatives of algebraic expressions very similar to how one would compute them by hand. We will largely focus on numerical and automatic differentiation as these are the most relevant to our application but we note that while symbolic differentiation will yield accurate derivatives, the function needs to be represented by closed-form expressions. In addition, this method can suffer from a problem known as expression swell (see [10] for more details).

Numerical Differentiation

Numerical differentiation simply involves using a finite difference approximation of a the derivative. Say we have a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ and we would like to compute the derivative of this function with respect to the vector $\mathbf{x}$. This would result in the Jacobian matrix,

$$\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]_{ij} = \frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\mathbf{x} + h\mathbf{e}_j) - f_i(\mathbf{x} - h\mathbf{e}_j)}{2h} \tag{5.1}$$

which, since we are using a centered difference, will have truncation error $\mathcal{O}(h^2)$ and would perform $\mathcal{O}(nm)$ evaluations. So this is rather expensive for the above application but it seems the error could essentially be driven to zero by making h smaller and smaller. Unfortunately, this is not the case and this is due to finite differencing being an inherently ill-conditioned and unstable numerical procedure. While the truncation error for finite differencing shrinks as $h \to 0$, the round-off error grows due to limited machine precision leading to loss of significance. This happens because we are subtracting two numbers the are nearly identical (numerator of Eq. (5.1)) and dividing this by a number that is near zero (h). When we subtract two numbers that are nearly identical using a computer, the leading significant bits representing these numbers cancel out and the result is represented by lower order, less significant bits, leading to round-off error. Then, when we divide by a number near zero, this error is amplified.

We look at a simple example to illustrate this. Let $f(x) = \sin(x)$, then $f'(x) = \cos(x)$. If we want the derivative at $x = 1$, we would just plug 1 into the derivative, leading to the value $f'(1) = \cos(1) = 0.5403023058681398$, which is exact (up to machine precision). Now, pretending we do not actually have access to the derivative function we can apply Eq. (5.1) for varying h values. In Figure 5.1, we can see that the truncation error shrinks as h gets smaller but beyond some optimal value for h, the error actually gets worse and when

h is near machine precision (for double precision) the error becomes worse than the largest spacing we used! Clearly this procedure depends on a parameter h and if this parameter is not within some optimal window, the error can quickly get outside of an acceptable range, requiring recomputation with a different h.

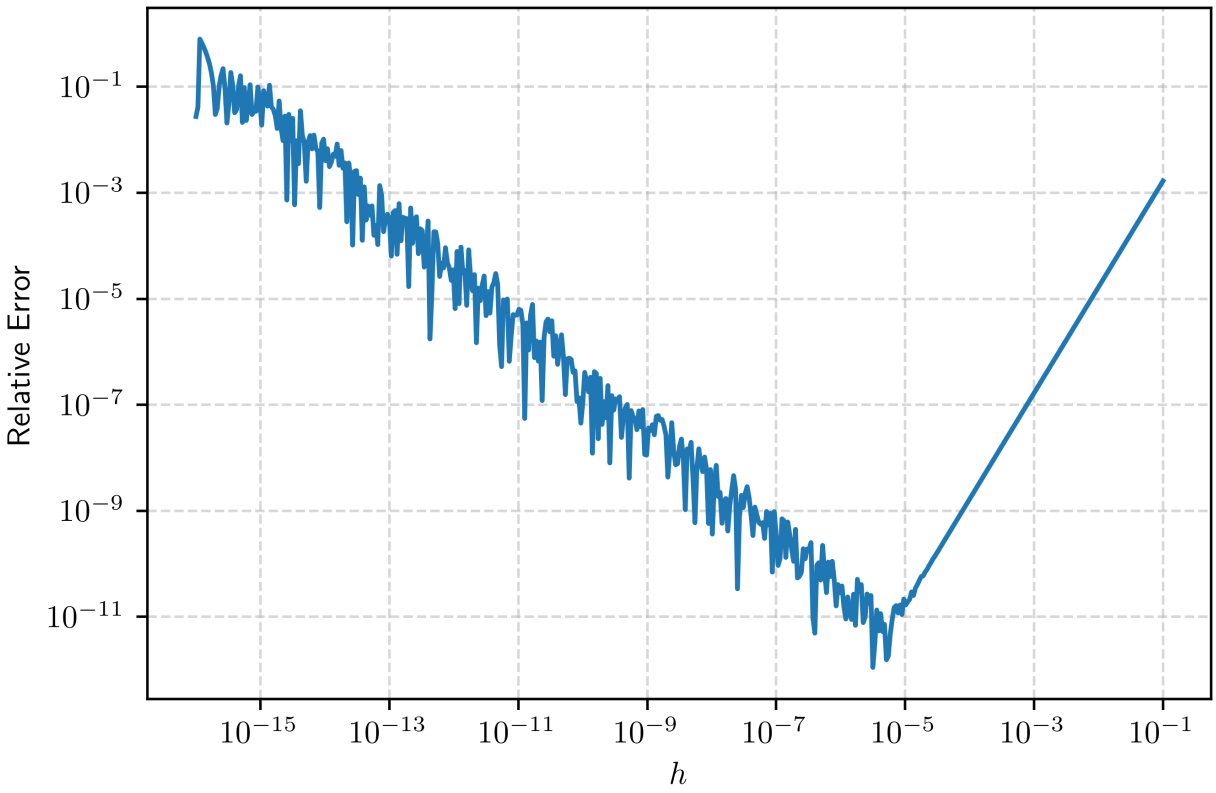

Figure 5.1: Finite difference spacing vs. relative error for finite difference approximation of derivative of $f(x) = \sin(x)$ at $x = 1$.

Automatic Differentiation

Automatic differentiation can be viewed as a hybrid of symbolic and numerical differentiation. In this approach, the function f is treated as a computer program involving a series of elementary operations (sometimes called *primitives*) for which the derivatives of each of these operations is known (addition, multiplication, trigonometric or exponential operations, etc.).

Evaluating a function this way is often referred to as the *evaluation trace* or *primal trace* (see Figure 5.2, left). A computational graph is also generated from the evaluation trace to identify dependencies between the elements of the evaluation trace (see Figure 5.2, right). Using this evaluation trace, we can compute the derivative of the function with respect to some parameter by tracking the derivative of each of these composed elementary operations and combining them through the chain rule to produce the final derivative we seek. This essentially amounts to performing symbolic differentiation on the the elementary operations and keeping track of the numerical value, without the need to compute or store the entire symbolic derivative. This can result in efficient computations of derivatives while maintaining perfect accuracy (up to machine precision). There are two modes of AD, forward mode and reverse mode. We cover forward mode AD first and in more detail as this mode is simpler to understand. We refer the reader to any of the cited surveys for a more detailed treatment. We will follow the example covered in [10] and [40] with $f(x_1, x_2) = \log(x_1) + x_1 x_2 - \sin(x_2)$ and seek this function's derivative with respect to x_1 at $(x_1, x_2) = (2, 5)$.

x_1		$= 2$
x_2		$= 5$
v_1	$= \log x_1$	$= \log 2$
v_2	$= x_1 \times x_2$	$= 2 \times 5$
v_3	$= \sin x_2$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

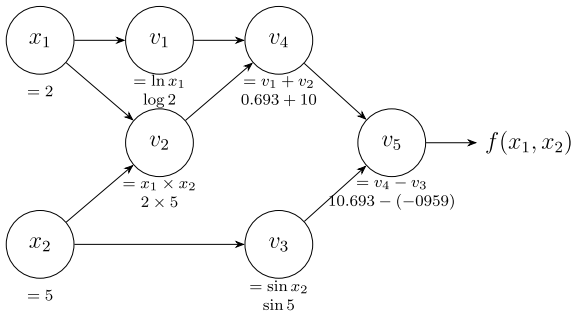

Figure 5.2: Left: Primal trace. Right: Corresponding computational graph. Source: Fang et al. [40], licensed under CC BY 4.0.

In forward mode AD, one computes the evaluation trace of f as in Figure 5.3, left. To do this, let x_1 and x_2 be themselves and let v_i represent intermediate variables (sometimes called *primals*) that when composed, result in the final function value. Then, in tangent, one computes the derivatives of the intermediate variables v_i, generating the tangent trace

and keeping track of the numerical values. This can be viewed as propagating derivatives through the computational graph through a Jacobian-vector product. Say we are interested in computing the derivative of the final value $y = f(x_1, x_2)$ with respect to x_1. For each v_i, let $\dot{v}_i = \dfrac{\partial v_i}{\partial x_1}$ and then $(\dot{x}_1, \dot{x}_2) = (1, 0)$ is the vector we propagate through the computational graph. As we move through the evaluation trace on the left, we compute derivatives of the primals on right by using symbolic differentiation (see Figure 5.3). When we make our way through the primal and tangent traces, we arrive at $y = v_5 = 11.652$ on the left, the function value $f(2, 5)$, and the corresponding derivative on the right, $\dot{y} = \dot{v}_5 = 5.5$, which is the derivative of the function f with respect to x_1, evaluated at $(2, 5)$. Since f is a simple function, we can compute its derivative by hand and get $\dfrac{\partial f}{\partial x_1} = \dfrac{1}{x_1} + x_2$. When we evaluate this at $(2, 5)$, we get $f_{x_1}(2, 5) = 5.5$. We now have the derivative exactly (up to machine precision). Conversely, if we compute the derivative using finite differences, we get error behavior very similar to Figure 5.1. Forward mode requires a tangent trace for each input variable/parameter.

x_1		$= 2$
x_2		$= 5$
v_1	$= \log x_1$	$= \log 2$
v_2	$= x_1 \times x_2$	$= 2 \times 5$
v_3	$= \sin x_2$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

$\dot{x}_1$	$= \partial x_1 / \partial x_1$	$= 1$
$\dot{x}_2$	$= \partial x_2 / \partial x_1$	$= 0$
$\dot{v}_1$	$= \dot{x}_1 / x_1$	$= 1/2$
$\dot{v}_2$	$= \dot{x}_1 \times x_2 + \dot{x}_2 \times x_1$	$= 1 \times 5 + 0 \times 2$
$\dot{v}_3$	$= \dot{x}_2 \times \cos x_2$	$= 0 \times \cos 5$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
$\dot{y}$	$= \dot{v}_5$	$= 5.5$

Figure 5.3: Left: Primal trace. Right: Corresponding tangent trace. Source: Fang et al. [40], licensed under CC BY 4.0.

For reverse-mode, we first compute the evaluation trace forward using a forward pass. Then, we perform a backward pass to compute the derivative. This pass is often referred to as the *adjoint trace*. Let $\bar{v}_i = \dfrac{\partial y}{\partial v_i}$. We set $\bar{v}_5 = \bar{y} = 1$ and perform the reverse adjoint trace using dependencies from the computational graph to arrive at the derivative of y with respect to

both x_1 an x_2 (see Figure 5.4) that is accurate up to machine precision. Backpropagation can be viewed as a special case of reverse-mode AD. Reverse mode requires an adjoint trace for each output variable.

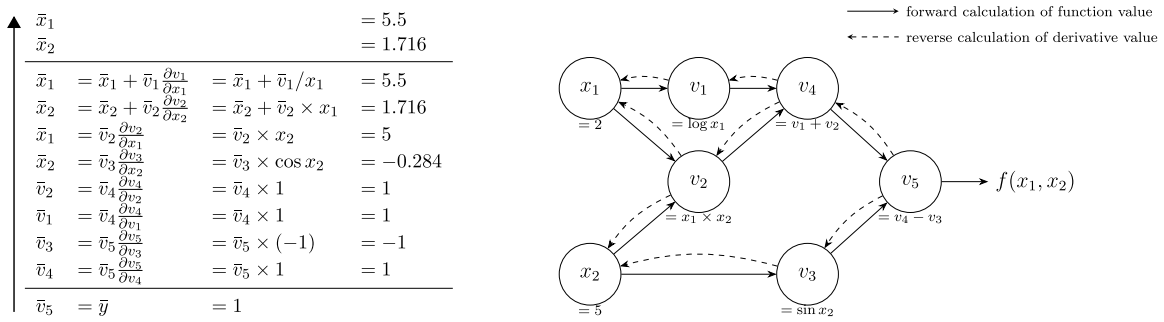

Figure 5.4: Left: Adjoint trace. Right: Corresponding computational graph. Source: Fang et al. [40], licensed under CC BY 4.0.

If our function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, forward mode will be computationally advantageous when $n << m$ and reverse mode will be more efficient when $n >> m$. Since f will usually represent a loss (a scalar value) in the context of neural networks and the input space will consist of millions or billions of weights (sometimes even trillions for LLMs), reverse mode is the go-to mode for most machine learning applications.

5.2.2 Neural Ordinary Differential Equations

A very recent field coming out of the deep learning boom is the field of neural differential equations (NDEs). These can be viewed as differential equations in which the vector field is parameterized by a neural network. These equations arise naturally when looking at the dynamics of hidden states in certain types of neural networks and observing what happens when one looks at the limit of the hidden state dynamics as layer depth goes to infinity. We will only very briefly cover the basics of this work as we are really only interested in a viewpoint this theory provides and tools that arose from this theory. For the interested

reader, we suggest the seminal paper by Chen et al. [26] and the wonderful dissertation of Kidger [85] for more details.

Let's say we have input data $\mathbf{x} \in \mathbb{R}^n$ and corresponding output data $\mathbf{y} \in \mathbb{R}^m$ and we want to train a network $\mathbf{f}$ that approximates the underlying process that produced $\mathbf{y}$ from $\mathbf{x}$ in an optimal way (we ignore possible networks for initial and final layers for simplicity). In certain types of neural networks (RNNs, ResNet, normalizing flows, etc.) the dynamics of hidden states $\mathbf{h}_t$ can be represented by a difference equation. For residual networks [70], this looks like,

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \mathbf{f}(\mathbf{h}_t, \theta_t) \tag{5.2}$$

where θ_t denotes the parameters for layer t, with $t \in \{0, ..., T\}$ (discrete). By passing data through the network and tuning the θ_t's via a form of backpropagation, one can obtain a more accurate $\mathbf{f}$ that maps the initial data $\mathbf{h}_0 = \mathbf{x}$ to target data $\mathbf{h}_T$ such that the loss with respect to $\mathbf{y}$ is minimal. This can be viewed as an Euler step (with $\Delta t = 1$) corresponding to the following differential equation

$$\frac{d\mathbf{h}(t)}{dt} = \mathbf{f}(\mathbf{h}(t), t, \theta) \tag{5.3}$$

where now $t \in [0, T]$ (continuous). This correspondence with continuous system allows the use of more sophisticated ODE solvers that can potentially speed up the training process (by taking larger steps Δt where the dynamics are not so complicated). In this formulation, $\mathbf{f}$ can be thought of as a parameterized vector field that trajectories $\mathbf{h}(t)$ evolve in. By tuning the vector field, one can better model the underlying governing process that generates $\mathbf{y}$ from $\mathbf{x}$ by minimizing the loss between $\mathbf{h}(T)$ and $\mathbf{y}$. In addition, the continuous nature

of this formulation can generally allow $\mathbf{f}$ to be a simpler network, with fewer parameters compared to a standard residual network, for example. Consider an ODE with a relatively simple RHS which, when integrated, can result in complex, highly nonlinear behavior. This roughly captures the idea of how a neural ODE can capture complicated behavior with a simpler network. Now the steps of the solver can be seen as the evaluation trace and one can apply AD to this computational graph to obtain accurate derivatives with respect to parameters[1].

This is the key takeaway for us: automatic differentiation can be performed on numerical solutions to differential equations to obtain accurate derivatives of the solution (or some function of the solution) of the differential equation with respect to some parameters. In the neural ODE world, they are typically interested in derivatives of the loss with respect to the network parameters, though one could obtain derivatives of the final state with respect to the initial state as well. We focus on this viewpoint.

To make this more explicit, we can view numerical solutions of differential equations as composed function evaluations of the initial conditions. For example, if we have the system

$$\frac{dx(t)}{dt} = f(x(t), t) \tag{5.4}$$

and wish to solve this numerically using an Euler scheme,

$$x_{i+1} = x_i + hf(x_i, t_i) \tag{5.5}$$

[1]In Chen et al. [26], the authors suggest another method to do this called the *adjoint sensitivity method* that is more memory efficient. The `torchdiffeq` package implements this method by default. Conversely, Kidger [85] advocates for directly applying reverse-mode AD to the solver trace using checkpointing for better memory efficiency. This is the default in the `Diffrax` package.

we could write the final position as at $t_f = t_{n+1}$,

$$x(t_{n+1}) = x_0 + \sum_{i=0}^{n} hf(x_i, t_i) \tag{5.6}$$

We can rewrite this in terms of the initial condition for clarity,

$$x(t_f) = x_0 + hf(x_0, t_0) + hf(x_0 + hf(x_0, t_0), t_1) + hf(x_0 + hf(x_0 + hf(x_0, t_0), t_1), t_2) + \dots \tag{5.7}$$

Now we have a representation of $x(t)$ which depends on the initial condition x_0, though it is in terms of deep function compositions which only become more complicated when using more sophisticated solvers. Taking derivatives w.r.t. x_0 by hand or symbolically becomes very messy (especially as n increases). AD allows us to solve this simply, without the need to rewrite the solution in terms of initial conditions or compute any derivatives by hand by differentiating the steps of the solver. Simply solve the differential equation and apply the desired AD method to obtain accurate (up to machine precision and accuracy of the solver) derivatives of the final state with respect to the initial state. Due to the boom in deep learning, many powerful tools exist designed to perform AD for machine learning applications and neural ODEs. We can take advantage of these tools for our application.

5.2.3 LCS Extraction

Now we review how to obtain 2-dimensional hyperbolic and elliptic LCS from their variational theory and cover the standard numerical approach. Given we have a dynamical system

$$\begin{aligned} &\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}(\mathbf{x}(t), t), \qquad \mathbf{x} \in U \subset M, \quad t \in I \subset \mathbb{R} \\ &\mathbf{x}(t_0) = \mathbf{x}_0 \end{aligned} \tag{5.8}$$

where we assume $\mathbf{v}(\mathbf{x}, t)$ is sufficiently smooth and M is a smooth manifold. Then, we can define the flow maps $\{\mathbf{F}_{t_0}^t\}$ associated with the dynamical system given by,

$$\begin{aligned} \mathbf{F}_{t_0}^t &: U \to U \\ &: \mathbf{x}_0 \mapsto \mathbf{x}(t; t_0, \mathbf{x}_0) \end{aligned} \tag{5.9}$$

which are solutions to the differential equation in Eq. (5.8), i.e.,

$$\mathbf{F}_{t_0}^t(\mathbf{x}_0) := \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{v}(\mathbf{x}(s), s) ds \tag{5.10}$$

We can take derivatives of these flow maps with respect to initial conditions to obtain the linearized flow maps,

$$\begin{aligned} \nabla \mathbf{F}_{t_0}^t(\mathbf{x}_0) &: T_{\mathbf{x}_0} U \to T_{\mathbf{x}} U \\ &: \mathbf{u}_{\mathbf{x}_0} \mapsto \mathbf{u}_{\mathbf{x}} \end{aligned} \tag{5.11}$$

and the corresponding right Cauchy Green strain tensor,

$$\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0) = \left(\nabla \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0)\right)^\top \nabla \mathbf{F}_{t_0}^{t_0+T}(\mathbf{x}_0) \tag{5.12}$$

which, as previously mentioned, is both symmetric and positive-definite. This implies that $\mathbf{C}_{t_0}^{t_0+T}(\mathbf{x}_0)$ has real eigenvalues λ_i and corresponding orthonormal eigenvectors $\boldsymbol{\xi}_i$ with $i \in$

$\{1, 2, ..., n\}$ such that,

$$\lambda_1 \geq ... \geq \lambda_n > 0 \text{ and,} \tag{5.13}$$

$$\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j \rangle = \delta_{ij}. \tag{5.14}$$

Recall that hyperbolic LCS are obtained as solution curves of the following differential equation,

$$\mathbf{r}'(s) = \boldsymbol{\xi}_2(\mathbf{r}(s)), \quad |\boldsymbol{\xi}_2| = 1 \tag{5.15}$$

and elliptic LCS can be obtained as closed solution curves (limit cycles) in the following differential equations,

$$\mathbf{r}'(s) = \boldsymbol{\eta}_\lambda^{\pm}(\mathbf{r}(s)), \quad \boldsymbol{\eta}_\lambda^{\pm} = \sqrt{\frac{\lambda^2 - \lambda_2}{\lambda_1 - \lambda_2}}\boldsymbol{\xi}_1 \pm \sqrt{\frac{\lambda_1 - \lambda^2}{\lambda_1 - \lambda_2}}\boldsymbol{\xi}_2 \tag{5.16}$$

defined on the domain

$$U_\lambda = \{\mathbf{x}_0 \in U : \ \lambda_1 \neq \lambda_2, \ \lambda_2 < \lambda^2 < \lambda_1\}. \tag{5.17}$$

Clearly, the accuracy of these eigenvector fields of $\mathbf{C}$ is of the utmost importance if one wants to obtain accurate LCS. Before the importance of the eigenvectors was understood, the typical approach to finding coherent structures was to compute the FTLE field and then look for regions of high FTLE (sometimes extract FTLE ridges) to obtain approximations of LCS. When the variational theory of LCS was developed [41, 59, 61, 67], the importance of the eigenvectors became clear.

When computing FTLE, the typical process involved computing particle trajectories for a grid of initial conditions and then approximating the gradient of the flow map with finite differences of final positions of trajectories coming from neighboring initial conditions. From here the maximum eigenvalue can be computed and the FTLE formula applied (Eq. (1.14)). This relatively coarse spacing ended up being fine for the FTLE and could actually be seen as preferable [92]. It quickly became clear that, while the eigenvalues of the linearized flow map are relatively robust to the finite difference spacing, the eigenvectors are extremely sensitive to this spacing. To obtain more accurate eigenvectors, Farazmand and Haller [41] suggested using an auxiliary grid around each main grid point where the auxiliary spacing h is very small. Then, when performing the particle integration step, integrate the main grid point along with this small auxiliary grid and use these trajectories to compute finite differences of the flow map to obtain a more accurate approximation of the gradient of the flow map (Eq. (5.11)).

This approach has served as a fix up to this point, but there are still arguably issues with this. For one, if the gradient of the flow map needed to be computed on a $n_x \times n_y$ grid of initial conditions, with the main grid this would result in $n_x n_y$ particle integrations. However this becomes $5n_x n_y$ particle integrations when using the auxiliary grid method – a $5\times$ increase in the most expensive step! In addition, as mentioned above, finite differencing is an inherently ill-conditioned and unstable numerical procedure. If one does not choose an optimal h to began with, they may need to run the simulation multiple times to find a suitable h. This means one would need to perform the expensive particle integration for at least the auxiliary grid multiple times, resulting in potentially long computation times when viewed as a whole. Another consideration here is that we are not performing finite differencing on a simple, explicit function. The auxiliary grid method involves integrating a number of nearby points to an initial condition and using the final positions of these

trajectories for finite differencing. In the hyperbolic case, the regions we are most interested in (high FTLE regions) are extremely sensitive to perturbations to the initial condition (this is essentially what FTLE measures) and therefore, different auxiliary grid spacings could theoretically result in very different trajectories for the auxiliary grid points. Clearly, an approach that circumvents these shortcomings would be highly beneficial.

5.3 AD LCS

Now that we have laid the framework, the idea is quite simple: solve the ODE using the neural ODE framework – generate an evaluation trace of the ODE solver steps and perform automatic differentiation by computing either the forward tangent trace or the reverse adjoint trace. Instead of applying AD to some neural network and computing derivatives with respect to parameters, we apply AD to the solutions of our ODE and compute the derivatives of the final positions with respect to initial conditions. This gives us approximations of the linearized flow map that are as accurate as possible given the ODE solver and precision. We then use these accurate linearized flow maps to obtain accurate eigenvectors of $\mathbf{C}$, and use these eigenvectors to compute accurate LCS. This approach also eliminates the parameter h, avoiding the potential need to re-run the simulation to find an acceptable h. We can quite easily take advantage of existing powerful tools created for the neural ODE workflow to implement this method. Some of these tools allow us to use interpolants that are differentiable in the AD framework, allowing us to apply this method to flows for which we only have numerical velocity data. We refer to LCS obtained from this method as *Automatic Differentiation Lagrangian Coherent Structures (AD LCS).*

We mainly focus on 2D hyperbolic and elliptic LCS here but note that this method can be used for a flow of any dimension and can be used for any type of coherent structure

method which relies on linearized flow maps (parabolic LCS [42], 3D hyperbolic and elliptic LCS [12], diffusive transport barriers [68], active transport barriers [69], shape coherent sets [101], etc.).

For deep neural networks, AD is advantageous largely due to its efficiency compared to other computational differentiation methods. In our application, the accuracy of the derivatives is the key advantage over other methods. As we will see, in our implementation AD remains efficient and often outperforms the auxiliary grid method in terms of computation times. While our implementation utilizes highly efficient libraries (detailed below), which provide state-of-the-art implementations for general-purpose AD and neural ODEs, we acknowledge that further optimizations to our specific context may be possible. These libraries are often optimized for large-scale machine learning problems with high-dimensional state spaces (large n). Our application involves low-dimensional systems ($n = m = 2$ or $n = m = 3$). These are cases that are not given much attention in the machine learning world and there may be different optimizations to the workflow in this context that would provide greater efficiency. In addition, there may be more efficient ways to implement and deal with interpolants of velocity fields in this context. We do not comment on these topics further here and leave these open as a directions for future work.

5.3.1 Implementation

The current implementation makes use of `JAX` [15] for automatic differentiation and `Diffrax` [85] for differentiable ODE solvers. `JAX` is a powerful library, developed by Google Research teams, that provides a unified interface and syntax for numerical array computations that can run on CPUs, GPUs or TPUs. `JAX` provides built-in JIT compilation (via Open XLA [29]), automatic vectorization, and, most importantly to us, efficient automatic differen-

tiation. We make use of the JIT compilation and automatic vectorization for improved computation speeds. `Diffrax` is a Python library, developed by Patrick Kidger, for solving a wide variety of differential equations that implements a number of efficient solvers and integrates seamlessly with `JAX`. All AD is performed via reverse-mode AD directly applied to the solver trace with checkpointing for improved memory handling, the default in `Diffrax` (see Chapter 5 in [85] for more details, this is referred to as the "discretise-then-optimise" approach in the cited work).

`JAX` currently implements a gridded linear interpolant that we use for numerical flows. There is a package `interpax` [30], that implements splines as well, though we have not yet had success with this package due to memory issues that arise when applying it in our context to the flows we are using. We are actively exploring ways to implement efficient interpolants in this framework. For now, we just use the gridded linear interpolant provided by `JAX`.

5.3.2 Related Work

As mentioned, the earliest appearance of these ideas that we are aware of are from an abstract for an APS DFD talk by Gilpin in 2021 [51]. From a brief correspondence with the author, he was primarily interested in computing FTLE using a similar method and was able to get it working with ODEs which have a closed form RHS. In the following year, Tyler and Wittig [158] published a very nice paper that tackled these ideas through the lens of Differential Algebra, and they coined the method they developed DA-LCS. While many of the underlying ideas are similar to our method (they go a step further by applying DA to the eigenvectors as well since derivatives of eigenvectors are needed for 3D LCS), the implementations are rather different. Surprisingly, this work has not received much attention. This is perhaps because Differential Algebra is a more niche field, because they focus specifically on hyperbolic 3D

LCS, or because their software implementation could be seen as harder to get up and running for a practitioner. Whatever the case, we hope to shed light on this excellent work. A key difference between our work and those mentioned above is that we have successfully applied our method to flows defined by numerical data. While this could theoretically be done in the DA-LCS framework, the authors did not address this case.

5.4 Results

Since we are most interested in using AD to compute LCS, we focus on the eigenvectors of the Cauchy Green tensor, though we note one could compute FTLE with this method as well. As a baseline, we use AD with a high order solver (DOPRI8 [125]), strict error tolerances (relative tolerance = 1E-14, absolute tolerance = 1E-16), and double precision and compare all other methods to this. For AD, we use both the DOPRI8 method (an explicit 8th order Runge-Kutta method with a 7th order embedded method for error correction) and TSIT5 [157] (a relatively new explicit 5th order Runge-Kutta method with an embedded 4th order method for error correction). These solvers are all provided by the `Diffrax` package. For numerical differentiation, we use the same exact solvers provided by `Diffrax`, but we compute the gradient of the flow map with standard finite differencing instead of AD.

For all methods, we use 7 evenly spaced (on a log scale) relative tolerances ranging from 1E-3 to 1E-11 and corresponding absolute tolerances from 1E-5 to 1E-13 for the ODE solvers. For numerical differentiation we use 7 evenly spaced (on a log scale) auxiliary grid spacings ranging from 1E-2 to 1E-8. We perform all these simulations using both single and double precision.

We look at the QGE flow [107] as a test bed to compare AD and numerical differentiation for computing linearized flow maps. We look at this flow at $t_0 = 0.0$ (this corresponds to

$t_0 = 10.0$ in the original data set) and use an integration time of $T = 0.1$. The flow is defined on the domain $[0, 1] \times [0, 2]$ and the velocity data is on a 257×513 grid with even spacing. Refer to the cited paper for more details.

All computations are performed on a single GPU, as `JAX` is designed to leverage the high-performance capabilities of GPUs and TPUs through XLA compilation. The GPU used for these computations is a NVIDIA GeForce(R) GTX 1080Ti (Founder's Edition) with 11 GB VRAM (GDDRX5), CC 6.1, and 3584 CUDA cores. We make use of `JAX's` automatic vectorization for both the direct ODE solve (for numerical differentiation) and the AD method (for automatic differentiation) for improved performance.

As a measure of error, we compute the absolute value of the difference between the angle of the baseline (high order, strict tolerance AD method) eigenvectors and the estimated eigenvectors (all other methods), accounting for the sign ambiguity, and then averaging these errors over all grid points (excluding boundary points) and refer to this as the mean absolute error (MAE). The results can be seen in Figure 5.5. For more clarity, we break up the plots by ODE solver in Figure 5.6 (TSIT5) and Figure 5.7 (DOPRI8).

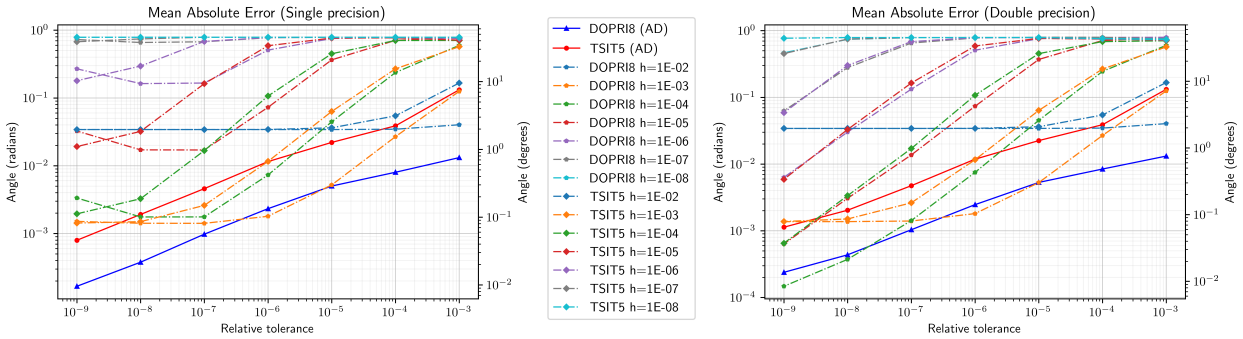

Figure 5.5: Comparison of mean absolute error for TSIT5 and DOPRI8 with different tolerances and auxiliary grid spacings (for numerical differentiation). Left: single precision, Right: double precision. Dashed lines correspond to numerical differentiation, solid lines correspond to automatic differentiation.

It is immediately clear that, with the exception of a few specific pairings of auxiliary grid spacings and tolerances, the AD method produces errors that are consistently below that of their numerical differentiation counterparts at the same tolerances. We can see that the numerical differentiation methods are quite sensitive to the auxiliary grid spacing h being used and the optimal spacing seems to be somewhere between h = 1E-3 to h = 1E-4 (see Appendix C.2). This sensitivity holds true for choosing auxiliary grid spacing both too small and too large. For this flow, choosing an auxiliary grid spacing too large results in an essentially consistent error on the order of 10^0 degrees (regardless of tolerance) while choosing h too small results in extremely large average errors near $\pi/2$ (45°). The AD methods do not have to worry about choosing an h and thus shows steadily decreasing errors with tolerance.

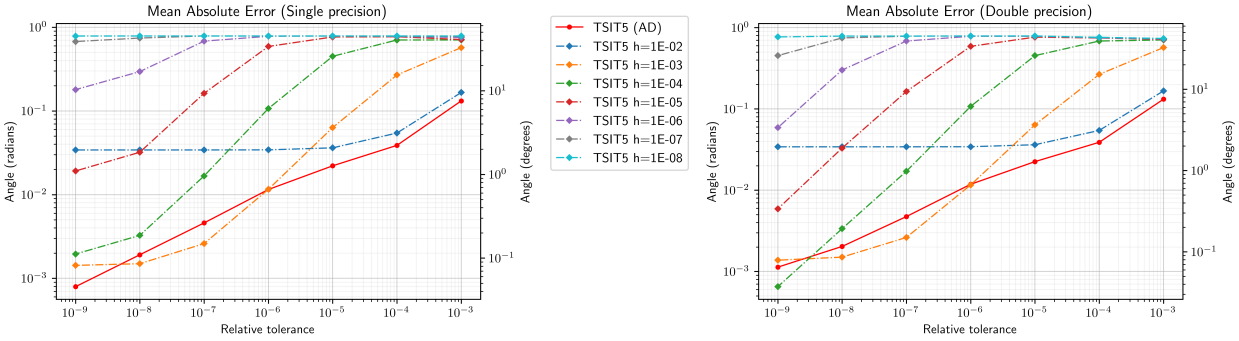

Figure 5.6: Comparison of mean absolute error for TSIT5 with different tolerances and auxiliary grid spacings (for numerical differentiation). Left: single precision, Right: double precision. Dashed lines correspond to numerical differentiation, solid lines correspond to automatic differentiation.

We notice something surprising when we compare methods that used different precisions. In

Figure 5.8 and Figure 5.9 we look at how precision affects errors for TSIT5 and DOPRI8 respectively. For numerical differentiation, as the tolerance becomes more stringent, the single precision methods begin to perform worse (this is more pronounced for the DOPRI8 method). Surprisingly, for AD methods, precision has little effect on the error; in fact, the single-precision variants often outperform their double-precision counterparts while running in about half the time (see Appendix C.1). We do not currently have an explanation for this result and this is an area that requires further investigation.

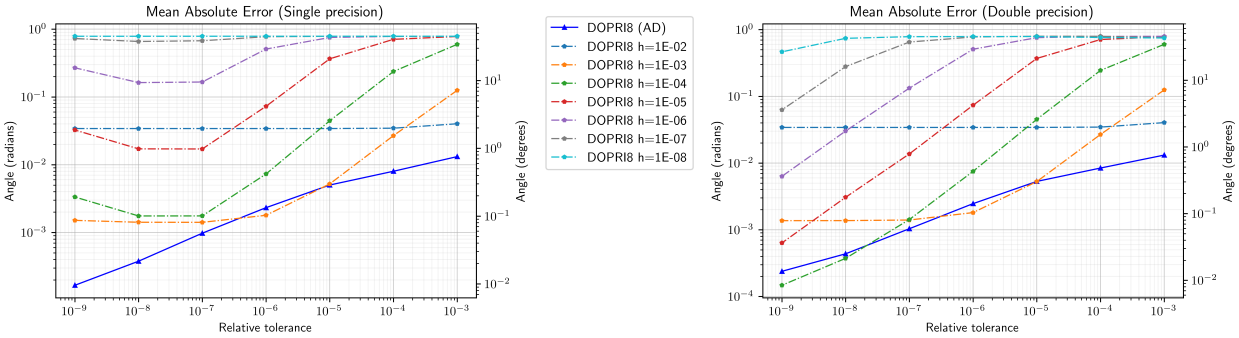

Figure 5.7: Comparison of mean absolute error for DOPRI8 with different tolerances and auxiliary grid spacings (for numerical differentiation). Left: single precision, Right: double precision. Dashed lines correspond to numerical differentiation, solid lines correspond to automatic differentiation.

In Table 5.1, we compare the top 7 performers (in terms of average error) for each method that have a runtime below 60 seconds. The full tables for the AD methods and numerical differentiation methods can be found in Appendices C.1 and C.2 respectively.

The top-performing methods from both AD and numerical differentiation produce errors on roughly the same order, but again we note that this is for specific auxiliary grid values for numerical differentiation. We can also see that runtimes for AD and numerical differentiation are about the same (when considering the same ODE solver with the same tolerances) and in fact, AD is slightly faster in most cases. In addition, for the numerical differentiation methods, different combinations of auxiliary grid sizing and tolerances produce the lowest

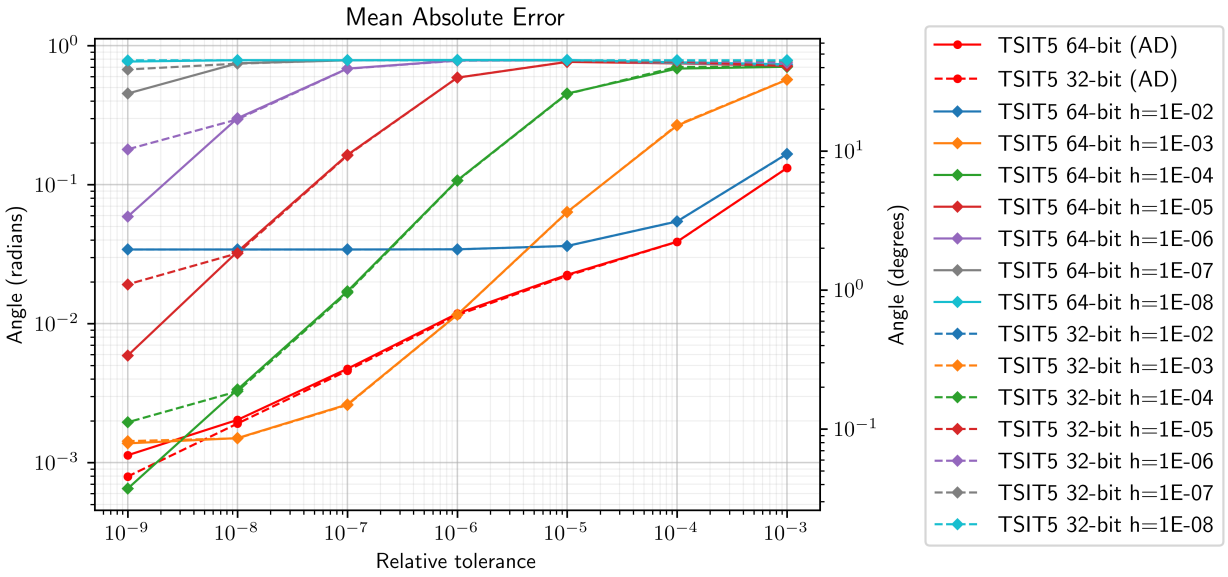

Figure 5.8: Comparison of mean absolute error for TSIT5 with different tolerances, auxiliary grid spacings (for numerical differentiation), and precision. Dashed lines correspond to single precision (32-bit) and solid lines correspond to double precision (64-bit).

errors and if one does not pick a "good" combination on the first run, the user would need to re-run their simulation to find a more optimal combination. On the other hand, AD consistently produces low errors, and sometimes does so with fast runtimes (e.g., the single-precision TSIT5 method with rtol = 1E-6 and atol = 1E-8 has an MAE on the order of 10^{-1} degrees with a 4.85s runtime and the single-precision DOPRI8 method with rtol = 1E-4 and atol = 1E-6 also has MAE on the order of 10^{-1} degrees with a runtime of 4.04s).

To get a grasp on what these errors mean in reality, we look at a few plots as the top performers table can be somewhat misleading. In Figures 5.10 and 5.11, we observe how the grid spacing affects the eigenvectors using the double precision DOPRI8 method with relative tolerance = 1E-8, absolute tolerance = 1E-10, and auxiliary grid spacings h = 1E-2, h = 1E-4, h = 1E-8. We overlay the eigenvector field corresponding to the smaller eigenvalue, as this is the field used to extract hyperbolic LCS (the eigenvectors should be aligned with FTLE ridges). In Figure 5.10, we can clearly see that when h is too small, the eigenvector field is

Table 5.1: Comparison of Top Performers by Average Error (Runtime < 60s)

Differentiation	Method	Precision	h	Rel Tol	Abs Tol	Avg Err (rad)	Avg Err (deg)	Runtime (s)
Automatic	TSIT5	Single	—	1E-9	1E-11	7.95×10^{-4}	0.0455	47.49
Automatic	TSIT5	Single	—	1E-8	1E-10	1.91×10^{-3}	0.1095	24.74
Automatic	TSIT5	Double	—	1E-8	1E-10	2.03×10^{-3}	0.1164	58.61
Automatic	TSIT5	Single	—	1E-7	1E-9	4.59×10^{-3}	0.2631	11.63
Automatic	TSIT5	Double	—	1E-7	1E-9	4.74×10^{-3}	0.2714	21.82
Automatic	TSIT5	Single	—	1E-6	1E-8	1.15×10^{-2}	0.6599	4.85
Automatic	TSIT5	Double	—	1E-6	1E-8	1.18×10^{-2}	0.6776	8.51
Automatic	DOPRI8	Single	—	1E-7	1E-9	9.78×10^{-4}	0.0560	42.80
Automatic	DOPRI8	Single	—	1E-6	1E-8	2.33×10^{-3}	0.1333	19.37
Automatic	DOPRI8	Double	—	1E-6	1E-8	2.46×10^{-3}	0.1411	38.65
Automatic	DOPRI8	Single	—	1E-5	1E-7	5.02×10^{-3}	0.2876	9.23
Automatic	DOPRI8	Double	—	1E-5	1E-7	5.30×10^{-3}	0.3039	16.47
Automatic	DOPRI8	Single	—	1E-4	1E-6	8.01×10^{-3}	0.4590	4.04
Automatic	DOPRI8	Double	—	1E-4	1E-6	8.42×10^{-3}	0.4822	7.51
Numerical	TSIT5	Single	1E-3	1E-9	1E-11	1.43×10^{-3}	0.0818	47.02
Numerical	TSIT5	Single	1E-3	1E-8	1E-10	1.50×10^{-3}	0.0857	25.79
Numerical	TSIT5	Double	1E-3	1E-8	1E-10	1.50×10^{-3}	0.0860	41.00
Numerical	TSIT5	Single	1E-4	1E-9	1E-11	1.95×10^{-3}	0.1118	46.15
Numerical	TSIT5	Single	1E-3	1E-7	1E-9	2.60×10^{-3}	0.1490	12.22
Numerical	TSIT5	Double	1E-3	1E-7	1E-9	2.62×10^{-3}	0.1503	16.78
Numerical	TSIT5	Single	1E-4	1E-8	1E-10	3.26×10^{-3}	0.1867	26.30
Numerical	DOPRI8	Single	1E-3	1E-7	1E-9	1.41×10^{-3}	0.0810	44.44
Numerical	DOPRI8	Single	1E-4	1E-7	1E-9	1.76×10^{-3}	0.1007	44.78
Numerical	DOPRI8	Single	1E-3	1E-6	1E-8	1.80×10^{-3}	0.1029	21.26
Numerical	DOPRI8	Double	1E-3	1E-6	1E-8	1.81×10^{-3}	0.1035	34.61
Numerical	DOPRI8	Single	1E-3	1E-5	1E-7	5.21×10^{-3}	0.2986	11.23
Numerical	DOPRI8	Double	1E-3	1E-5	1E-7	5.27×10^{-3}	0.3022	16.20
Numerical	DOPRI8	Single	1E-4	1E-6	1E-8	7.30×10^{-3}	0.4184	21.76

extremely inaccurate and will not extract any LCS (at least not any true LCS). It may seem that the larger grid spacing (h = 1E-2) would work, but upon further inspection we see that is not the case. In Figure 5.11, we focus in a hyperbolic LCS (the high FTLE ridge) and can see that the larger grid spacing is mostly correct, but sometimes produces eigenvectors not aligned with the FTLE ridge. This would lead to the LCS algorithm terminating solution curves that pushed the trajectory off the ridge, leading to a failed extraction of the full hyperbolic LCS. The middle plot (h = 1E-4) produces satisfactory eigenvectors and would lead to accurate extraction of LCS.

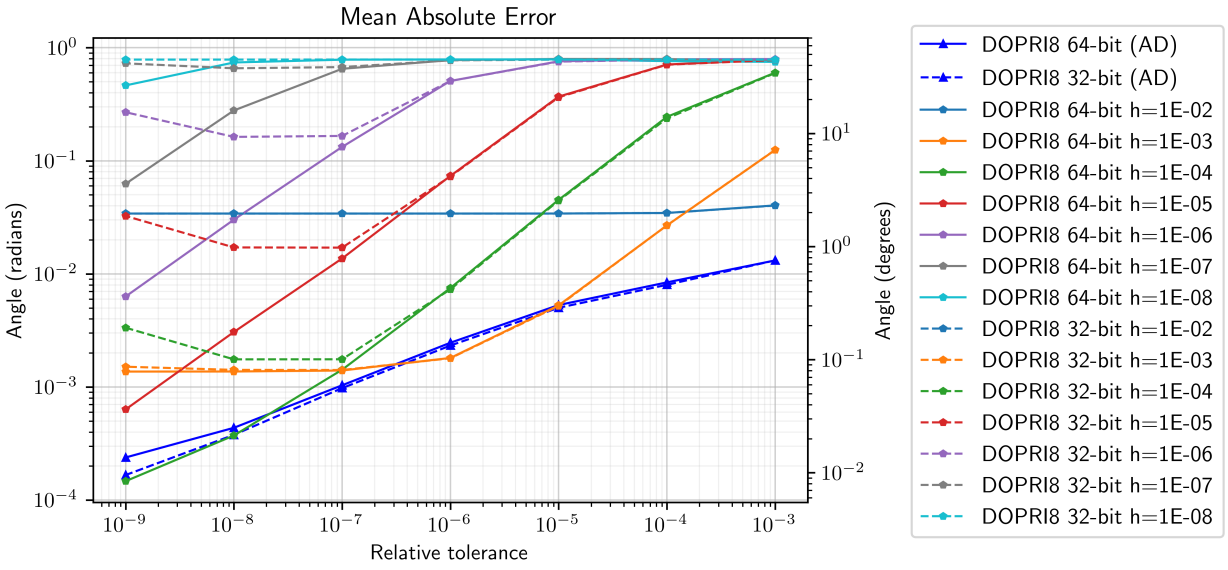

Figure 5.9: Comparison of mean absolute error for DOPRI8 with different tolerances, auxiliary grid spacings (for numerical differentiation), and precision. Dashed lines correspond to single precision (32-bit) and solid lines correspond to double precision (64-bit).

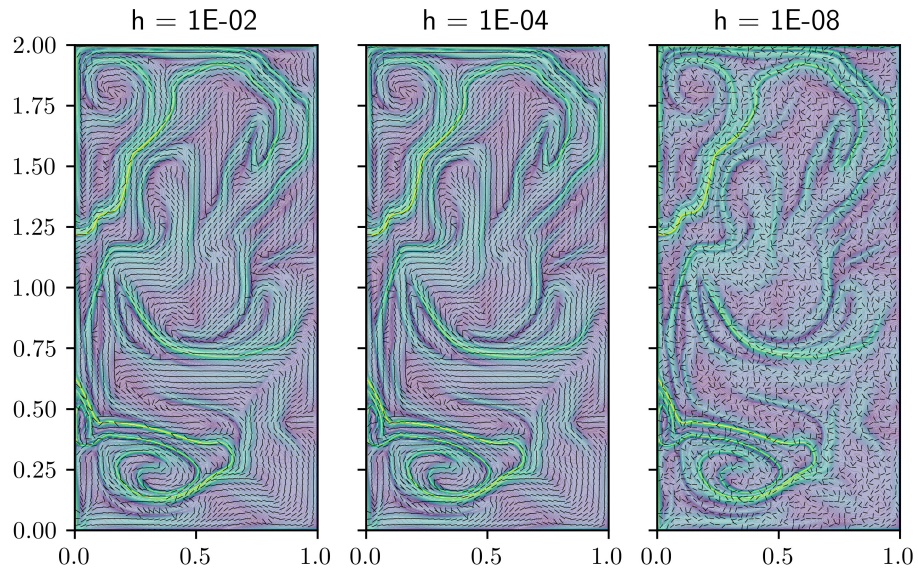

Figure 5.10: Eigenvectors overlaid on FTLE field for double precision DOPRI8 method with relative tolerance = 1E-8, absolute tolerance = 1E-10, and auxiliary grid spacings Left: h = 1E-2, Middle: h = 1E-4, and Right: h = 1E-8.

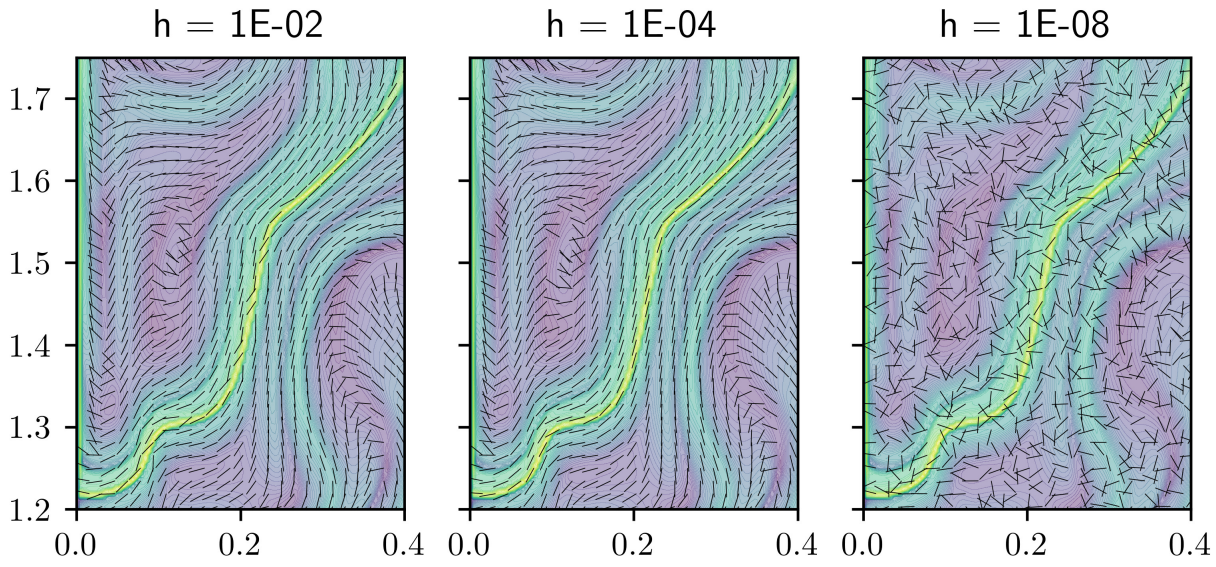

Figure 5.11: Eigenvectors overlaid on FTLE field for double precision DOPRI8 method with relative tolerance = 1E-8, absolute tolerance = 1E-10, and auxiliary grid spacings Left: h = 1E-2, Middle: h = 1E-4, and Right: h = 1E-8. Focus on a hyperbolic LCS.

When we look at the AD case, we focus again on the DOPRI8 method as this showed the best results. We produce the same overlay plots, but this time use the baseline eigenvectors which were treated as the "true" values and compare to the single precision DOPRI8 method with the laxest tolerances (relative tolerance = 1E-3 and absolute tolerance = 1E-5). This is arguably the most important point. When using the DOPRI8 method, even when using the laxest tolerances and single precision, we can obtain eigenvectors with sufficiently low errors (MAE = 0.7548°) with a very low runtime (2.14 seconds). In both Figure 5.12 and Figure 5.13, we can see that these eigenvectors fields are nearly indistinguishable and, even with these very lax tolerances, accurate LCS can be extracted. Therefore, for an optimal balance of accuracy and efficient, we advocate to use a high-order method (like DOPRI8) with relatively lax tolerances.

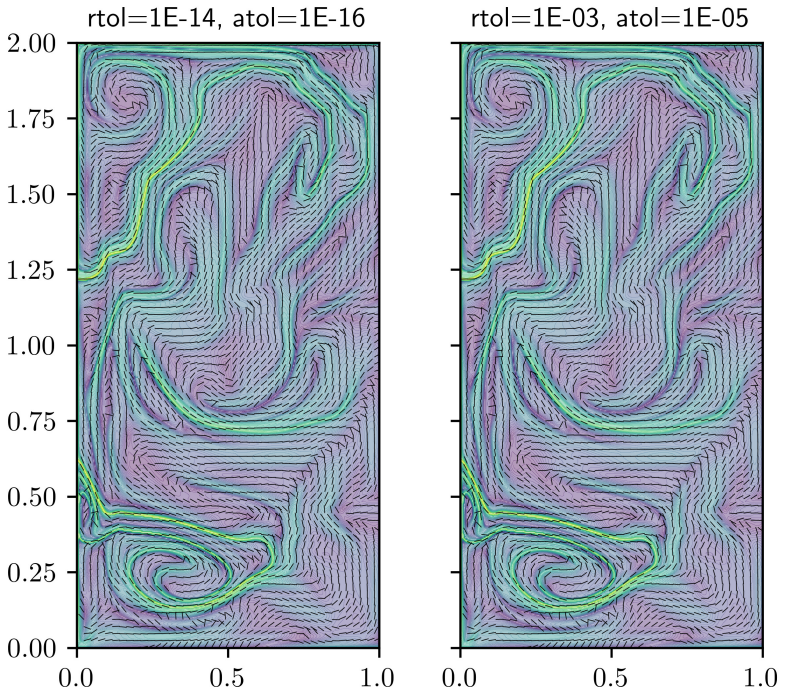

Figure 5.12: Eigenvectors overlaid on FTLE field for Left: double precision DOPRI8 method with relative tolerance = 1E-14, absolute tolerance = 1E-16, and Right: single precision DOPRI8 method with relative tolerance = 1E-3, absolute tolerance = 1E-5.

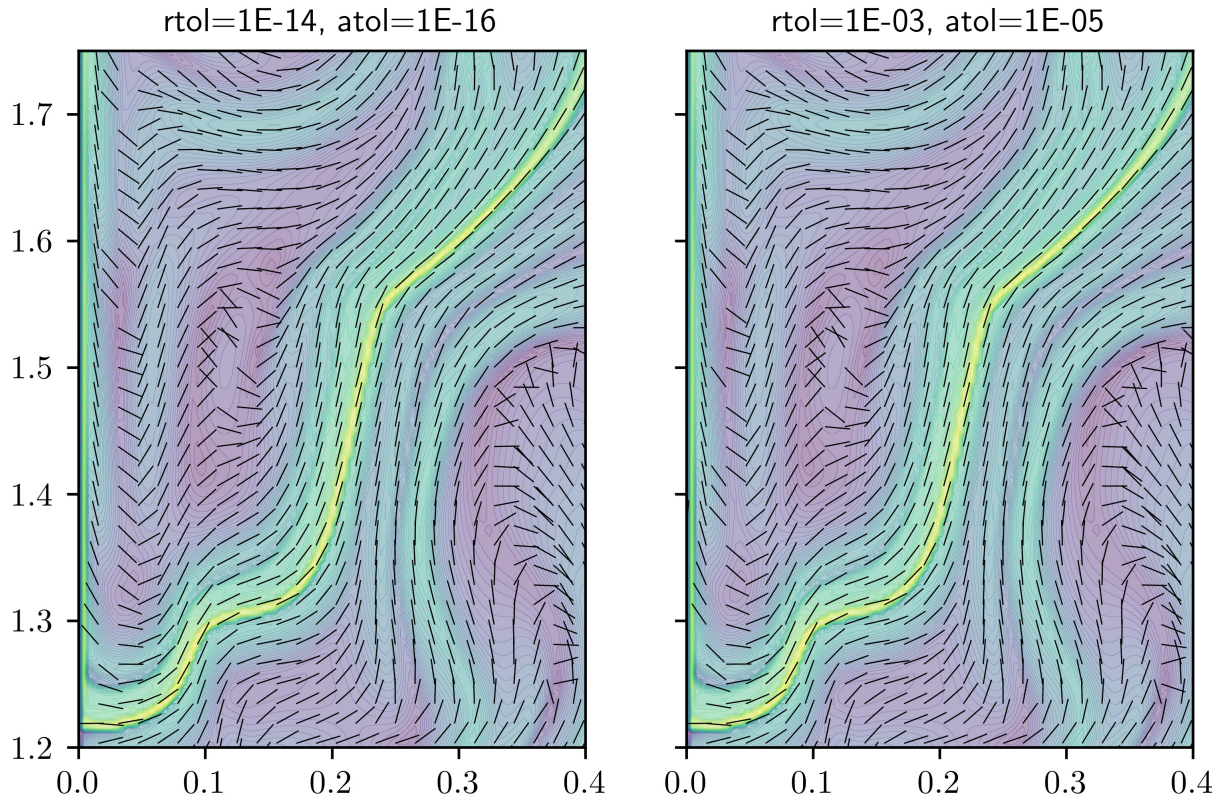

Figure 5.13: Eigenvectors overlaid on FTLE field for Left: double precision DOPRI8 method with relative tolerance = 1E-14, absolute tolerance = 1E-16, and Right: single precision DOPRI8 method with relative tolerance = 1E-3, absolute tolerance = 1E-5. Focus on a hyperbolic LCS.

5.5 Conclusion

We have presented a new framework for computing LCS which provides more accurate linearized flow maps needed for LCS extraction. By using the neural ODE framework and applying AD to numerical solutions of ODEs, we can obtain accurate LCS with one less parameter (auxiliary grid spacing) and no need to re-run simulations to find acceptable values for this parameter. Furthermore, we have demonstrated that the AD method yields satisfactory results across a wide range of error tolerances, the same of which cannot be said for the

numerical differentiation approach. We have shown this methodology can be implemented quite simply using existing AD and neural ODE packages, allowing this approach to be adopted by a wider audience. In addition, we have shown this approach works well for flows which are represented by numerical velocity data, a case that had not been demonstrated before.

There is still much to do moving forward. The main focuses will be on integrating efficient cubic spline interpolants to further decrease our true errors with respect to reality, streamline this implementation and release a public package which makes this easy for non-experts to use, compare the actual LCS that is computed from the various methods to find what is an "acceptable" error for the eigenvectors, and do all of these comparisons on a variety of different flows with a deeper analysis. In addition, there are some subtle points relating to robustness to precision and the fact that numerical differentiation produced lower errors than AD for *any* runs (when the tolerances and solvers were equal) that need to be investigated further. The robustness to precision may be due to the tolerances being larger than the loss of precision when going from double to single precision and the avoidance of the numerical differentiation routine that causes this to become an issue at lower tolerances. The slightly better errors achieved by numerical differentiation for the few isolated cases may be due to the linear interpolant being used and how `JAX` is dealing with differentiating this interpolant. At this time, we cannot say for sure if any of these explanations are the true reasons and a deeper dive is required to settle these questions.

In the end, the goal here was really to demonstrate that using AD for LCS extraction is the optimal way moving forward. LCS extraction relies on precise and accurate eigenvectors of the Cauchy Green tensor and standard numerical differentiation is not the best tool for this job. AD provides derivatives that are as accurate as possible given the numerical setting, and this is precisely what is needed for accurate LCS extraction. We hope this approach is

adopted the LCS community and are excited to see these tools used in practice.

Chapter 6

Conclusions

6.1 Summary

This dissertation made a number of contributions to the numerical framework for coherent structures and furthering the theory. In Ch. 2, we presented `NumbaCS`, a Python package that efficiently implements many coherent structure methods in a user-friendly manner. This package provides an easy-to-use starting point for those new to the theory and a powerful tool for experienced practitioners. By making use of various open-source Python packages, `NumbaCS` is able to maintain the simple user experience provided by dynamically-typed languages while achieving runtimes closer to that of a statically-typed language.

In Ch. 3, a number of coherent structure methods are applied to a large-scale atmospheric transport event, the 2020 "Godzilla" dust storm. We demonstrated the utility of these tools on an atmospheric phenomena at a scale not previously seen for Earth's atmosphere. We showed that, even with a simplified implementation of these tools, key transport structures responsible for the evolution of this historic dust storm could be identified, some of which had gone unnoticed in the literature. Through this presentation, we aimed at making a wider audience of atmospheric scientists aware of these tools and their potential utility.

In Ch. 4, we presented an extension of the theory of finite-time diffusive transport barriers to the instantaneous case, much like what was done with objective Eulerian coherent structures.

By defining the appropriate functional, we derived barrier equations for instantaneous elliptic diffusive flux-rate barriers for general diffusion structure, the instantaneous analogues of diffusive transport barriers. By looking at an example with nontrivial diffusion structure, we demonstrated that these instantaneous barriers correctly did not identify diffusive barriers that were destroyed by the diffusion structure, much like finite-time structures.

Finally, in Ch. 5, we presented a new framework for computing LCS from their variational theory. By leveraging automatic differentiation and ideas coming from neural differential equations, a new method to compute accurate linearized flow maps needed for LCS extraction is detailed. We demonstrated the usefulness of this approach on a complicated flow represented by numerical velocity data, a case not previously tackled. We showed this approach consistently produces sufficiently low errors in eigenvector angles (when using the higher order solver) while eliminating the auxiliary grid spacing parameter needed for the traditional method. By removing this parameter, this method also eliminates the potential need to re-run simulations to find an acceptable window for this parameter, a computationally expensive process. We argue this is the optimal way to compute linearized flow maps for LCS extraction moving forward. In addition, we present a straightforward framework for implementing this method in Python.

While each chapter addresses a specific aspect of the emerging field of objective coherent structures, together they help build a more complete and refined picture of how to understand, model, and compute key transport structures in fluid flows. This dissertation advances the field through the development of new powerful numerical tools and methods, a demonstration of coherent structure methods in a large-scale transport application – revealing previously unseen structure, and extending the theory for instantaneous diffusive and stochastic transport barriers. By integrating numerical innovation, theoretical development, and practical application, this work improves the computation and application of coherent

structure theory and helps extend its reach to wider audiences and broader contexts.

These contributions collectively strengthen the analytical and computational foundations of the coherent structure framework. `NumbaCS` provides a highly efficient and user-friendly package for computing and extracting coherent structures, making these tools accessible to a wider audience while being easily applicable to large-scale, real-world scenarios. The "Godzilla" dust application demonstrates a simple implementation for a practitioner, and the power of these tools for identifying key transport structures. The theoretical extension opens the door to studies of instantaneous transport barriers in diffusive and stochastic environments, widening the applicability. The new AD LCS methods provides a more robust and consistent method for extraction of LCS and is equally applicable to any extraction method which relies on accurate linearized flow maps. Through these contributions, this dissertation assists in the maturation of the coherent structure theory and helps further establish these tools as essential components for advancing our understanding of transport in fluid flows.

6.2 Future Directions

While this dissertation contributes to the maturation of the theory of Lagrangian and objective Eulerian coherent structures and their implementations, it also identifies key avenues for research moving forward and further optimizations of presented implementations.

`NumbaCS` was presented in Ch. 2 and detailed a road map for development. Implementing many of these methods is the main direction moving forward. In addition, there a few already implemented methods that could benefit from further optimizations that will be tackled in the coming months. We wish to emphasize that the automatic differentiation framework presented in Ch. 5 does not impede any plans for development of `NumbaCS` moving forward.

We understand that not everyone has access to GPUs capable of being utilized within the `JAX` framework and therefore, the CPU focused implementation of `NumbaCS` remains useful to a wider audience. In addition, though not a true apples to apples comparison, `NumbaCS` is able to achieve similar runtimes to the GPU implementation we put forth in Ch. 5. We are actively investigating efficient CPU implementations of AD that could be integrated into `NumbaCS` or utilized in a new CPU focused coherent structure package.

The atmospheric application of coherent structures in Ch. 3 aimed to demonstrate the usefulness of these tools to a wider audience of atmospheric scientists, though some interesting potential directions came out of this work. While we don't claim coherent structures are the solution to every problem, we do believe they provide valuable insight lacking from many traditional methods in the geophysical sciences. With this in mind, a push to demonstrate how these tools can be incorporated into the standard atmospheric scientist's or oceanographic scientist's workflow remains an important focus. For real-time applications and data production, much like "real-time" velocity, temperature, pressure, vorticity, etc. are provided by various national atmospheric and oceanographic agencies, backward time coherent structure information could just as easily be provided by these agencies if coherent structure methods were introduced into their workflows. By using something like the flow map composition method detailed in Ch. 2, this data could be produced rapidly. Providing these tools and the data they output to a wider audience may produce interesting and novel applications not previously seen before and could potentially improve certain forecasts in these fields.

By defining diffusive flux-rate barriers in Ch. 4, we have successfully expanded the theory of diffusive transport barriers to the instantaneous case for general diffusion structure. Possible avenues moving forward could be defining an appropriate persistence metric (or showing the advective persistence metric is appropriate), properly defining the instantaneous version of the diffusive barrier strength field, and further studying the coupling between diffusion

structure and diffusivity for which this theory produces satisfactory results. In addition, we hope practitioners find novel applications for the general diffusion structure case, whether they be in diffusive or stochastic settings.

The work which provides the most opportunity moving forward is the AD LCS presented in Ch. 5. This work is relatively new and there is much we wanted to do that could not be completed in time to fit in this dissertation. The clearest is to do a deeper and more systematic study of the accuracy boost over numerical differentiation. We would like to test on more flows (both analytical and numerical) and for more frames to obtain a better picture of the advantages. Another important step moving forward is to fully develop an open-source package that implements the methods detailed in this chapter as well as the LCS extraction techniques. Along the way, we would like to dive further into optimizations tailored to the specific low-dimensional systems arising from our application. Some more subtle points warrant further investigation; the main two being the precision robustness of AD, and the fact that numerical differentiation outperformed AD for *any* cases (when ODE solver and tolerance were the same). We could speculate on why that happened, but that would be only speculation and requires a deeper dive to produce a satisfactory answer. This AD LCS work is the topic we are most excited about moving forward as it provides a substantial improvement over the traditional method and many avenues for future research.

Although the theory of coherent structures is relatively new (in the grand scheme of science), we believe it is at an important stage in its maturation. Most of the base theory has been put on rigorous footing and the numerical implementations have become more efficient and accurate. The next step is implementing these tools into existing workflows in various geophysical disciplines. It is through this integration and wide availability to experts in the geophysical sciences that the true power of these tools can be realized. The main aim of this dissertation has been to contribute to bringing these methods to the next level of

maturation through efficient and user-friendly numerical implementations, demonstrating their application in a large-scale geophysical context, extending the theory in specific cases, and the development of a new and improved numerical framework for computation. It is my hope that this work has fulfilled that aim, at least in part, and that it will serve as a stepping stone for continued growth in this field and its real-world applications.

Bibliography

[1] Global Modeling and Assimilation Office (GMAO) (2015), MERRA-2 inst3_3d_asm_Np: 3d, 3-Hourly, Instantaneous, Pressure-Level, Assimilation, Assimilated Meteorological Fields V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), 2015. Accessed on 09-22-2023.

[2] Martin S. Alnaes, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. Unified Form Language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40, 2014. doi: 10.1145/2566630.

[3] Silvia Alonso-Pérez, Emilio Cuevas, Xavier Querol, Juan Carlos Guerra, and Carlos Pérez. African dust source regions for observed dust outbreaks over the Subtropical Eastern North Atlantic region, above 25 N. *Journal of Arid Environments*, 78:100–109, 2012.

[4] A. Asutosh, V. Vinoj, Nuncio Murukesh, Ramakrishna Ramisetty, and Nishant Mittal. Investigation of June 2020 giant Saharan dust storm using remote sensing observations and model reanalysis. *Scientific reports*, 12(1):1–14, 2022.

[5] A Badza, T. Mattner, and S. Balasuriya. How sensitive are Lagrangian coherent structures to uncertainties in data? *Physica D: Nonlinear Phenomena*, 444:133580, 2023. ISSN 0167-2789. doi: 10.1016/j.physd.2022.133580.

[6] Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and

Garth N. Wells. DOLFINx: the next generation FEniCS problem solving environment. preprint, 2023.

[7] Michael Bartholomew-Biggs, Steven Brown, Bruce Christianson, and Laurence Dixon. Automatic differentiation of algorithms. *Journal of Computational and Applied Mathematics*, 124(1):171–190, 2000. ISSN 0377-0427. doi: https://doi.org/10.1016/S0377-0427(00)00422-2. URL https://www.sciencedirect.com/science/article/pii/S0377042700004222. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[8] Alex Pablo Encinas Bartos, Bálint Kaszás, and George Haller. haller-group/TBarrier: TBarrier, Jun 2022. URL https://doi.org/10.5281/zenodo.6779400.

[9] Mike Bauer, George Tselioudis, and William B. Rossow. A New Climatology for Investigating Storm Influences in and on the Extratropics. *Journal of Applied Meteorology and Climatology*, 55(5):1287 – 1303, 2016. doi: 10.1175/JAMC-D-15-0245.1. URL https://journals.ametsoc.org/view/journals/apme/55/5/jamc-d-15-0245.1.xml.

[10] Atilim Gunes Baydin, Barak A. Pearlmutter, and Alexey Andreyevich Radul. Automatic differentiation in Machine Learning: a Survey. *CoRR*, abs/1502.05767, 2015. URL http://arxiv.org/abs/1502.05767.

[11] Hongru Bi, Siyu Chen, Daizhou Zhang, Yong Wang, Litai Kang, Khan Alam, Mingjin Tang, Yu Chen, Yue Zhang, and Danfeng Wang. The Circumglobal Transport of Massive African Dust and Its Impacts on the Regional Circulation in Remote Atmosphere. *Bulletin of the American Meteorological Society*, 105(3):E605 – E622, 2024. doi: 10.1175/BAMS-D-23-0072.1. URL https://journals.ametsoc.org/view/journals/bams/105/3/BAMS-D-23-0072.1.xml.

[12] Daniel Blazevski and George Haller. Hyperbolic and elliptic transport barriers in three-dimensional unsteady flows. *Physica D: Nonlinear Phenomena*, 273-274:46–62, 2014. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2014.01.007. URL https://www.sciencedirect.com/science/article/pii/S0167278914000165.

[13] P Bonasoni, T Colombo, R Lenaz, G Tesi, F Evangelisti, G Giovanelli, F Ravegnani, and R Santaguida. Effect of Saharan dust transport on ozone and carbon dioxide concentration. *The Impact of Desert Dust Across the Mediterranean*, pages 313–322, 1996.

[14] Michael G Bosilovich, Rob Lucchesi, and M Suarez. MERRA-2: File specification. Technical report, 2015.

[15] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

[16] F. Briol and F. d'Ovidio. Lagrangian, 2011. URL https://github.com/CNES/aviso-lagrangian.

[17] Steven L. Brunton and Clarence W. Rowley. Fast computation of finite-time Lyapunov exponent fields for unsteady flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(1):017503, 01 2010. ISSN 1054-1500. doi: 10.1063/1.3270044.

[18] Roxana Bujack and Ariane Middel. State of the art in flow visualization in the environmental sciences. *Environmental Earth Sciences*, 79(2):65, Jan 2020. ISSN 1866-6299. doi: 10.1007/s12665-019-8800-4. URL https://doi.org/10.1007/s12665-019-8800-4.

[19] Roxana Bujack, Soumya Dutta, Irene Baeza Rojo, Duan Zhang, and Tobias Günther. Objective Finite-Time Saddles and their Connection to FTLE. In *Eurographics Conference on Visualization*, 2019. URL https://api.semanticscholar.org/CorpusID:140081247.

[20] Suzana J. Camargo and Stephen E. Zebiak. Improving the Detection and Tracking of Tropical Cyclones in Atmospheric General Circulation Models. *Weather and Forecasting*, 17(6):1152 – 1162, 2002. doi: 10.1175/1520-0434(2002)017<1152:ITDATO>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/wefo/17/6/1520-0434_2002_017_1152_itdato_2_0_co_2.xml.

[21] Toby N Carlson and Joseph M Prospero. The large-scale movement of Saharan air outbreaks over the northern equatorial Atlantic. *Journal of Applied Meteorology and Climatology*, 11(2):283–297, 1972.

[22] D. Carvalho. An assessment of NASA's GMAO MERRA-2 reanalysis surface winds. *Journal of Climate*, 32(23):8261–8281, 2019.

[23] Enrique Castillo, Alberto Luceño, and Pablo Pedregal. Composition Functionals in Calculus of Variations: Application to Products and Quotients. *Mathematical Models and Methods in Applied Sciences*, 18(01):47–75, 2008. doi: 10.1142/S0218202508002607.

[24] Sudip Chakraborty, Bin Guan, Duane E. Waliser, Arlindo M. da Silva, Sophie Uluatam, and Peter Hess. Extending the Atmospheric River Concept to Aerosols: Climate and Air Quality Impacts. *Geophysical Research Letters*, 48(9):e2020GL091827, 2021. doi: https://doi.org/10.1029/2020GL091827. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020GL091827. e2020GL091827 2020GL091827.

[25] Edmund K. M. Chang. Projected Significant Increase in the Number of Extreme Extratropical Cyclones in the Southern Hemisphere. *Journal of Climate*, 30(13):4915 – 4935, 2017. doi: 10.1175/JCLI-D-16-0553.1. URL https://journals.ametsoc.org/view/journals/clim/30/13/jcli-d-16-0553.1.xml.

[26] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential equations. *CoRR*, abs/1806.07366, 2018. URL http://arxiv.org/abs/1806.07366.

[27] Isabelle Chiapello, G Bergametti, Bernadette Chatenet, Philippe Bousquet, François Dulac, and E Santos Soares. Origins of African dust transported over the northeastern tropical Atlantic. *Journal of Geophysical Research: Atmospheres*, 102(D12):13701–13709, 1997.

[28] Shirish Chinchalkar. The application of automatic differentiation to problems in engineering analysis. *Computer Methods in Applied Mechanics and Engineering*, 118(1): 197–207, 1994. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(94)90113-9. URL https://www.sciencedirect.com/science/article/pii/0045782594901139.

[29] OpenXLA Community. OpenXLA: GitHub Repository. https://github.com/openxla/xla, 2025.

[30] Rory Conlin. interpax, 2023. URL https://doi.org/10.5281/zenodo.10028967.

[31] C. Coulliette and S. Wiggins. Intergyre transport in a wind-driven, quasigeostrophic double gyre: an application of lobe dynamics. *Nonlinear Processes in Geophysics*, 8: 69–94, 2001.

[32] Jezabel Curbelo and Irina I. Rypina. A Three Dimensional Lagrangian Analysis of the Smoke Plume From the 2019/2020 Australian Wildfire Event. *Journal*

of Geophysical Research: Atmospheres, 128(21):e2023JD039773, 2023. doi: https://doi.org/10.1029/2023JD039773. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2023JD039773. e2023JD039773 2023JD039773.

[33] Jezabel Curbelo and Irina I. Rypina. A Three Dimensional Lagrangian Analysis of the Smoke Plume From the 2019/2020 Australian Wildfire Event. *Journal of Geophysical Research: Atmospheres*, 128(21):e2023JD039773, 2023. doi: 10.1029/2023JD039773. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2023JD039773. e2023JD039773 2023JD039773.

[34] John Dabiri. LCS MATLAB Kit, 2009. URL https://dabirilab.com/software.

[35] Andrew Dawson. Windspharm: A High-Level Library for Global Wind Field Computations Using Spherical Harmonics. *Journal of Open Research Software*, Aug 2016. doi: 10.5334/jors.129.

[36] M. Dellnitz, G. Froyland, C. Horenkamp, K. Padberg-Gehle, and A. S. Gupta. Seasonal variability of the subpolar gyres in the Southern Ocean: a numerical investigation based on transfer operators. *Nonlinear Processes in Geophysics*, 16:655–664, 2009.

[37] Philip C. Du Toit. *Transport and Separatrices in Time-Dependent Flows*. PhD thesis, California Institute of Technology, 2010. URL https://resolver.caltech.edu/CaltechTHESIS:10072009-165901284.

[38] Philip C. du Toit and Jerrold E. Marsden. Horseshoes in hurricanes. *Journal of Fixed Point Theory and Applications*, 7(2):351–384, Oct 2010. ISSN 1661-7746. doi: 10.1007/s11784-010-0028-6. URL https://doi.org/10.1007/s11784-010-0028-6.

[39] Lovely Euphrasie-Clotilde, Thomas Plocoste, and France-Nor Brute. Particle Size

Analysis of African Dust Haze over the Last 20 Years: A Focus on the Extreme Event of June 2020. *Atmosphere*, 12(4):502, 2021.

[40] Yu-Hsueh Fang, He-Zhe Lin, Jie-Jyun Liu, and Chih-Jen Lin. A Step-by-step Introduction to the Implementation of Automatic Differentiation, 2024. URL https://arxiv.org/abs/2402.16020.

[41] Mohammad Farazmand and George Haller. Computing Lagrangian coherent structures from their variational theory. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(1):013128, 03 2012. ISSN 1054-1500. doi: 10.1063/1.3690153.

[42] Mohammad Farazmand, Daniel Blazevski, and George Haller. Shearless transport barriers in unsteady two-dimensional flows and maps. *Physica D: Nonlinear Phenomena*, 278-279:44–57, 2014. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2014.03.008. URL https://www.sciencedirect.com/science/article/pii/S0167278914000712.

[43] Herbert Federer. *Geometric Measure Theory*, pages 207–340. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. ISBN 978-3-642-62010-2. doi: 10.1007/978-3-642-62010-2_4. URL https://doi.org/10.1007/978-3-642-62010-2_4.

[44] Diana Francis, Ricardo Fonseca, Narendra Nelli, Juan Cuesta, Michael Weston, Amato Evan, and Marouane Temimi. The Atmospheric Drivers of the Major Saharan Dust Storm in June 2020. *Geophysical Research Letters*, 47(24), 2020. doi: 10.1029/2020GL090102.

[45] G. Froyland, K. Padberg, M. H. England, and A. M. Treguier. Detection of Coherent Oceanic Structures via Transfer Operators. *Phys. Rev. Lett.*, 98:224503, 2007.

[46] G. Froyland, N. Santitissadeekorn, and A. Monahan. Optimally coherent sets in geophysical flows: A new approach to delimiting the stratospheric polar vortex. *Physical Review E*, 82:056311, 2010.

[47] Gary Froyland and Oliver Junge. Robust FEM-Based Extraction of Finite-Time Coherent Sets Using Scattered, Sparse, and Incomplete Trajectories. *SIAM Journal on Applied Dynamical Systems*, 17(2):1891–1924, 2018. doi: 10.1137/17M1129738. URL https://doi.org/10.1137/17M1129738.

[48] Gary Froyland, Naratip Santitissadeekorn, and Adam Monahan. Transport in time-dependent dynamical systems: Finite-time coherent sets. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(4):043116, 11 2010. ISSN 1054-1500. doi: 10.1063/1.3502450. URL https://doi.org/10.1063/1.3502450.

[49] Daniel Garaboa-Paz, Jorge Eiras-Barca, Florian Huhn, and Vicente Pérez-Muũuzuri. Lagrangian coherent structures along atmospheric rivers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(6):063105, 06 2015. ISSN 1054-1500. doi: 10.1063/1.4919768. URL https://doi.org/10.1063/1.4919768.

[50] Ronald Gelaro, Will McCarty, Max J. Suarez, Ricardo Todling, Andrea Molod, Lawrence Takacs, Cynthia A. Randles, Anton Darmenov, Michael G. Bosilovich, Rolf Reichle, Krzysztof Wargan, Lawrence Coy, Richard Cullather, Clara Draper, Santha Akella, Virginie Buchard, Austin Conaty, Arlindo M. da Silva, Wei Gu, Gi-Kong Kim, Randal Koster, Robert Lucchesi, Dagmar Merkova, Jon Eric Nielsen, Gary Partyka, Steven Pawson, William Putman, Michele Rienecker, Siegfried D. Schubert, Meta Sienkiewicz, and Bin Zhao. The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2). *Journal of Climate*, 30

(14):5419 – 5454, 2017. doi: https://doi.org/10.1175/JCLI-D-16-0758.1. URL https://journals.ametsoc.org/view/journals/clim/30/14/jcli-d-16-0758.1.xml.

[51] William Gilpin. Detecting coherent structures with automatic differentiation. In *APS Division of Fluid Dynamics Meeting Abstracts*, APS Meeting Abstracts, page N01.091, January 2021.

[52] Gregor Gläser, Heini Wernli, Astrid Kerkweg, and Franziska Teubler. The transatlantic dust transport from North Africa to the Americas—Its characteristics and source regions. *Journal of Geophysical Research: Atmospheres*, 120(21):11–231, 2015.

[53] Piyush Grover, Shane D Ross, Mark A Stremler, and Pankaj Kumar. Topological chaos, braiding and bifurcation of almost-cyclic sets. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):043135, 2012.

[54] Tobias Günther, Ákos Horváth, Wayne Bresky, Jaime Daniels, and Stefan Alexander Buehler. Lagrangian Coherent Structures and Vortex Formation in High Spatiotemporal-Resolution Satellite Winds of an Atmospheric Kármán Vortex Street. *Journal of Geophysical Research: Atmospheres*, 126(19):e2021JD035000, 2021. doi: 10.1029/2021JD035000. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021JD035000.

[55] Morton E. Gurtin, Eliot Fried, and Lallit Anand. *The Mechanics and Thermodynamics of Continua*. Cambridge University Press, 2010.

[56] J.K. Hale. *Ordinary Differential Equations*. Dover Publications, 2009.

[57] G. Haller. Finding finite-time invariant manifolds in two-dimensional velocity fields. *Chaos*, 10:99–108, 2000.

[58] G. Haller. Lagrangian coherent structures from approximate velocity data. *Physics of Fluids*, 14(6):1851–1861, 06 2002. ISSN 1070-6631. doi: 10.1063/1.1477449.

[59] G. Haller. A variational theory of hyperbolic Lagrangian Coherent Structures. *Physica D: Nonlinear Phenomena*, 240(7):574–598, 2011. ISSN 0167-2789. doi: 10.1016/j.physd.2010.11.010.

[60] G. Haller. *Transport Barriers and Coherent Structures in Flow Data: Advective, Diffusive, Stochastic and Active Methods*. Cambridge University Press, 2023.

[61] G. Haller and F. J. Beron-Vera. Coherent Lagrangian vortices: the black holes of turbulence. *Journal of Fluid Mechanics*, 731:R4, 2013.

[62] G. Haller and A.C. Poje. Finite time transport in aperiodic flows. *Physica D: Nonlinear Phenomena*, 119(3):352–380, 1998. ISSN 0167-2789. doi: 10.1016/S0167-2789(98)00091-8.

[63] G. Haller and G. Yuan. Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D*, 147:352–370, 2000.

[64] G. Haller, A. Hadjighasem, M. Farazmand, and F. Huhn. Defining coherent vortices objectively from the vorticity. *Journal of Fluid Mechanics*, 795:136–173, 2016. doi: 10.1017/jfm.2016.151.

[65] George Haller. Lagrangian Coherent Structures. *Annual Review of Fluid Mechanics*, 47(1):137–162, 2015. doi: 10.1146/annurev-fluid-010313-141322. URL https://doi.org/10.1146/annurev-fluid-010313-141322.

[66] George Haller. *Transport Barriers and Coherent Structures in Flow Data: Advective, Diffusive, Stochastic and Active Methods*. Cambridge University Press, 2023.

[67] George Haller and Francisco J. Beron-Vera. Geodesic theory of transport barriers in two-dimensional flows. *Physica D: Nonlinear Phenomena*, 241(20):1680–1702, 2012. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2012.06.012. URL https://www.sciencedirect.com/science/article/pii/S016727891200187X.

[68] George Haller, Daniel Karrasch, and Florian Kogelbauer. Material barriers to diffusive and stochastic transport. *Proceedings of the National Academy of Sciences*, 115(37): 9074–9079, 2018. doi: 10.1073/pnas.1720177115. URL https://www.pnas.org/doi/abs/10.1073/pnas.1720177115.

[69] George Haller, Stergios Katsanoulis, Markus Holzner, Bettina Frohnapfel, and Davide Gatti. Objective barriers to the transport of dynamically active vector fields. *Journal of Fluid Mechanics*, 905:A17, 2020. doi: 10.1017/jfm.2020.737.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[71] H. Hersbach, B. Bell, P. Berrisford, G. Biavati, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, I. Rozum, A. Schepers, D.and Simmons, C. Soci, D. Dee, and J-N. Thépaut. ERA5 hourly data on pressure levels from 1940 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS), 2023. Accessed on 13-04-2024.

[72] H. Hersbach, B. Bell, P. Berrisford, G. Biavati, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, I. Rozum, A. Schepers, D.and Simmons, C. Soci, D. Dee, and J-N. Thépaut. ERA5 hourly data on single levels from 1940 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS), 2023. Accessed on 13-04-2024.

[73] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[74] Albert Jarvis. NumbaCS, 2024. URL https://github.com/alb3rtjarvis/numbacs.

[75] Albert Jarvis, Ali Hossein Mardi, Hosein Foroutan, and Shane D. Ross. Atmospheric transport structures shaping the "Godzilla" dust plume. *Atmospheric Environment*, 333:120638, 2024. ISSN 1352-2310. doi: https://doi.org/10.1016/j.atmosenv.2024.120638. URL https://www.sciencedirect.com/science/article/pii/S1352231024003133.

[76] Emily G. Johnston and Azriel Rosenfeld. Digital Detection of Pits, Peaks, Ridges, and Ravines. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-5(4):472–480, 1975. doi: 10.1109/TSMC.1975.5408443.

[77] Binson Joseph and Bernard Legras. Relation between Kinematic Boundaries, Stirring, and Barriers for the Antarctic Polar Vortex. *J. Atmos. Sci.*, 59:1198–1212, 2002.

[78] Oliver Junge, Alvaro de Diego, Daniel Karrasch, and Nathanael Schilling. CoherentStructures.jl, 2020. URL https://github.com/CoherentStructures/CoherentStructures.jl.

[79] Daniel Karrasch. Comment on "A variational theory of hyperbolic Lagrangian coherent structures, Physica D 240 (2011) 574–598". *Physica D: Nonlinear Phenomena*, 241(17): 1470–1473, 2012. ISSN 0167-2789. doi: 10.1016/j.physd.2012.05.008.

[80] Daniel Karrasch and Nathanael Schilling. Fast and robust computation of coherent Lagrangian vortices on very large two-dimensional domains. *The SMAI Journal of computational mathematics*, 6:101–124, 2020. doi: 10.5802/smai-jcm.63. URL http://www.numdam.org/articles/10.5802/smai-jcm.63/.

[81] Daniel Karrasch, Florian Huhn, and George Haller. Automated detection of coherent Lagrangian vortices in two-dimensional unsteady flows. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2173):20140639, 2015. doi: 10.1098/rspa.2014.0639. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2014.0639.

[82] V Mohan Karyampudi, Stephen P Palm, John A Reagen, Hui Fang, William B Grant, Raymond M Hoff, Cyril Moulin, Harold F Pierce, Omar Torres, Edward V Browell, et al. Validation of the Saharan dust plume conceptual model using lidar, Meteosat, and ECMWF data. *Bulletin of the American Meteorological Society*, 80(6):1045–1076, 1999.

[83] Stergios Katsanoulis. BarrierTool, 2020. URL https://github.com/katsanoulis/BarrierTool.

[84] Arash Khatibi and Stefan Krauter. Validation and performance of satellite meteorological dataset MERRA-2 for solar and wind applications. *Energies*, 14(4):882, 2021.

[85] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[86] Robert C. Kirby. Algorithm 839: FIAT, a New Paradigm for Computing Finite Element Basis Functions. *ACM Transactions on Mathematical Software*, 30:502–516, 2004. doi: 10.1145/1039813.1039820.

[87] Robert C. Kirby. FIAT: Numerical Construction of Finite Element Basis Functions. In Anders Logg, Kent-Andre Mardal, and Garth N. Wells, editors, *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 13. Springer, 2012.

[88] Robert C. Kirby and Anders Logg. A Compiler for Variational Forms. *ACM Transactions on Mathematical Software*, 32, 2006. doi: 10.1145/1163641.1163644.

[89] Christian Lagares and Guillermo Araya. A GPU-Accelerated Particle Advection Methodology for 3D Lagrangian Coherent Structures in High-Speed Turbulent Boundary Layers. *Energies*, 16(12):4800, 2023. doi: 10.3390/en16124800.

[90] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: a LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340052. doi: 10.1145/2833157.2833162. URL https://doi.org/10.1145/2833157.2833162.

[91] F. Lekien and C. Coulliette. MANGEN: Computation of hyperbolic trajectories, invariant manifolds and lobes of dynamical systems defined as 2D+1 data sets. *in preparation*, 2003.

[92] F. Lekien and S. D. Ross. The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2, 2010.

[93] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 465–470, 1996. doi: 10.1109/CVPR.1996.517113.

[94] Qiantao Liu, Zhongwei Huang, Zhiyuan Hu, Qingqing Dong, and Shuting Li. Long-Range Transport and Evolution of Saharan Dust Over East Asia From 2007 to 2020. *Journal of Geophysical Research: Atmospheres*, 127(18):e2022JD036974, 2022. doi: https://doi.org/10.1029/2022JD036974. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022JD036974. e2022JD036974 2022JD036974.

[95] Yi Liu, Chris Wilson, Melissa A. Green, and Chris W. Hughes. Gulf Stream Transport and Mixing Processes via Coherent Structure Dynamics. *Journal of Geophysical Research: Oceans*, 123(4):3014–3037, 2018. doi: 10.1002/2017JC013390. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017JC013390.

[96] Anders Logg and Garth N. Wells. DOLFIN: Automated Finite Element Computing. *ACM Transactions on Mathematical Software*, 37, 2010. doi: 10.1145/1731022.1731030.

[97] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. doi: 10.1007/978-3-642-23099-8.

[98] Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. FFC: the FEniCS Form Compiler. In Anders Logg, Kent-Andre Mardal, and Garth N. Wells, editors, *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 11. Springer, 2012.

[99] Anders Logg, Garth N. Wells, and Johan Hake. DOLFIN: a C++/Python Finite Element Library. In Anders Logg, Kent-Andre Mardal, and Garth N. Wells, editors, *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 10. Springer, 2012.

[100] Carlos Lopesino, Francisco Balibrea-Iniesta, Víctor J García-Garrido, Stephen Wiggins, and Ana M Mancho. A theoretical framework for Lagrangian descriptors. *International Journal of Bifurcation and Chaos*, 27(01):1730001, 2017.

[101] Tian Ma and Erik M. Bollt. Differential Geometry Perspective of Shape Coherence and Curvature Evolution by Finite-Time Nonhyperbolic Splitting. *SIAM Journal on Applied Dynamical Systems*, 13(3):1106–1136, 2014.

[102] R. S. MacKay, J. D. Meiss, and I. C. Percival. Transport in Hamiltonian systems. *Physica D*, 13:55–81, 1984.

[103] Ana M. Mancho, Des Small, and Stephen Wiggins. A tutorial on dynamical systems concepts applied to Lagrangian transport in oceanic flows defined as finite time data sets: Theoretical and computational issues. *Physics Reports*, 437(3):55 – 124, 2006. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2006.09.005. URL http://www.sciencedirect.com/science/article/pii/S0370157306003401.

[104] Ali Hossein Mardi, Miguel Ricardo Hilario, Regina Hanlon, Cristina González-Martín, David Schmale, Armin Sorooshian, and Hosein Foroutan. Long-Term Seasonal Trends in Sources and Pathways of Trans-Atlantic Dust Plumes and their Implications for Transport of Microorganisms. March 2023. doi: 10.22541/essoar.167979593.35998646/v1. URL http://dx.doi.org/10.22541/essoar.167979593.35998646/v1.

[105] M. Mathur, G. Haller, T. Peacock, J. E. Ruppert-Felsot, and H. L. Swinney. Uncovering the Lagrangian skeleton of turbulence. *Physical Review Letters*, 98:144502, 2007.

[106] J. D. Meiss. Symplectic maps, variational principles, and transport. *Rev. Mod. Phys.*, 64:795–848, 1992.

[107] Changhong Mou, Zhu Wang, David R. Wells, Xuping Xie, and Traian Iliescu. Reduced Order Models for the Quasi-Geostrophic Equations: A Brief Survey. *Fluids*, 6(1), 2021. ISSN 2311-5521. doi: 10.3390/fluids6010016. URL https://www.mdpi.com/2311-5521/6/1/16.

[108] Shibabrat Naik, Francois Lekien, and Shane D. Ross. Computational method for phase space transport with applications to lobe dynamics and rate of escape. *Regular and Chaotic Dynamics*, 22(3):272–297, May 2017. ISSN 1468-4845. doi: 10.1134/S1560354717030078. URL https://doi.org/10.1134/S1560354717030078.

[109] Paul K. Newton. *The N-Vortex Problem: Analytical Techniques*, volume 145 of *Appl. Math. Sci. Series*. Springer-Verlag, Berlin-Heidelberg-New York, 2001.

[110] Peter Nolan. Dynlab, 2024. URL https://github.com/hokiepete/dynlab.

[111] Peter Nolan, Hosein Foroutan, and Shane D. Ross. Pollution Transport Patterns Obtained Through Generalized Lagrangian Coherent Structures. *Atmosphere*, 11(2), 2020. ISSN 2073-4433. doi: 10.3390/atmos11020168. URL https://www.mdpi.com/2073-4433/11/2/168.

[112] Peter Nolan, Mattia Serra, and Shane D. Ross. Finite-time Lyapunov exponents in the instantaneous limit and material transport. *Nonlinear Dyn*, 100:3825–3852, 2020. doi: 10.1007/s11071-020-05713-4.

[113] M. J. Olascoaga, F. J. Beron-Vera, G. Haller, J. Triñanes, M. Iskandarani, E. F. Coelho, B. K. Haus, H. S. Huntley, G. Jacobs, A. D. Kirwan Jr., B. L. Lipphardt Jr., T. M. Özgökmen, A. J. H. M. Reniers, and A. Valle-Levinson. Drifter motion in the Gulf of Mexico constrained by altimetric Lagrangian coherent structures. *Geophysical Research Letters*, 40(23):6171–6175, 2013. doi: https://doi.org/10.1002/2013GL058624. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2013GL058624.

[114] María J. Olascoaga and George Haller. Forecasting sudden changes in environmental pollution patterns. *Proceedings of the National Academy of Sciences*, 109(13):4738–4743, 2012. doi: 10.1073/pnas.1118574109. URL https://www.pnas.org/doi/abs/10.1073/pnas.1118574109.

[115] Kristian B. Ølgaard and Garth N. Wells. Optimisations for Quadrature Representations of Finite Element Tensors Through Automated Code Generation. *ACM Transactions on Mathematical Software*, 37, 2010. doi: 10.1145/1644001.1644009.

[116] K. Onu, F. Huhn, and G. Haller. LCS Tool: A computational platform for Lagrangian coherent structures. *Journal of Computational Science*, 7:26–36, 2015. ISSN 1877-7503. doi: 10.1016/j.jocs.2014.12.002. URL https://www.sciencedirect.com/science/article/pii/S187775031400163X.

[117] Thomas Peacock and George Haller. Lagrangian coherent structures: The hidden skeleton of fluid flows. *Physics Today*, 66(2):41–47, 02 2013. ISSN 0031-9228. doi: 10.1063/PT.3.1886. URL https://doi.org/10.1063/PT.3.1886.

[118] Ronald Peikert and Filip Sadlo. Height Ridge Computation and Filtering for Visualization. In *2008 IEEE Pacific Visualization Symposium*, pages 119–126, 2008. doi: 10.1109/PACIFICVIS.2008.4475467.

[119] Jifeng Peng and Rorik Peterson. Attracting structures in volcanic ash transport. *Atmospheric Environment*, 48:230–239, 2012.

[120] R. T. Pierrehumbert. Chaotic mixing of tracer and vorticity by modulated traveling Rossby waves. *Geophys. Astrophys. Fluid Dyn.*, 58:285–319, 1991.

[121] R. T. Pierrehumbert. Large-scale horizontal mixing in planetary atmospheres. *Phys. Fluids A*, 3:1250–1260, 1991.

[122] Henri Poincaré. Sur la problème des trois corps et les équations de la dynamique. *Acta Math.*, 13:1–271, 1890.

[123] Henri Poincaré. *Les Méthodes Nouvelles de la Mécanique Céleste (New Methods in Celestial Mechanics)*, volume 1-3. Gauthier-Villars, Paris, 1892-1899. Reprinted by Dover, New York, 1957.

[124] Ilze Pretorius, Wayne C. Schou, Brian Richardson, Shane D. Ross, Toni M. Withers, David G. Schmale III, and Tara M. Strand. In the wind: Invasive species travel

along predictable atmospheric pathways. *Ecological Applications*, 33(3):e2806, 2023. doi: https://doi.org/10.1002/eap.2806. URL https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1002/eap.2806.

[125] P. J Prince and J. R. Dormand. High order embedded Runge–Kutta formulae. *J. Comp. Appl. Math*, 7(1):67–75, 1981.

[126] B. Pu and Q. Jin. A Record-Breaking Trans-Atlantic African Dust Plume Associated with Atmospheric Circulation Extremes in June 2020. *Bulletin of the American Meteorological Society*, 102(7):E1340 – E1356, 2021. doi: https://doi.org/10.1175/BAMS-D-21-0014.1. URL https://journals.ametsoc.org/view/journals/bams/102/7/BAMS-D-21-0014.1.xml.

[127] B Remini. Awesome, the dust of the Sahara in the sky of the America continent Godzilla, the biggest dust storm in half a century. *LARHYSS Journal P-ISSN 1112-3680/E-ISSN 2521-9782*, (43):139–167, 2020.

[128] S. D. Ross and P. Tallapragada. Detecting and exploiting chaotic transport in mechanical systems. In S. Banerjee, L. Rondoni, and M. Mitra, editors, *Applications of Chaos and Nonlinear Dynamics in Science and Engineering*, volume 2, pages 155–183. Springer, New York, 2012.

[129] B. Rutherford, G. Dangelmayr, and M. T. Montgomery. Lagrangian coherent structures in tropical cyclone intensification. *Atmospheric Chemistry and Physics*, 12(12): 5483–5507, 2012. doi: 10.5194/acp-12-5483-2012. URL https://acp.copernicus.org/articles/12/5483/2012/.

[130] I. I. Rypina, M. G. Brown, F. J. Beron-Vera, H. Kocak, M. J. Olascoaga, and I. A. Udovydchenkov. On the Lagrangian dynamics of atmospheric zonal jets and the permeability of the stratospheric polar vortex. *J. Atmos. Sci.*, 64:3595–3610, 2007.

[131] Filip Sadlo and Daniel Weiskopf. Time-Dependent 2-D Vector Field Topology: An Approach Inspired by Lagrangian Coherent Structures. *Computer Graphics Forum*, 29, 2010. URL https://api.semanticscholar.org/CorpusID:428227.

[132] N Santitissadeekorn, G Froyland, and A Monahan. Optimally coherent sets in geophysical flows: a transfer-operator approach to delimiting the stratospheric polar vortex. *Phys Rev E Stat Nonlin Soft Matter Phys*, 82(5 Pt 2):056311 – ?, 2010. ISSN 1550-2376.

[133] Themistoklis Sapsis and George Haller. Inertial Particle Dynamics in a Hurricane. *Journal of the Atmospheric Sciences*, 66(8):2481 – 2492, 2009. doi: https://doi.org/10.1175/2009JAS2865.1. URL https://journals.ametsoc.org/view/journals/atsc/66/8/2009jas2865.1.xml.

[134] Benjamin Schindler, Ronald Peikert, Raphael Fuchs, and Holger Theisel. *Ridge Concepts for the Visualization of Lagrangian Coherent Structures*, pages 221–235. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-23175-9. doi: 10.1007/978-3-642-23175-9_15.

[135] David G. Schmale and Shane D. Ross. Highways in the Sky: Scales of Atmospheric Transport of Plant Pathogens. *Annual Review of Phytopathology*, 53(1):591–611, 2015.

[136] David G. Schmale and Shane D. Ross. High-flying microbes: Aerial drones and chaos theory help researchers explore the many ways that microorganisms spread havoc around the world. *Scientific American*, February:32–37, 2017.

[137] DG Schmale III, SD Ross, TL Fetters, P Tallapragada, AK Wood-Jones, and B Dingus. Isolates of Fusarium graminearum collected 40–320 meters above ground level cause Fusarium head blight in wheat and produce trichothecene mycotoxins. *Aerobiologia*, 28(1):1–11, 2012.

[138] Matthew W. Scroggs, Igor A. Baratta, Chris N. Richardson, and Garth N. Wells. Basix: a runtime finite element basis evaluation library. *Journal of Open Source Software*, 7 (73):3982, 2022. doi: 10.21105/joss.03982.

[139] Matthew W. Scroggs, Jørgen S. Dokken, Chris N. Richardson, and Garth N. Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software*, 48(2):18:1–18:23, 2022. doi: 10.1145/3524456.

[140] Carmine Senatore and Shane D. Ross. Detection and characterization of transport barriers in complex flows via ridge extraction of the finite time Lyapunov exponent field. *International Journal for Numerical Methods in Engineering*, 86(9):1163–1174, 2011. doi: 10.1002/nme.3101.

[141] Mattia Serra and George Haller. Objective Eulerian coherent structures. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(5):053110, 2016.

[142] Mattia Serra and George Haller. Efficient computation of null geodesics with applications to coherent vortex detection. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2199):20160807, 2017. doi: 10.1098/rspa.2016.0807. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2016.0807.

[143] Mattia Serra and George Haller. Forecasting long-lived Lagrangian vortices from their objective Eulerian footprints. *Journal of Fluid Mechanics*, 813:436–457, 2017. doi: 10.1017/jfm.2016.865.

[144] Mattia Serra, Pratik Sathe, Francisco Beron-Vera, and George Haller. Uncovering the Edge of the Polar Vortex. *Journal of the Atmospheric Sciences*, 74(11):3871 – 3885, 2017. doi: 10.1175/JAS-D-17-0052.1.

[145] Mattia Serra, Pratik Sathe, Irina Rypina, Anthony Kirincich, Shane D. Ross, Pierre Lermusiaux, Arthur Allen, Thomas Peacock, and George Haller. Search and rescue at sea aided by hidden flow structures. *Nature Communications*, 11(1), May 2020. doi: 10.1038/s41467-020-16281-x.

[146] Mattia Serra, Pratik Sathe, Irina Rypina, Anthony Kirincich, Shane D Ross, Pierre Lermusiaux, Arthur Allen, Thomas Peacock, and George Haller. Search and rescue at sea aided by hidden flow structures. *Nature Communications*, 11(1):2525, 2020.

[147] S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of Lagrangian coherent structures: mixing and transport in two-dimensional aperiodic flows. *Physica D*, 212:271–304, 2005.

[148] C. Steger. An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):113–125, 1998. doi: 10.1109/34.659930.

[149] A. F. Stein, R. R. Draxler, G. D. Rolph, B. J. B. Stunder, M. D. Cohen, and F. Ngan. NOAA's HYSPLIT Atmospheric Transport and Dispersion Modeling System. *Bulletin of the American Meteorological Society*, 96(12):2059 – 2077, 2015. doi: 10.1175/BAMS-D-14-00110.1. URL https://journals.ametsoc.org/view/journals/bams/96/12/bams-d-14-00110.1.xml.

[150] M. A. Stremler, S. D. Ross, P. Grover, and P. Kumar. Topological chaos and periodic braiding of almost-cyclic sets. *Physical Review Letters*, 106:114101, 2011.

[151] Steven H Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC press, second edition, 2014.

[152] Mathalai Balan Sudharsan, Steven L. Brunton, and James J. Riley. Lagrangian coherent structures and inertial particle dynamics. *Physical review. E*, 93 3:033108, 2015. URL https://api.semanticscholar.org/CorpusID:32179502.

[153] P. Tallapragada and S. D. Ross. Particle segregation by Stokes number for small neutrally buoyant spheres in a fluid. *Physical Review E*, 78:036308, 2008.

[154] P. Tallapragada, S. D. Ross, and D. Schmale. Lagrangian coherent structures are associated with fluctuations in airborne microbial populations. *Chaos*, 21:033122, 2011.

[155] Phanindra Tallapragada and Shane D. Ross. A set oriented definition of finite-time Lyapunov exponents and coherent sets. *Communications in Nonlinear Science and Numerical Simulation*, 18(5):1106 – 1126, 2013.

[156] O. Torres. OMPS-NPP L2 NM aerosol index swath orbital V2. *Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), doi*, 10: 89, 2019.

[157] Ch Tsitouras. Runge–kutta pairs of order 5 (4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62(2):770–775, 2011.

[158] Jack Tyler and Alexander Wittig. An improved numerical method for hyperbolic Lagrangian Coherent Structures using Differential Algebra. *Journal of Computational Science*, 65:101883, 2022. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2022.101883. URL https://www.sciencedirect.com/science/article/pii/S1877750322002423.

[159] Erin Walker, Daniel Mitchell, and William Seviour. The numerous approaches to tracking extratropical cyclones and the challenges they present. *Weather*, 75(11):336–341,

2020. doi: https://doi.org/10.1002/wea.3861. URL https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/wea.3861.

[160] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, volume 2 of *Texts in Applied Mathematics Science*. Springer-Verlag, Berlin, 2nd edition, 2003.

[161] Pablo Winant, Chase Coleman, and Spencer Lyon. interpolation.py, 2017. URL https://github.com/EconForge/interpolation.py.

[162] Nicholas Wogan. numbalsoda, 2021. URL https://github.com/Nicholaswogan/numbalsoda.

[163] Kai Yang, Russell R Dickerson, Simon A Carn, Cui Ge, and Jun Wang. First observations of SO2 from the satellite Suomi NPP OMPS: Widespread air pollution events over China. *Geophysical Research Letters*, 40(18):4957–4962, 2013.

[164] Hongbin Yu, Lorraine A. Remer, Ralph A. Kahn, Mian Chin, and Yan Zhang. Satellite perspective of aerosol intercontinental transport: From qualitative tracking to quantitative characterization. *Atmospheric Research*, 124:73–100, 2013. ISSN 0169-8095. doi: https://doi.org/10.1016/j.atmosres.2012.12.013. URL https://www.sciencedirect.com/science/article/pii/S0169809513000070.

[165] Hongbin Yu, Mian Chin, Huisheng Bian, Tianle Yuan, Joseph M. Prospero, Ali H. Omar, Lorraine A. Remer, David M. Winker, Yuekui Yang, Yan Zhang, and Zhibo Zhang. Quantification of trans-Atlantic dust transport from seven-year (2007–2013) record of CALIPSO lidar measurements. *Remote Sensing of Environment*, 159:232–249, 2015. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2014.12.010. URL https://www.sciencedirect.com/science/article/pii/S0034425714005021.

[166] Hongbin Yu, Qian Tan, Lillian Zhou, Yaping Zhou, Huisheng Bian, Mian Chin, Claire L

Ryder, Robert C Levy, Yaswant Pradhan, Yingxi Shi, et al. Observation and modeling of the historic "Godzilla" African dust intrusion into the Caribbean Basin and the southern US in June 2020. *Atmospheric Chemistry and Physics*, 21(16):12359–12383, 2021.

Appendices

Appendix A

Supplemental Material: Atmospheric transport structures shaping the "Godzilla" dust storm

A.1 Implications of Column-averaged velocity fields

In the text, we mainly focus on FTLE fields derived from column-averaged velocity fields. These velocity fields are averaged from pressure surfaces between 500 hPa - 800 hPa. This is done for a number of reasons. The main reason we do this is because we are comparing with column-averaged aerosol index data and wanted to make an attempt to capture the influence of the wind at all levels at which dust was present. An additional reason we do this is because, most of these pressure surfaces intersect with the ground at some point and velocity data is not provided where this happens. While these regions were often not in the area we were focused on, we were performing particle integrations starting from a grid within the region we are focused on and therefore, these particle paths sometimes enter these regions of ground interference. Using the averaged velocity field allows us to have a velocity field that is defined all over the globe since averaging at a given point only includes those levels for which we have data. Alternatively, when using a single pressure surface with missing data, one is forced to find some way to deal with these areas. Since there is ground

interference we just set the velocity to 0 at these points so when particles enter this area they stop right where they enter. This is what we did for the 600 hPa pressure surface and it resulted in some artificial looking areas because of it.

By performing this averaging, we are no longer solving our system over a well defined manifold. This is in some sense a mathematical formality and can be avoided by imposing a manifold over which our new vector field is defined (say at the average pressure of pressure surfaces used, 650 hPa). In addition, since we are using velocity fields from the atmosphere on a pressure surface, they are essentially incompressible and are treated as such. It is possible to introduce non-negligible compressibility into our averaged vector field by way of the averaging. We checked for this and found that the average divergence over all grid points and all times used in our calculations was 4.60×10^{-8} s^{-1} for the averaged fields and -2.01×10^{-8} s^{-1} for a single pressure surface (600h Pa). The maximum divergence (in magnitude) over all grid points and times was 2.90×10^{-4} s^{-1} for the averaged fields and 4.90×10^{-4} s^{-1} for the 600 hPa velocity fields. Therefore, we conclude that the averaged fields are roughly as compressible as a single pressure surface and this amount is insignificant. While all these concerns could be quelled in our case, it is wise to check these things if taking an approach like this.

A.2 Integration Time

As mentioned, the integration time chosen should be tied to some characteristic time scale of the transport. We settled on approximately half the time it took for the plume to traverse the Atlantic (i.e., half of $\sim$ 8 days) as we were interested in structures that influenced the transport of the plume on this time scale. Often, there is more than one integration time that can be chosen that yields satisfactory results. In Figure A.1 below, we show the effect

of integration time on the resulting FTLE field. In general, as can be seen in the figure, shorter integration time magnitudes $|T|$ reveal an FTLE field that is broader, with fewer—and shorter—ridges. In fact, the FTLE field shows "instantaneous" structure in the $|T| \to 0$ limit, as discussed in [112], which give the short-time attracting ($T \to 0^-$) or repelling ($T \to 0^+$) structures. As the integration time magnitude $|T|$ is increased, the FTLE field is sharper, with more—and longer—ridges. For our application, picking a shorter time such as $T = -1$, or -2 days ((a) and (b) from Figure A.1) could potentially miss important structures and produce less well-defined ridges. Choosing a much longer time such as $T = -5$, -6, -7, or -8 days ((e), (f), (g), and (h) from Figure A.1) results in an overly complicated FTLE field. Choosing an integration time of $T = -3$ or -4 days ((c) and (d) from Figure A.1) yields results that capture all structures of importance while not being overly complicated. Picking $T = -3$ days would have been reasonable for the entirety of this paper and in some applications, choosing between candidate integration times comes down to user decision when a single time is not obvious. One could argue that choosing a shorter time is advantageous as this results in a cheaper computation, yet at the risk of missing out on potentially important small scale features. This computational advantage continues when performing ridge or LCS extraction, as shorter integration times result in fewer ridges.

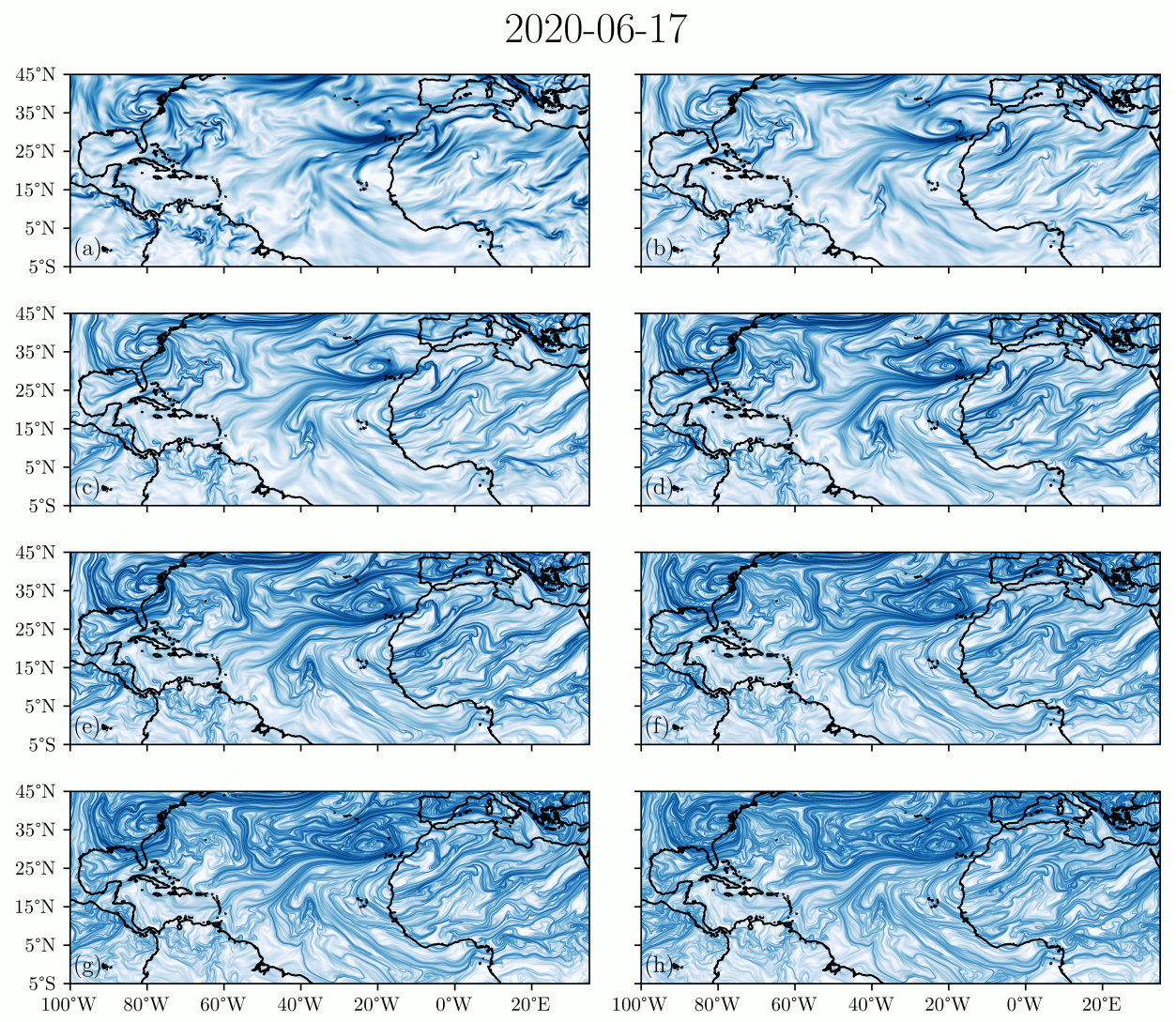

Figure A.1: Backward FTLE field from column-averaged velocity fields on June 17, 2020 for integration times T = (a) -1 day, (b) -2 days, (c) -3 days, (d) -4 days, (e) -5 days, (f) -6 days, (g) -7 days, (h) -8 days,

A.3 600 hPa

In this section, we provide figures that were used in the analysis of the dust storm from the 3.1 (FTLE) and 3.2 (Eulerian Combined with Lagrangian Analysis - Early June and Mid

June Vortex Comparison) sections of the main text but instead with all of the calculations and comparisons performed on a single pressure surface (600 hPa) instead of the averaged vector field we mainly focused on. No further analysis is presented as the same conclusions are drawn regardless of which underlying vector field was used.

3.1 - FTLE

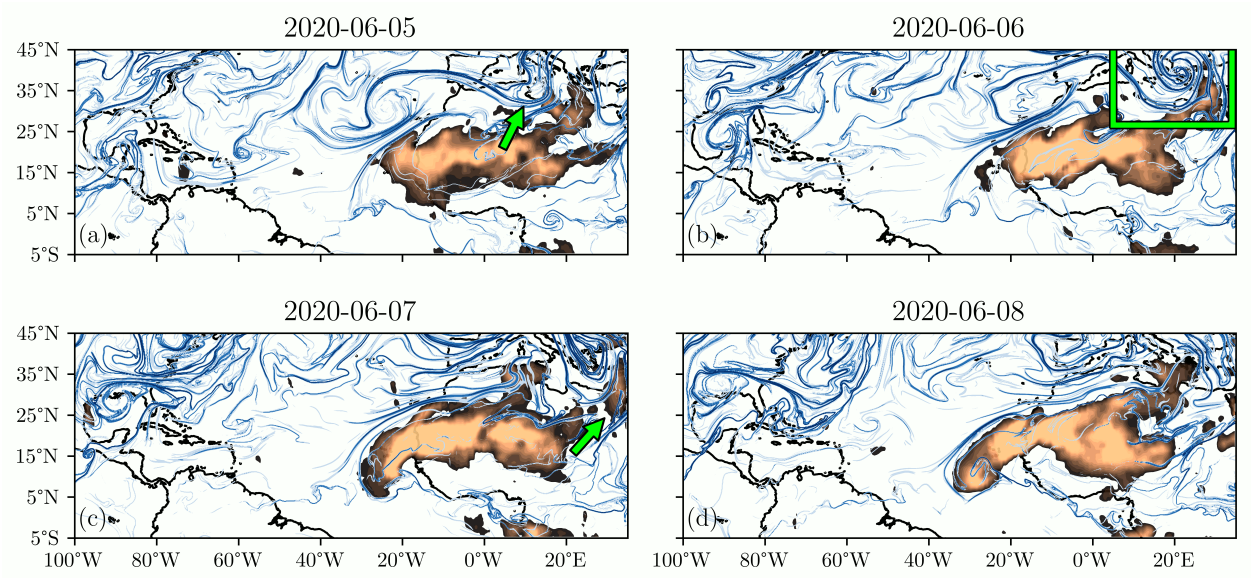

Figure A.2: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 5-8, 2020. See text for details.

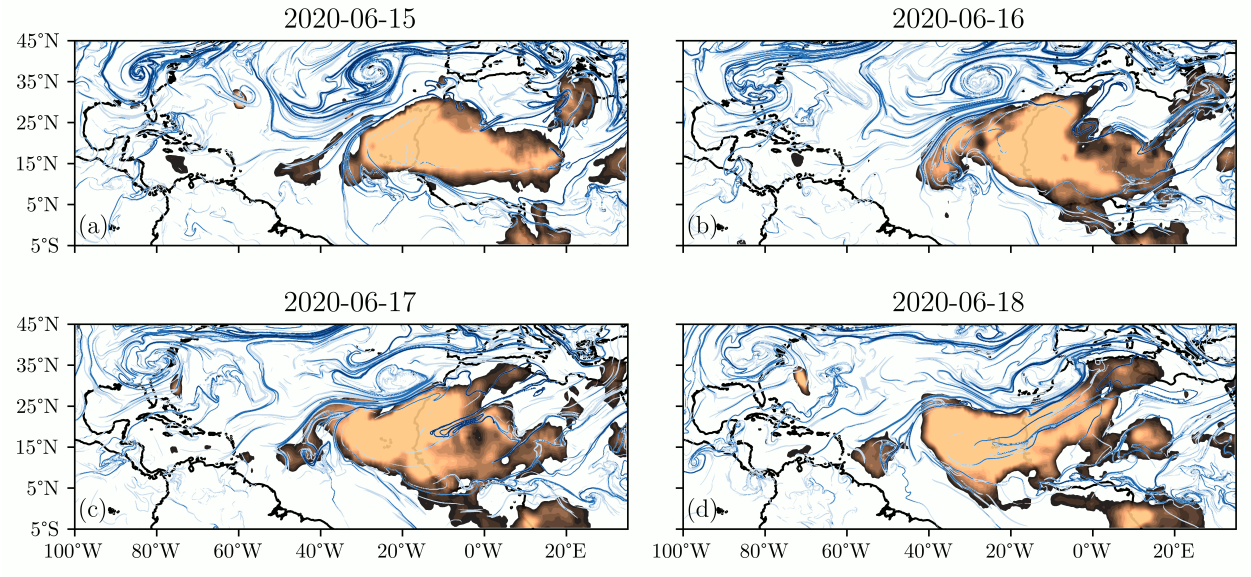

Figure A.3: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 15-18, 2020.

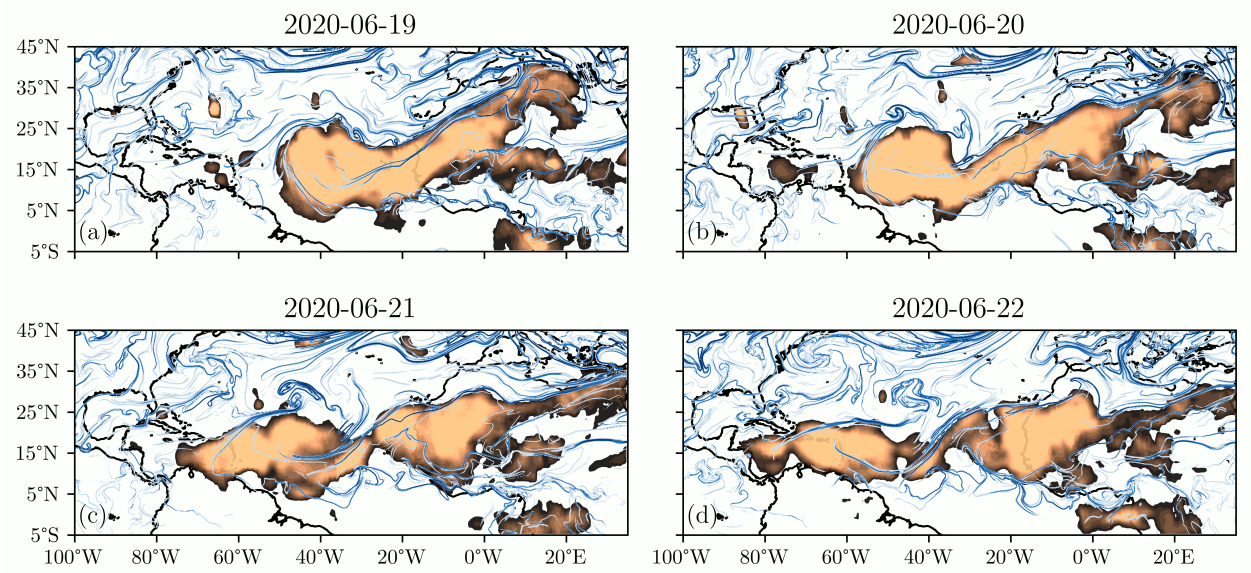

Figure A.4: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 19-22, 2020.

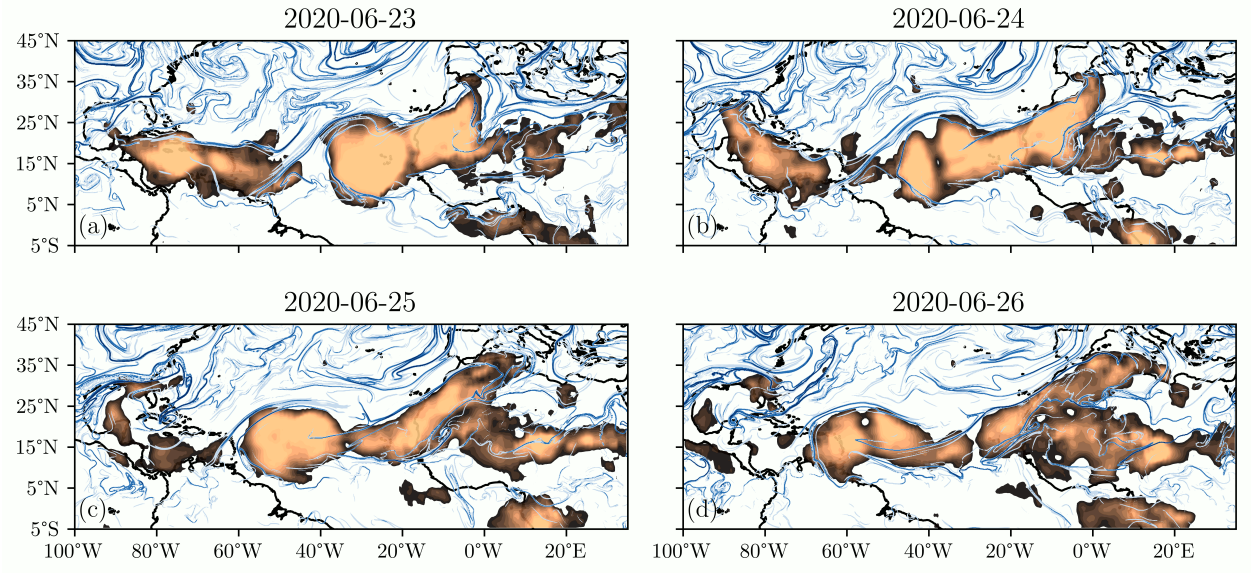

Figure A.5: Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 23-26, 2020.

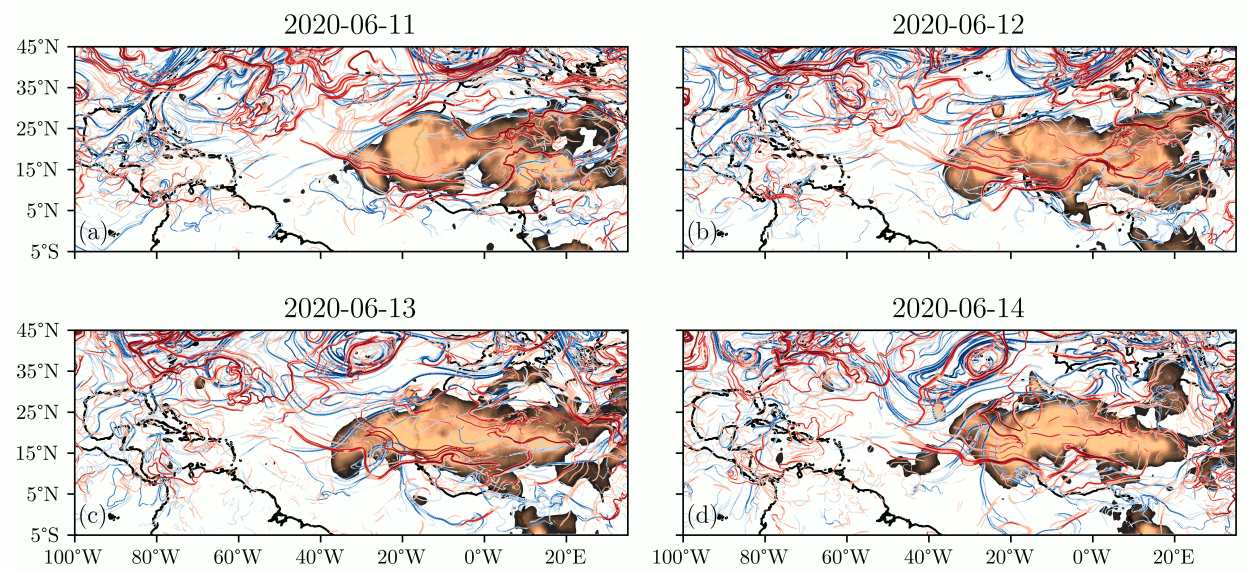

Figure A.6: Forward and Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 11-14, 2020.

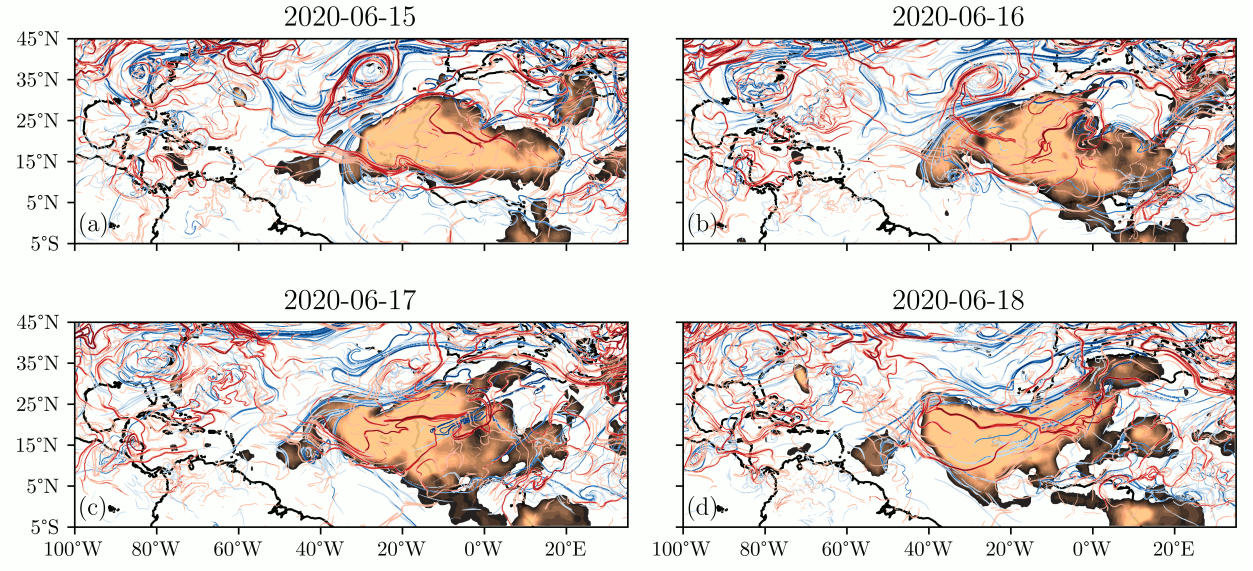

Figure A.7: Forward and Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 15-18, 2020.

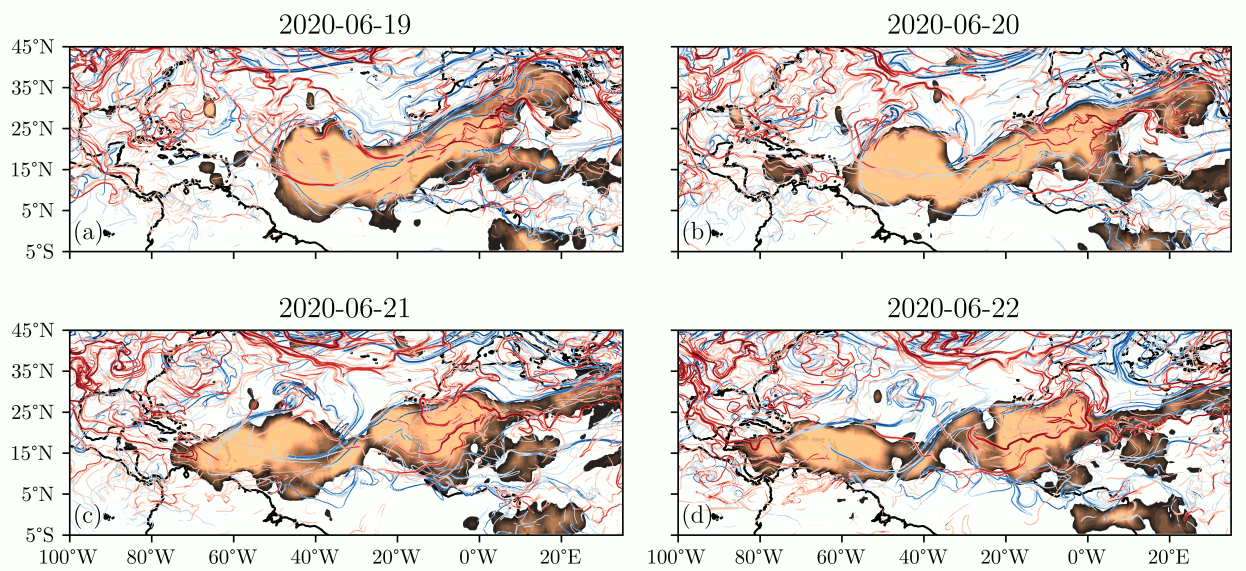

Figure A.8: Forward and Backward FTLE ridges overlaid on aerosol index data obtained from OMPS, June 19-22, 2020. See text for details.

3.2 - Eulerian Combined with Lagrangian Analysis

Early June and Mid June Vortex Comparison

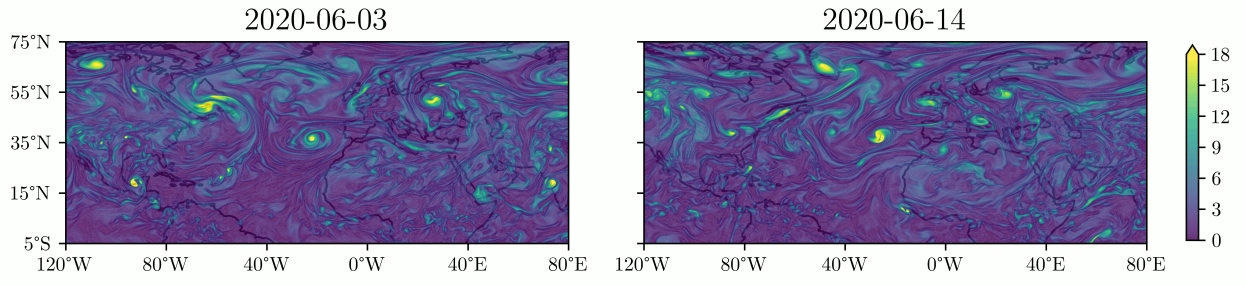

Figure A.9: Backward LAVD ($10^{-5}\times s^{-1}$, integration time = 1 day) on June 3, 2020 (left) and June 14, 2020 (right).

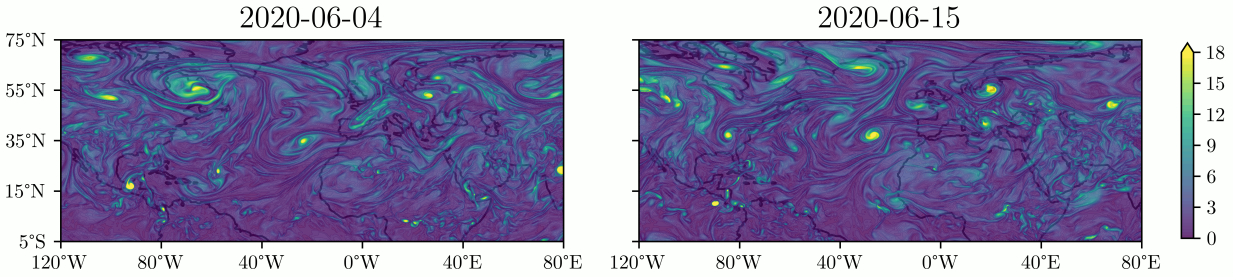

Figure A.10: Backward LAVD ($10^{-5}\times s^{-1}$, integration time = 1 day) on June 4, 2020 (left) and June 15, 2020 (right).

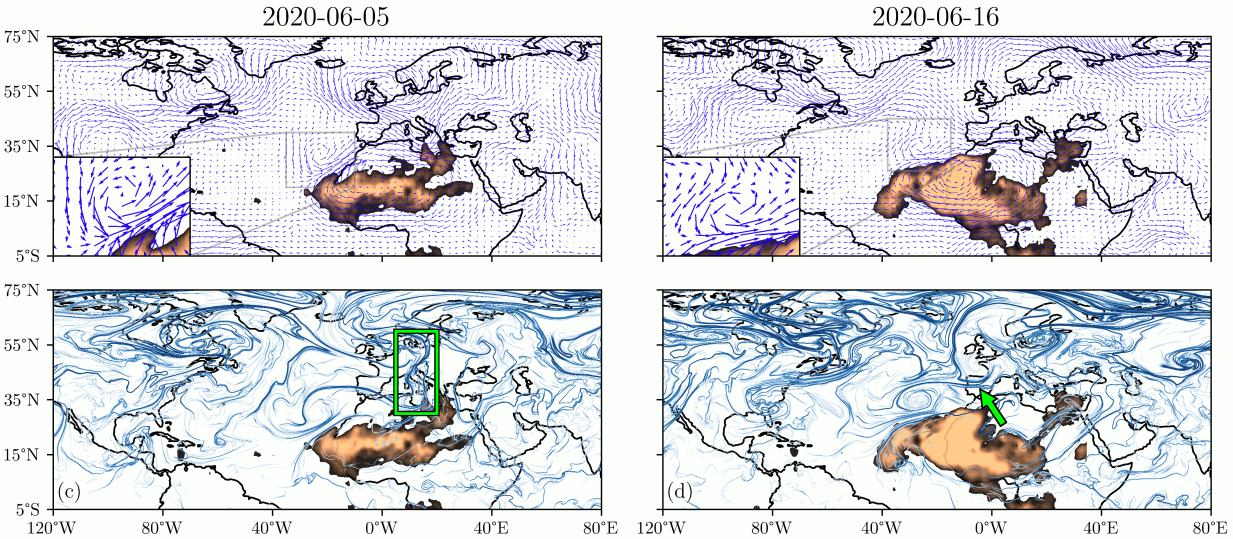

Figure A.11: Velocity field (top) and backward FTLE ridges (bottom) overlaid on OMPS aerosol index data on June 5, 2020 (left) and June 16, 2020 (right). See text for details.

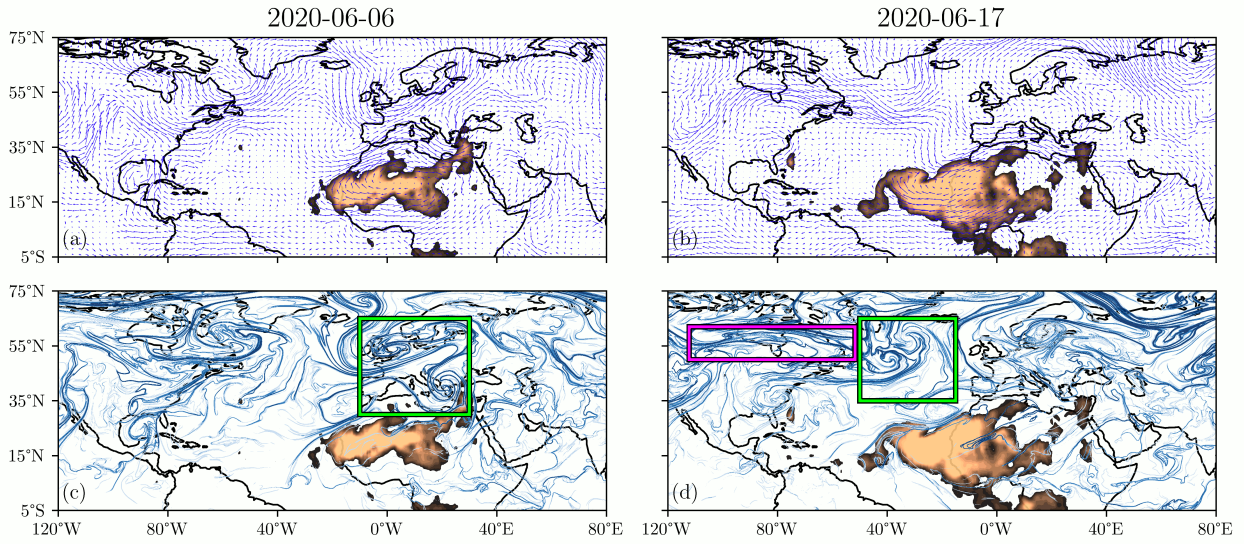

Figure A.12: Velocity field (top) and backward FTLE ridges (bottom) overlaid on OMPS aerosol index data on June 6, 2020 (left) and June 17, 2020 (right). See text for details.

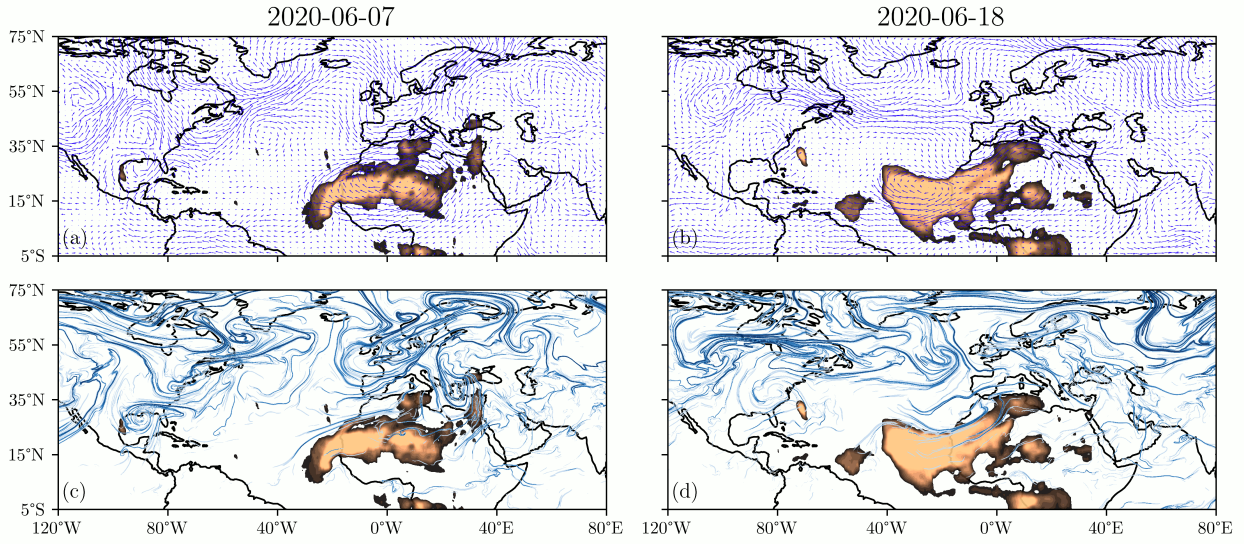

Figure A.13: Velocity field (top) and backward FTLE ridges (bottom) overlaid on OMPS aerosol index data on June 7, 2020 (left) and June 18, 2020 (right).

Appendix B

Appendices: Diffusive Flux-Rate Barriers for General Diffusion Structure

B.1 Time derivative of $\bar{\mathbf{T}}_{t_0}^t$ as $t \to t_0$

Recall

$$\bar{\mathbf{T}}_{t_0}^t = \frac{1}{t-t_0}\int_{t_0}^t \mathbf{T}_{t_0}^s ds \tag{B.1}$$

Then we take a derivative w.r.t. t

$$\begin{aligned}\frac{d}{dt}\bar{\mathbf{T}}_{t_0}^t &= \frac{d}{dt}\left(\frac{1}{t-t_0}\int_{t_0}^t \mathbf{T}_{t_0}^s ds\right)\\ &= \frac{-1}{(t-t_0)^2}\int_{t_0}^t \mathbf{T}_{t_0}^s ds + \frac{1}{t-t_0}\mathbf{T}_{t_0}^t \\ &= \frac{\mathbf{T}_{t_0}^t - \bar{\mathbf{T}}_{t_0}^t}{t-t_0}\end{aligned} \tag{B.2}$$

Dropping the t_0 subscript and writing in index notation we have,

$$\left[\frac{d}{dt}\bar{\mathbf{T}}^t_{t_0}\right]_{ij} = \frac{T^t_{ij} - \bar{T}^t_{ij}}{t - t_0} \tag{B.3}$$

By the mean value theorem for integrals, $\forall i, j \in \{1, ..., n\} \times \{1, ..., n\}$ $\exists t^* \in [t_0, t]$ such that $T^{t^*}_{ij} = \bar{T}^t_{ij}$ (t^* may be different for each i, j). Therefore,

$$\begin{aligned} \frac{T^t_{ij} - \bar{T}^t_{ij}}{t - t_0} &= \frac{T^t_{ij} - T^{t^*}_{ij}}{t - t_0} \\ &= \frac{T^t_{ij} - T^{t^*}_{ij}}{t - t^*}\frac{t - t^*}{t - t_0} \end{aligned} \tag{B.4}$$

Now we can take a limit as $t \to t_0$ (this implies $t^* \to t_0$ for each i, j)

$$\begin{aligned} \lim_{t\to t_0}\left[\frac{d}{dt}\bar{\mathbf{T}}^t_{t_0}\right]_{ij} &= \lim_{t\to t_0}\left(\frac{T^t_{ij} - T^{t^*}_{ij}}{t - t^*}\frac{t - t^*}{t - t_0}\right) \\ &= \lim_{t\to t_0}\left(\frac{T^t_{ij} - T^{t^*}_{ij}}{t - t^*}\right)\lim_{t\to t_0}\left(\frac{t - t^*}{t - t_0}\right) \\ &= \dot{T}_{ij} \\ &= \left[\dot{\mathbf{T}}_{t_0}\right]_{ij} \end{aligned} \tag{B.5}$$

where we use the fact that limits of products are products of limits when the limit of each factor exists. Therefore,

$$\lim_{t\to t_0}\frac{d}{dt}\bar{\mathbf{T}}^t_{t_0} = \dot{\mathbf{T}}_{t_0} \tag{B.6}$$

B.2 Taylor expansion of $\mathbf{T}_{t_0}^t$ for general D

For the general case of the diffusion tensor when $\mathbf{D} = \mathbf{D}(\mathbf{F}_{t_0}^t, t)$, the diffusive flux tensor is given by

$$\mathbf{T}_{t_0}^t(\mathbf{x}_0) = \left[\nabla_0 \mathbf{F}_{t_0}^t(\mathbf{x}_0)\right]^{-1} \mathbf{D}(\mathbf{F}_{t_0}^t(\mathbf{x}_0), t) \left[\nabla_0 \mathbf{F}_{t_0}^t(\mathbf{x}_0)\right]^{-\top}, \tag{B.7}$$

where $t = t_0 + \tau$. Now we aim to expand $\mathbf{T}$ through first order in τ. First we must compute the first derivative w.r.t. τ,

$$\begin{aligned}
\frac{d\mathbf{T}}{d\tau} &= \frac{d(\nabla\mathbf{F})^{-1}}{d\tau}\mathbf{D}(\nabla\mathbf{F})^{-\top} + (\nabla\mathbf{F})^{-1}\frac{d\mathbf{D}}{d\tau}(\nabla\mathbf{F})^{-\top} + (\nabla\mathbf{F})^{-1}\mathbf{D}\frac{d(\nabla\mathbf{F})^{-\top}}{d\tau} \\
&= -(\nabla\mathbf{F})^{-1}\mathbf{L}_1\mathbf{D}(\nabla\mathbf{F})^{-\top} + (\nabla\mathbf{F})^{-1}\frac{d\mathbf{D}}{d\tau}(\nabla\mathbf{F})^{-\top} - (\nabla\mathbf{F})^{-1}\mathbf{D}\mathbf{L}_1^{\top}(\nabla\mathbf{F})^{-\top} \\
&= -\left(\nabla\mathbf{F}\right)^{-1}\left(\mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_1^{\top}\right)\left(\nabla\mathbf{F}\right)^{-\top}
\end{aligned} \tag{B.8}$$

where it is understood $\mathbf{L}_1 = \mathbf{L}_1(\mathbf{x}, t_0 + \tau) = \nabla\mathbf{v}(\mathbf{x}, t_0 + \tau)$ and $\mathbf{D} = \mathbf{D}(\mathbf{x}, t_0 + \tau)$ where $\mathbf{x} = \mathbf{F}_{t_0}^{t_0+\tau}(\mathbf{x}_0)$ is the spatial point occupied by the material point $\mathbf{x}_0$ at time $t_0 + \tau$ before we evaluate at $\tau = 0$. $\frac{d\mathbf{D}}{d\tau} = \nabla\mathbf{D}\cdot\mathbf{v} + \frac{\partial\mathbf{D}}{\partial\tau}$ is the material derivative of $\mathbf{D}$. Evaluating at $\tau = 0$ we have,

$$\left.\frac{d\mathbf{T}}{d\tau}\right|_{\tau=0} = -\left(\mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_1^{\top}\right) = -2\mathbf{S}_{\mathbf{D}} \tag{B.9}$$

where everything is now evaluated at ($\mathbf{x}_0$, t_0) and $\mathbf{S}_{\mathbf{D}}$ is the diffusion weighted Eulerian rate-of-strain tensor, defined as

$$\mathbf{S}_{\mathbf{D}} \coloneqq \frac{1}{2}\left(\mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_{\mathbf{1}}{}^{\top}\right) \tag{B.10}$$

Though not currently needed, we extend to second order as well. For second order, we need

the second derivative w.r.t. τ,

$$\begin{aligned}\frac{d^2\mathbf{T}}{d\tau^2} &= -\frac{d(\nabla\mathbf{F})^{-1}}{d\tau}\mathbf{A}(\nabla\mathbf{F})^{-\top} - (\nabla\mathbf{F})^{-1}\frac{d\mathbf{A}}{d\tau}(\nabla\mathbf{F})^{-\top} - (\nabla\mathbf{F})^{-1}\mathbf{A}\frac{d(\nabla\mathbf{F})^{-\top}}{d\tau} \\ &= (\nabla\mathbf{F})^{-1}\left(\mathbf{L}_1\mathbf{A} - \frac{d\mathbf{A}}{d\tau} + \mathbf{A}\mathbf{L}_1^\top\right)(\nabla\mathbf{F})^{-\top}\end{aligned} \tag{B.11}$$

where $\mathbf{A} = \mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_1^\top$. Therefore, for $\frac{d\mathbf{A}}{d\tau}$ we have,

$$\begin{aligned}\frac{d\mathbf{A}}{d\tau} &= \frac{d\mathbf{L}_1}{d\tau}\mathbf{D} + \mathbf{L}_1\frac{d\mathbf{D}}{d\tau} - \frac{d^2\mathbf{D}}{d\tau^2} + \frac{d\mathbf{D}}{d\tau}\mathbf{L}_1^\top + \mathbf{D}\frac{d\mathbf{L}_1^\top}{d\tau} \\ &= \left(\mathbf{L}_2 - \mathbf{L}_1^2\right)\mathbf{D} + \mathbf{L}_1\frac{d\mathbf{D}}{d\tau} - \frac{d^2\mathbf{D}}{d\tau^2} + \frac{d\mathbf{D}}{d\tau}\mathbf{L}_1^\top + \mathbf{D}\left(\mathbf{L}_2^\top - \left(\mathbf{L}_1^\top\right)^2\right)\end{aligned} \tag{B.12}$$

where we used the fact that $\frac{d\mathbf{L}_1}{d\tau} = \mathbf{L}_2 - \mathbf{L}_1^2$, $\frac{d\mathbf{L}_1^\top}{d\tau} = \mathbf{L}_2^\top - (\mathbf{L}_1^\top)^2$ and $\mathbf{L}_2 = \nabla\mathbf{a}$ where $\mathbf{a}$ is the material derivative of $\mathbf{v}$ (i.e. $\frac{d\mathbf{v}}{d\tau} = \nabla\mathbf{v}\cdot\mathbf{v} + \frac{\partial\mathbf{v}}{\partial\tau}$). Putting it all together, we have,

$$\begin{aligned}\frac{d^2\mathbf{T}}{d\tau^2} = (\nabla\mathbf{F})^{-1}\Bigg(\mathbf{L}_1\left(\mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_1^\top\right) - \Bigg(\left(\mathbf{L}_2 - \mathbf{L}_1^2\right)\mathbf{D} + \mathbf{L}_1\frac{d\mathbf{D}}{d\tau} - \frac{d^2\mathbf{D}}{d\tau^2} \\ + \frac{d\mathbf{D}}{d\tau}\mathbf{L}_1^\top + \mathbf{D}\left(\mathbf{L}_2^\top - \left(\mathbf{L}_1^\top\right)^2\right)\Bigg) + \left(\mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_1^\top\right)\mathbf{L}_1^\top\Bigg)(\nabla\mathbf{F})^{-\top}\end{aligned} \tag{B.13}$$

Evaluating at $\tau = 0$ we have,

$$\begin{aligned}\left.\frac{d^2\mathbf{T}}{d\tau^2}\right|_{\tau=0} &= \mathbf{L}_1\left(\mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_1^\top\right) - \Bigg(\left(\mathbf{L}_2 - \mathbf{L}_1^2\right)\mathbf{D} + \mathbf{L}_1\frac{d\mathbf{D}}{d\tau} - \frac{d^2\mathbf{D}}{d\tau^2} \\ &\quad + \frac{d\mathbf{D}}{d\tau}\mathbf{L}_1^\top + \mathbf{D}\left(\mathbf{L}_2^\top - \left(\mathbf{L}_1^\top\right)^2\right)\Bigg) + \left(\mathbf{L}_1\mathbf{D} - \frac{d\mathbf{D}}{d\tau} + \mathbf{D}\mathbf{L}_1^\top\right)\mathbf{L}_1^\top\end{aligned} \tag{B.14}$$

where everything is now evaluated at $\mathbf{x}_0$ and t_0. For simplicity, we will just write $\ddot{\mathbf{T}}$ for this tensor. Then we can write the Taylor expansion as

$$\begin{aligned}\mathbf{T} &= \mathbf{T}\big|_{\tau=0} + \tau\frac{d\mathbf{T}}{d\tau}\bigg|_{\tau=0} + \frac{\tau^2}{2}\frac{d^2\mathbf{T}}{d\tau^2}\bigg|_{\tau=0} + \mathcal{O}(\tau^3)\\ &= \mathbf{D} - 2\tau\mathbf{S_D} + \frac{\tau^2}{2}\ddot{\mathbf{T}} + \mathcal{O}(\tau^3)\end{aligned} \tag{B.15}$$

B.3 Objectivity of $\mathbf{S_D}$

Here we show that the diffusion weigthed Eulerian rate-of-strain tensor $\mathbf{S_D}$ is objective. Consider a general frame change $\mathbf{x} \mapsto \mathbf{y}$ given by,

$$\mathbf{x} = \mathbf{Q}(t)\mathbf{y} + \mathbf{b}(t) \tag{B.16}$$

where $\mathbf{Q}(t)$ is a time-dependent rotation ($\implies \mathbf{Q}(t)\mathbf{Q}(t)^\top = \mathbf{Q}(t)^\top\mathbf{Q}(t) = \mathbf{I}$) and $\mathbf{b}(t) \in \mathbb{R}^n$ is a time dependent translation. Then, an Eulerian tensor $\mathbf{A}(\mathbf{x}, t)$ is *objective* (or *frame indifferent*) if under a general frame change (B.16) the transformed tensor $\tilde{\mathbf{A}}(\mathbf{y}, t)$, is similar to $\mathbf{A}(\mathbf{x}, t)$ with $\mathbf{Q}(t)$ as the similarity transformation matrix, i.e.,

$$\tilde{\mathbf{A}}(\mathbf{y}, t) = \mathbf{Q}(t)^\top\mathbf{A}(\mathbf{x}, t)\mathbf{Q}(t) \tag{B.17}$$

By differentiating Eq. (B.16) w.r.t. t and rearranging we can obtain the transformed velocity,

$$\begin{aligned}\mathbf{v}(\mathbf{x}, t) = \dot{\mathbf{x}} &= \dot{\mathbf{Q}}(t)\mathbf{y} + \mathbf{Q}(t)\dot{\mathbf{y}} + \dot{\mathbf{b}}(t)\\ \implies \tilde{\mathbf{v}}(\mathbf{y}, t) = \dot{\mathbf{y}} &= \mathbf{Q}(t)^\top(\mathbf{v}(\mathbf{x}, t) - \dot{\mathbf{Q}}(t)\mathbf{y} - \dot{\mathbf{b}}(t))\end{aligned} \tag{B.18}$$

which is not objective (for a vector field $\mathbf{v}(\mathbf{x}, t)$ to be objective it must satisfy $\tilde{\mathbf{v}}(\mathbf{y}, t) = \mathbf{Q}^\top(t)\mathbf{v}(\mathbf{x}, t)$). From here on out we will suppress arguments and it will be implied that quantities with $\tilde{}$ are functions of $\mathbf{y}$ and those without are functions of $\mathbf{x}$. Take a $\mathbf{y}$ derivative

of the transformed velocity to obtain the transformed velocity gradient,

$$\widetilde{\nabla}\tilde{\mathbf{v}} = \mathbf{Q}^\top \left(\nabla \mathbf{v} \frac{d\mathbf{x}}{d\mathbf{y}} - \dot{\mathbf{Q}} \right) = \mathbf{Q}^\top \nabla \mathbf{v} \mathbf{Q} - \mathbf{Q}^\top \dot{\mathbf{Q}} \tag{B.19}$$

where $\widetilde{\nabla} = \frac{\partial}{\partial \mathbf{y}}$. So clearly, $\nabla \mathbf{v}$ is not objective. It is a classic result that the Eulerian rate-of-strain tensor $\mathbf{S} = \frac{1}{2}(\nabla \mathbf{v} + (\nabla \mathbf{v})^\top)$ is objective [55]. Note that Haller [60] shows that the diffusion structure tensor $\mathbf{D}$ is objective, as a consequence of requiring the diffusive flux to be objective. So we have,

$$\widetilde{\mathbf{D}}(\mathbf{y}, t) = \mathbf{Q}(t)^\top \mathbf{D}(\mathbf{x}, t) \mathbf{Q}(t) \tag{B.20}$$

and its material derivative transforms as,

$$\dot{\widetilde{\mathbf{D}}} = \dot{\mathbf{Q}}^\top \mathbf{D} \mathbf{Q} + \mathbf{Q}^\top \dot{\mathbf{D}} \mathbf{Q} + \mathbf{Q}^\top \mathbf{D} \dot{\mathbf{Q}} \tag{B.21}$$

showing the classic result that even for an objective field, in general, its material derivative will not be objective [55]. Before we proceed, we state some properties of rotation tensors that will be useful to us. Note,

$$\begin{aligned} \dot{\mathbf{Q}\mathbf{Q}^\top} &= \dot{\mathbf{Q}}\mathbf{Q}^\top + \mathbf{Q}\dot{\mathbf{Q}}^\top = \mathbf{0} \\ \implies \dot{\mathbf{Q}}\mathbf{Q}^\top &= -\mathbf{Q}\dot{\mathbf{Q}}^\top = -[\dot{\mathbf{Q}}\mathbf{Q}^\top]^\top \end{aligned} \tag{B.22}$$

showing that $\dot{\mathbf{Q}}\mathbf{Q}^\top$ is skew-symmetric. This implies

$$\mathbf{Q}^\top \dot{\mathbf{Q}} \mathbf{Q}^\top = -\dot{\mathbf{Q}}^\top \tag{B.23}$$

Also note that the skew-symmetric property allows us to write the transpose of the transformed velocity gradient as

$$(\widetilde{\nabla}\tilde{\mathbf{v}})^\top = \mathbf{Q}^\top(\nabla\mathbf{v})^\top\mathbf{Q} + \mathbf{Q}^\top\dot{\mathbf{Q}} \tag{B.24}$$

To slightly simplify notation we will use the common continuum mechanics notation for the spatial gradient of the velocity field, $\mathbf{L} = \nabla\mathbf{v}$. Recall we defined the diffusion weighted Eulerian rate-of-strain tensor as

$$\mathbf{S_D} = \frac{1}{2}\left(\mathbf{LD} - \dot{\mathbf{D}} + \mathbf{DL}^\top\right) \tag{B.25}$$

leading to the transformed version

$$\tilde{\mathbf{S}}_\mathbf{D} = \frac{1}{2}\left(\tilde{\mathbf{L}}\tilde{\mathbf{D}} - \dot{\tilde{\mathbf{D}}} + \tilde{\mathbf{D}}\tilde{\mathbf{L}}^\top\right) \tag{B.26}$$

Now substituting in using eqs. (B.19), (B.20), (B.21), and (B.24)

$$\begin{aligned}\tilde{\mathbf{S}}_\mathbf{D} = \frac{1}{2}\Big(&(\mathbf{Q}^\top\mathbf{LQ} - \mathbf{Q}^\top\dot{\mathbf{Q}})(\mathbf{Q}^\top\mathbf{DQ}) \\ &-(\dot{\mathbf{Q}}^\top\mathbf{DQ} + \mathbf{Q}^\top\dot{\mathbf{D}}\mathbf{Q} + \mathbf{Q}^\top\mathbf{D}\dot{\mathbf{Q}}) \\ &+(\mathbf{Q}^\top\mathbf{DQ})(\mathbf{Q}^\top\mathbf{L}^\top\mathbf{Q} + \mathbf{Q}^\top\dot{\mathbf{Q}})\Big)\end{aligned} \tag{B.27}$$

Simplifying yields,

$$\begin{aligned}\tilde{\mathbf{S}}_\mathbf{D} =&\frac{1}{2}\left(\mathbf{Q}^\top\mathbf{LDQ} - \mathbf{Q}^\top\dot{\mathbf{D}}\mathbf{Q} + \mathbf{Q}^\top\mathbf{DL}^\top\mathbf{Q}\right) \\ &+\frac{1}{2}\left(-\mathbf{Q}^\top\dot{\mathbf{Q}}\mathbf{Q}^\top\mathbf{DQ} - \dot{\mathbf{Q}}^\top\mathbf{DQ} - \mathbf{Q}^\top\mathbf{D}\dot{\mathbf{Q}} + \mathbf{Q}^\top\mathbf{DQQ}^\top\dot{\mathbf{Q}}\right)\end{aligned} \tag{B.28}$$

We focus on the bottom term and use properties of $\mathbf{Q}$ as shown in Eq. (B.23). We ignore the $1/2$ factor our front and have,

$$\begin{aligned} &-\mathbf{Q}^\top\dot{\mathbf{Q}}\mathbf{Q}^\top\mathbf{D}\mathbf{Q} - \dot{\mathbf{Q}}^\top\mathbf{D}\mathbf{Q} - \mathbf{Q}^\top\mathbf{D}\dot{\mathbf{Q}} + \mathbf{Q}^\top\mathbf{D}\mathbf{Q}\mathbf{Q}^\top\dot{\mathbf{Q}} \\ &= \dot{\mathbf{Q}}^\top\mathbf{D}\mathbf{Q} - \dot{\mathbf{Q}}^\top\mathbf{D}\mathbf{Q} - \mathbf{Q}^\top\mathbf{D}\dot{\mathbf{Q}} + \mathbf{Q}^\top\mathbf{D}\dot{\mathbf{Q}} = \mathbf{0} \end{aligned} \tag{B.29}$$

Therefore the bottom term is the zero matrix and we have,

$$\begin{aligned} \tilde{\mathbf{S}}_\mathbf{D} &= \frac{1}{2}\left(\mathbf{Q}^\top\mathbf{L}\mathbf{D}\mathbf{Q} - \mathbf{Q}^\top\dot{\mathbf{D}}\mathbf{Q} + \mathbf{Q}^\top\mathbf{D}\mathbf{L}^\top\mathbf{Q}\right) \\ &= \mathbf{Q}^\top\frac{1}{2}\left(\mathbf{L}\mathbf{D} - \dot{\mathbf{D}} + \mathbf{D}\mathbf{L}^\top\right)\mathbf{Q} \\ &= \mathbf{Q}^\top\mathbf{S}_\mathbf{D}\mathbf{Q} \end{aligned} \tag{B.30}$$

Hence $\mathbf{S}_\mathbf{D}$ is objective.

B.4 Barrier equations in 2D

Here we go through the variational problem to arrive at the equations for diffusive flux-rate barriers. Recall we seek stationary solutions of the following functional,

$$\dot{\mathcal{T}}_{t_0}(\mathcal{M}_0) = \frac{\int_{\mathcal{M}_0}\langle\mathbf{n}_0, -\mathbf{S}_\mathbf{D}\mathbf{n}_0\rangle\, dA_0}{\int_{\mathcal{M}_0} dA_0}, \tag{B.31}$$

Using results about quotient functionals, stationary curves of this functional will coincide with stationary curves of,

$$\dot{\mathcal{E}}_{\dot{\mathcal{T}}_0} = \int_{\mathcal{M}_0}\left[\langle\mathbf{n}_0, -\mathbf{S}_\mathbf{D}\mathbf{n}_0\rangle - \mu\right] dA_0 \tag{B.32}$$

where μ is the constant value of the functional found by a minimizing solution $\mathcal{M}_0^*$,

$$\mu = \frac{\int_{\mathcal{M}_0^*} \langle \mathbf{n}_0, -\mathbf{S_D}\mathbf{n}_0 \rangle \, dA_0}{\int_{\mathcal{M}_0^*} dA_0} \tag{B.33}$$

Stationary curves will be null-geodesics of this metric, i.e.,

$$\langle \mathbf{n}_0, (-\mathbf{S_D} - \mu\mathbf{I})\mathbf{n}_0 \rangle = 0 \tag{B.34}$$

We now need to rewrite this in terms of tangents rather than normals. Let $\mathbf{Q}$ be a 90° counterclockwise rotation matrix. Then, we can rewrite Eq. (B.34) as

$$\begin{aligned} &\langle \mathbf{Q}\mathbf{r}', (-\mathbf{S_D} - \mu\mathbf{I})\mathbf{Q}\mathbf{r}' \rangle = 0 \\ &\left\langle \mathbf{r}', \mathbf{Q}^\top(-\mathbf{S_D} - \mu\mathbf{I})\mathbf{Q}\mathbf{r}' \right\rangle = 0 \\ &\left\langle \mathbf{r}', (-\mathbf{Q}^\top\mathbf{S_D}\mathbf{Q} - \mu\mathbf{I})\mathbf{r}' \right\rangle = 0 \end{aligned} \tag{B.35}$$

Following Haller and Beron-Vera [61], we rewrite the tangents in terms of the eigenvectors of $\mathbf{S_D}$ and scalars α, β such that $\alpha^2 + \beta^2 = 1$, i.e.,

$$\begin{aligned} &\left\langle \alpha\boldsymbol{e}_1 + \beta\mathbf{e}_2, (-\mathbf{Q}^\top\mathbf{S_D}\mathbf{Q} - \mu\mathbf{I})(\alpha\boldsymbol{e}_1 + \beta\mathbf{e}_2) \right\rangle = 0 \\ &\langle \alpha\boldsymbol{e}_1 + \beta\mathbf{e}_2, -\alpha\mathfrak{s}_2\boldsymbol{e}_1 - \beta\mathfrak{s}_1\boldsymbol{e}_2 - \mu\alpha\boldsymbol{e}_1 - \mu\beta\mathbf{e}_2 \rangle = 0 \\ &-\alpha^2\mathfrak{s}_2 - \mu\alpha^2 - \beta^2\mathfrak{s}_1 - \mu\beta^2 = 0 \end{aligned} \tag{B.36}$$

Letting $\beta^2 = 1 - \alpha^2$ we have,

$$\begin{aligned} &-\alpha^2 \mathfrak{z}_2 - \mu\alpha^2 - (1-\alpha^2)\mathfrak{z}_1 - \mu(1-\alpha^2) = 0 \\ &\alpha^2(\mathfrak{z}_1 - \mathfrak{z}_2) = \mathfrak{z}_1 + \mu \\ &\alpha = \sqrt{\frac{\mathfrak{z}_1 + \mu}{\mathfrak{z}_1 - \mathfrak{z}_2}} \implies \beta = \sqrt{\frac{-(\mu + \mathfrak{z}_2)}{\mathfrak{z}_1 - \mathfrak{z}_2}} \end{aligned} \tag{B.37}$$

These constants would define differential equations on the following domain,

$$U_\mu = \{\mathbf{x}_0 \in U : \mathfrak{z}_1 \neq \mathfrak{z}_2,\ -\mathfrak{z}_1 < \mu < -\mathfrak{z}_2\}. \tag{B.38}$$

By letting $\dot{\mathcal{T}}_0 = -\mu$, the constants become

$$\alpha = \sqrt{\frac{\mathfrak{z}_1 - \dot{\mathcal{T}}_0}{\mathfrak{z}_1 - \mathfrak{z}_2}} \implies \beta = \sqrt{\frac{\dot{\mathcal{T}}_0 - \mathfrak{z}_2}{\mathfrak{z}_1 - \mathfrak{z}_2}} \tag{B.39}$$

with corresponding domain of existence,

$$U_{\dot{\mathcal{T}}_0} = \{\mathbf{x}_0 \in U : \mathfrak{z}_1 \neq \mathfrak{z}_2,\ \mathfrak{z}_2 < \dot{\mathcal{T}}_0 < \mathfrak{z}_1\}. \tag{B.40}$$

Since we seek curves $\mathbf{r}(s)$ along which this metric is null, this leads to the differential equations in Eq. (4.42)

$$\mathbf{r}'(s) = \boldsymbol{\chi}^{\pm}_{\dot{\mathcal{T}}_0}(\mathbf{r}(s)), \quad \boldsymbol{\chi}^{\pm}_{\dot{\mathcal{T}}_0} = \sqrt{\frac{\mathfrak{z}_1 - \dot{\mathcal{T}}_0}{\mathfrak{z}_1 - \mathfrak{z}_2}}\boldsymbol{e}_1 \pm \sqrt{\frac{\dot{\mathcal{T}}_0 - \mathfrak{z}_2}{\mathfrak{z}_1 - \mathfrak{z}_2}}\boldsymbol{e}_2 \tag{B.41}$$

Appendix C

Appendices: Accurate and Efficient extraction of LCS from their variational theory using Automatic Differentiation

C.1 AD Table

Table C.1: Performance using Automatic Differentiation

Method	Precision	Rel Tol	Abs Tol	Avg Err (rad)	Avg Err (deg)	Runtime (s)
TSIT5	Single	1E-3	1E-5	1.32×10^{-1}	7.5535	1.17
TSIT5	Single	1E-4	1E-6	3.88×10^{-2}	2.2223	1.55
TSIT5	Single	1E-5	1E-7	2.21×10^{-2}	1.2660	2.35
TSIT5	Single	1E-6	1E-8	1.15×10^{-2}	0.6599	4.85
TSIT5	Single	1E-7	1E-9	4.59×10^{-3}	0.2631	11.63
TSIT5	Single	1E-8	1E-10	1.91×10^{-3}	0.1095	24.74
TSIT5	Single	1E-9	1E-11	7.95×10^{-4}	0.0455	47.49
TSIT5	Double	1E-3	1E-5	1.32×10^{-1}	7.5481	2.11
TSIT5	Double	1E-4	1E-6	3.89×10^{-2}	2.2268	2.54
TSIT5	Double	1E-5	1E-7	2.25×10^{-2}	1.2876	4.00
TSIT5	Double	1E-6	1E-8	1.18×10^{-2}	0.6776	8.51
TSIT5	Double	1E-7	1E-9	4.74×10^{-3}	0.2714	21.82
TSIT5	Double	1E-8	1E-10	2.03×10^{-3}	0.1164	58.61
TSIT5	Double	1E-9	1E-11	1.13×10^{-3}	0.0648	128.91
DOPRI8	Single	1E-3	1E-5	1.32×10^{-2}	0.7548	2.14
DOPRI8	Single	1E-4	1E-6	8.01×10^{-3}	0.4590	4.04
DOPRI8	Single	1E-5	1E-7	5.02×10^{-3}	0.2876	9.23
DOPRI8	Single	1E-6	1E-8	2.33×10^{-3}	0.1333	19.37
DOPRI8	Single	1E-7	1E-9	9.78×10^{-4}	0.0560	42.80
DOPRI8	Single	1E-8	1E-10	3.78×10^{-4}	0.0216	75.90
DOPRI8	Single	1E-9	1E-11	1.67×10^{-4}	0.0095	263.53
DOPRI8	Double	1E-3	1E-5	1.32×10^{-2}	0.7541	3.67
DOPRI8	Double	1E-4	1E-6	8.42×10^{-3}	0.4822	7.51
DOPRI8	Double	1E-5	1E-7	5.30×10^{-3}	0.3039	16.47
DOPRI8	Double	1E-6	1E-8	2.46×10^{-3}	0.1411	38.65
DOPRI8	Double	1E-7	1E-9	1.04×10^{-3}	0.0594	104.98
DOPRI8	Double	1E-8	1E-10	4.36×10^{-4}	0.0250	226.66
DOPRI8	Double	1E-9	1E-11	2.38×10^{-4}	0.0136	377.21

C.2 Numerical Differentiation Tables

Table C.2: Performance of TSIT5 (single) using Numerical Differentiation

Method	Precision	h	Rel Tol	Abs Tol	Avg Err (rad)	Avg Err (deg)	Runtime (s)
TSIT5	Single	1E-2	1E-3	1E-5	1.66×10^{-1}	9.5187	1.36
TSIT5	Single	1E-2	1E-4	1E-6	5.45×10^{-2}	3.1204	1.59
TSIT5	Single	1E-2	1E-5	1E-7	3.62×10^{-2}	2.0753	2.82
TSIT5	Single	1E-2	1E-6	1E-8	3.43×10^{-2}	1.9636	5.36
TSIT5	Single	1E-2	1E-7	1E-9	3.42×10^{-2}	1.9585	12.18
TSIT5	Single	1E-2	1E-8	1E-10	3.42×10^{-2}	1.9584	25.70
TSIT5	Single	1E-2	1E-9	1E-11	3.42×10^{-2}	1.9583	48.14
TSIT5	Single	1E-3	1E-3	1E-5	5.71×10^{-1}	32.7002	1.41
TSIT5	Single	1E-3	1E-4	1E-6	2.69×10^{-1}	15.4056	1.61
TSIT5	Single	1E-3	1E-5	1E-7	6.33×10^{-2}	3.6261	2.75
TSIT5	Single	1E-3	1E-6	1E-8	1.16×10^{-2}	0.6635	5.38
TSIT5	Single	1E-3	1E-7	1E-9	2.60×10^{-3}	0.1490	12.22
TSIT5	Single	1E-3	1E-8	1E-10	1.50×10^{-3}	0.0857	25.79
TSIT5	Single	1E-3	1E-9	1E-11	1.43×10^{-3}	0.0818	47.02
TSIT5	Single	1E-4	1E-3	1E-5	7.04×10^{-1}	40.3508	1.35
TSIT5	Single	1E-4	1E-4	1E-6	7.04×10^{-1}	40.3305	1.64
TSIT5	Single	1E-4	1E-5	1E-7	4.49×10^{-1}	25.7462	2.65
TSIT5	Single	1E-4	1E-6	1E-8	1.07×10^{-1}	6.1147	5.36
TSIT5	Single	1E-4	1E-7	1E-9	1.67×10^{-2}	0.9581	12.03
TSIT5	Single	1E-4	1E-8	1E-10	3.26×10^{-3}	0.1867	26.30
TSIT5	Single	1E-4	1E-9	1E-11	1.95×10^{-3}	0.1118	46.15
TSIT5	Single	1E-5	1E-3	1E-5	7.13×10^{-1}	40.8441	1.44
TSIT5	Single	1E-5	1E-4	1E-6	7.69×10^{-1}	44.0789	1.69
TSIT5	Single	1E-5	1E-5	1E-7	7.65×10^{-1}	43.8429	2.65
TSIT5	Single	1E-5	1E-6	1E-8	5.89×10^{-1}	33.7666	5.27
TSIT5	Single	1E-5	1E-7	1E-9	1.62×10^{-1}	9.2965	12.02
TSIT5	Single	1E-5	1E-8	1E-10	3.21×10^{-2}	1.8387	25.44
TSIT5	Single	1E-5	1E-9	1E-11	1.92×10^{-2}	1.0981	46.16
TSIT5	Single	1E-6	1E-3	1E-5	7.55×10^{-1}	43.2325	1.36
TSIT5	Single	1E-6	1E-4	1E-6	7.77×10^{-1}	44.5053	1.66
TSIT5	Single	1E-6	1E-5	1E-7	7.85×10^{-1}	45.0000	2.66
TSIT5	Single	1E-6	1E-6	1E-8	7.79×10^{-1}	44.6558	5.33
TSIT5	Single	1E-6	1E-7	1E-9	6.83×10^{-1}	39.1095	12.17
TSIT5	Single	1E-6	1E-8	1E-10	2.95×10^{-1}	16.9136	25.75
TSIT5	Single	1E-6	1E-9	1E-11	1.80×10^{-1}	10.2884	47.14
TSIT5	Single	1E-7	1E-3	1E-5	7.77×10^{-1}	44.5125	1.43
TSIT5	Single	1E-7	1E-4	1E-6	7.78×10^{-1}	44.6027	1.65
TSIT5	Single	1E-7	1E-5	1E-7	7.83×10^{-1}	44.8685	2.65
TSIT5	Single	1E-7	1E-6	1E-8	7.84×10^{-1}	44.9226	5.24
TSIT5	Single	1E-7	1E-7	1E-9	7.86×10^{-1}	45.0478	12.29
TSIT5	Single	1E-7	1E-8	1E-10	7.43×10^{-1}	42.5517	24.99
TSIT5	Single	1E-7	1E-9	1E-11	6.74×10^{-1}	38.6374	46.73
TSIT5	Single	1E-8	1E-3	1E-5	7.85×10^{-1}	44.9776	1.42
TSIT5	Single	1E-8	1E-4	1E-6	7.85×10^{-1}	45.0007	1.61
TSIT5	Single	1E-8	1E-5	1E-7	7.86×10^{-1}	45.0179	2.63
TSIT5	Single	1E-8	1E-6	1E-8	7.86×10^{-1}	45.0132	5.21
TSIT5	Single	1E-8	1E-7	1E-9	7.85×10^{-1}	45.0009	11.94
TSIT5	Single	1E-8	1E-8	1E-10	7.85×10^{-1}	45.0017	24.43
TSIT5	Single	1E-8	1E-9	1E-11	7.85×10^{-1}	44.9602	45.61

Table C.3: Performance of TSIT5 (double) using Numerical Differentiation

Method	Precision	h	Rel Tol	Abs Tol	Avg Err (rad)	Avg Err (deg)	Runtime (s)
TSIT5	Double	1E-2	1E-3	1E-5	1.67×10^{-1}	9.5533	2.07
TSIT5	Double	1E-2	1E-4	1E-6	5.44×10^{-2}	3.1150	2.38
TSIT5	Double	1E-2	1E-5	1E-7	3.62×10^{-2}	2.0766	3.67
TSIT5	Double	1E-2	1E-6	1E-8	3.43×10^{-2}	1.9637	7.06
TSIT5	Double	1E-2	1E-7	1E-9	3.42×10^{-2}	1.9586	17.01
TSIT5	Double	1E-2	1E-8	1E-10	3.42×10^{-2}	1.9584	42.54
TSIT5	Double	1E-2	1E-9	1E-11	3.42×10^{-2}	1.9584	89.31
TSIT5	Double	1E-3	1E-3	1E-5	5.69×10^{-1}	32.5994	2.06
TSIT5	Double	1E-3	1E-4	1E-6	2.66×10^{-1}	15.2441	2.38
TSIT5	Double	1E-3	1E-5	1E-7	6.36×10^{-2}	3.6456	3.66
TSIT5	Double	1E-3	1E-6	1E-8	1.16×10^{-2}	0.6651	7.03
TSIT5	Double	1E-3	1E-7	1E-9	2.62×10^{-3}	0.1503	16.78
TSIT5	Double	1E-3	1E-8	1E-10	1.50×10^{-3}	0.0860	41.00
TSIT5	Double	1E-3	1E-9	1E-11	1.38×10^{-3}	0.0789	89.88
TSIT5	Double	1E-4	1E-3	1E-5	7.05×10^{-1}	40.4091	2.11
TSIT5	Double	1E-4	1E-4	1E-6	6.84×10^{-1}	39.2123	2.30
TSIT5	Double	1E-4	1E-5	1E-7	4.53×10^{-1}	25.9368	3.66
TSIT5	Double	1E-4	1E-6	1E-8	1.07×10^{-1}	6.1552	6.82
TSIT5	Double	1E-4	1E-7	1E-9	1.71×10^{-2}	0.9794	16.46
TSIT5	Double	1E-4	1E-8	1E-10	3.36×10^{-3}	0.1926	41.93
TSIT5	Double	1E-4	1E-9	1E-11	6.50×10^{-4}	0.0372	88.79
TSIT5	Double	1E-5	1E-3	1E-5	7.16×10^{-1}	41.0464	2.08
TSIT5	Double	1E-5	1E-4	1E-6	7.46×10^{-1}	42.7153	2.46
TSIT5	Double	1E-5	1E-5	1E-7	7.64×10^{-1}	43.8013	3.65
TSIT5	Double	1E-5	1E-6	1E-8	5.89×10^{-1}	33.7323	6.80
TSIT5	Double	1E-5	1E-7	1E-9	1.64×10^{-1}	9.3869	16.35
TSIT5	Double	1E-5	1E-8	1E-10	3.30×10^{-2}	1.8895	40.90
TSIT5	Double	1E-5	1E-9	1E-11	5.89×10^{-3}	0.3375	88.57
TSIT5	Double	1E-6	1E-3	1E-5	7.24×10^{-1}	41.4679	2.12
TSIT5	Double	1E-6	1E-4	1E-6	7.49×10^{-1}	42.9108	2.53
TSIT5	Double	1E-6	1E-5	1E-7	7.83×10^{-1}	44.8809	3.65
TSIT5	Double	1E-6	1E-6	1E-8	7.83×10^{-1}	44.8575	7.19
TSIT5	Double	1E-6	1E-7	1E-9	6.84×10^{-1}	39.2186	16.34
TSIT5	Double	1E-6	1E-8	1E-10	3.01×10^{-1}	17.2362	41.08
TSIT5	Double	1E-6	1E-9	1E-11	5.89×10^{-2}	3.3722	88.59
TSIT5	Double	1E-7	1E-3	1E-5	7.31×10^{-1}	41.8745	2.23
TSIT5	Double	1E-7	1E-4	1E-6	7.53×10^{-1}	43.1265	2.39
TSIT5	Double	1E-7	1E-5	1E-7	7.86×10^{-1}	45.0282	3.70
TSIT5	Double	1E-7	1E-6	1E-8	7.89×10^{-1}	45.2023	7.05
TSIT5	Double	1E-7	1E-7	1E-9	7.84×10^{-1}	44.9482	16.35
TSIT5	Double	1E-7	1E-8	1E-10	7.47×10^{-1}	42.8095	40.91
TSIT5	Double	1E-7	1E-9	1E-11	4.53×10^{-1}	25.9318	88.57
TSIT5	Double	1E-8	1E-3	1E-5	7.36×10^{-1}	42.1552	2.07
TSIT5	Double	1E-8	1E-4	1E-6	7.60×10^{-1}	43.5446	2.33
TSIT5	Double	1E-8	1E-5	1E-7	7.90×10^{-1}	45.2488	3.70
TSIT5	Double	1E-8	1E-6	1E-8	7.89×10^{-1}	45.2180	7.07
TSIT5	Double	1E-8	1E-7	1E-9	7.87×10^{-1}	45.0667	16.35
TSIT5	Double	1E-8	1E-8	1E-10	7.87×10^{-1}	45.0908	41.51
TSIT5	Double	1E-8	1E-9	1E-11	7.69×10^{-1}	44.0464	88.69

Table C.4: Performance of DOPRI8 (single) using Numerical Differentiation

Method	Precision	h	Rel Tol	Abs Tol	Avg Err (rad)	Avg Err (deg)	Runtime (s)
DOPRI8	Single	1E-2	1E-3	1E-5	4.02×10^{-2}	2.304	2.72
DOPRI8	Single	1E-2	1E-4	1E-6	3.46×10^{-2}	1.981	5.73
DOPRI8	Single	1E-2	1E-5	1E-7	3.42×10^{-2}	1.959	11.99
DOPRI8	Single	1E-2	1E-6	1E-8	3.42×10^{-2}	1.958	22.90
DOPRI8	Single	1E-2	1E-7	1E-9	3.42×10^{-2}	1.958	48.68
DOPRI8	Single	1E-2	1E-8	1E-10	3.42×10^{-2}	1.958	85.73
DOPRI8	Single	1E-2	1E-9	1E-11	3.42×10^{-2}	1.958	295.77
DOPRI8	Single	1E-3	1E-3	1E-5	1.25×10^{-1}	7.167	2.53
DOPRI8	Single	1E-3	1E-4	1E-6	2.69×10^{-2}	1.539	5.39
DOPRI8	Single	1E-3	1E-5	1E-7	5.21×10^{-3}	0.299	11.23
DOPRI8	Single	1E-3	1E-6	1E-8	1.80×10^{-3}	0.103	21.26
DOPRI8	Single	1E-3	1E-7	1E-9	1.41×10^{-3}	0.081	44.44
DOPRI8	Single	1E-3	1E-8	1E-10	1.42×10^{-3}	0.081	81.01
DOPRI8	Single	1E-3	1E-9	1E-11	1.51×10^{-3}	0.087	272.74
DOPRI8	Single	1E-4	1E-3	1E-5	5.97×10^{-1}	34.198	2.52
DOPRI8	Single	1E-4	1E-4	1E-6	2.37×10^{-1}	13.591	5.62
DOPRI8	Single	1E-4	1E-5	1E-7	4.45×10^{-2}	2.549	10.87
DOPRI8	Single	1E-4	1E-6	1E-8	7.30×10^{-3}	0.418	21.76
DOPRI8	Single	1E-4	1E-7	1E-9	1.76×10^{-3}	0.101	44.78
DOPRI8	Single	1E-4	1E-8	1E-10	1.76×10^{-3}	0.101	81.31
DOPRI8	Single	1E-4	1E-9	1E-11	3.34×10^{-3}	0.192	271.37
DOPRI8	Single	1E-5	1E-3	1E-5	7.79×10^{-1}	44.606	2.63
DOPRI8	Single	1E-5	1E-4	1E-6	7.07×10^{-1}	40.535	5.47
DOPRI8	Single	1E-5	1E-5	1E-7	3.65×10^{-1}	20.905	11.00
DOPRI8	Single	1E-5	1E-6	1E-8	7.27×10^{-2}	4.167	21.29
DOPRI8	Single	1E-5	1E-7	1E-9	1.71×10^{-2}	0.978	45.00
DOPRI8	Single	1E-5	1E-8	1E-10	1.71×10^{-2}	0.982	79.25
DOPRI8	Single	1E-5	1E-9	1E-11	3.25×10^{-2}	1.865	270.09
DOPRI8	Single	1E-6	1E-3	1E-5	7.81×10^{-1}	44.776	2.56
DOPRI8	Single	1E-6	1E-4	1E-6	7.83×10^{-1}	44.838	5.35
DOPRI8	Single	1E-6	1E-5	1E-7	7.54×10^{-1}	43.172	10.87
DOPRI8	Single	1E-6	1E-6	1E-8	5.08×10^{-1}	29.116	21.83
DOPRI8	Single	1E-6	1E-7	1E-9	1.66×10^{-1}	9.509	44.19
DOPRI8	Single	1E-6	1E-8	1E-10	1.63×10^{-1}	9.325	78.97
DOPRI8	Single	1E-6	1E-9	1E-11	2.69×10^{-1}	15.408	269.94
DOPRI8	Single	1E-7	1E-3	1E-5	7.81×10^{-1}	44.747	2.50
DOPRI8	Single	1E-7	1E-4	1E-6	7.82×10^{-1}	44.779	5.36
DOPRI8	Single	1E-7	1E-5	1E-7	7.85×10^{-1}	44.955	11.17
DOPRI8	Single	1E-7	1E-6	1E-8	7.73×10^{-1}	44.289	21.07
DOPRI8	Single	1E-7	1E-7	1E-9	6.74×10^{-1}	38.622	44.18
DOPRI8	Single	1E-7	1E-8	1E-10	6.58×10^{-1}	37.697	80.36
DOPRI8	Single	1E-7	1E-9	1E-11	7.27×10^{-1}	41.665	271.47
DOPRI8	Single	1E-8	1E-3	1E-5	7.86×10^{-1}	45.022	2.51
DOPRI8	Single	1E-8	1E-4	1E-6	7.86×10^{-1}	45.019	5.32
DOPRI8	Single	1E-8	1E-5	1E-7	7.86×10^{-1}	45.007	11.32
DOPRI8	Single	1E-8	1E-6	1E-8	7.85×10^{-1}	45.003	21.70
DOPRI8	Single	1E-8	1E-7	1E-9	7.86×10^{-1}	45.010	44.04
DOPRI8	Single	1E-8	1E-8	1E-10	7.85×10^{-1}	44.981	78.54
DOPRI8	Single	1E-8	1E-9	1E-11	7.85×10^{-1}	44.989	267.62

Table C.5: Performance of DOPRI8 (double) using Numerical Differentiation

Method	Precision	h	Rel Tol	Abs Tol	Avg Err (rad)	Avg Err (deg)	Runtime (s)
DOPRI8	Double	1E-2	1E-3	1E-5	4.03×10^{-2}	2.312	3.78
DOPRI8	Double	1E-2	1E-4	1E-6	3.46×10^{-2}	1.981	7.62
DOPRI8	Double	1E-2	1E-5	1E-7	3.42×10^{-2}	1.959	16.04
DOPRI8	Double	1E-2	1E-6	1E-8	3.42×10^{-2}	1.958	33.26
DOPRI8	Double	1E-2	1E-7	1E-9	3.42×10^{-2}	1.958	87.46
DOPRI8	Double	1E-2	1E-8	1E-10	3.42×10^{-2}	1.958	180.77
DOPRI8	Double	1E-2	1E-9	1E-11	3.42×10^{-2}	1.958	317.98
DOPRI8	Double	1E-3	1E-3	1E-5	1.25×10^{-1}	7.145	3.96
DOPRI8	Double	1E-3	1E-4	1E-6	2.68×10^{-2}	1.537	7.47
DOPRI8	Double	1E-3	1E-5	1E-7	5.27×10^{-3}	0.302	16.20
DOPRI8	Double	1E-3	1E-6	1E-8	1.81×10^{-3}	0.104	34.61
DOPRI8	Double	1E-3	1E-7	1E-9	1.40×10^{-3}	0.080	86.05
DOPRI8	Double	1E-3	1E-8	1E-10	1.37×10^{-3}	0.079	179.78
DOPRI8	Double	1E-3	1E-9	1E-11	1.37×10^{-3}	0.078	308.97
DOPRI8	Double	1E-4	1E-3	1E-5	6.02×10^{-1}	34.493	3.78
DOPRI8	Double	1E-4	1E-4	1E-6	2.45×10^{-1}	14.023	7.51
DOPRI8	Double	1E-4	1E-5	1E-7	4.52×10^{-2}	2.589	15.58
DOPRI8	Double	1E-4	1E-6	1E-8	7.51×10^{-3}	0.430	33.59
DOPRI8	Double	1E-4	1E-7	1E-9	1.42×10^{-3}	0.081	84.52
DOPRI8	Double	1E-4	1E-8	1E-10	3.73×10^{-4}	0.021	177.68
DOPRI8	Double	1E-4	1E-9	1E-11	1.47×10^{-4}	0.008	312.17
DOPRI8	Double	1E-5	1E-3	1E-5	7.84×10^{-1}	44.893	3.77
DOPRI8	Double	1E-5	1E-4	1E-6	7.11×10^{-1}	40.714	7.50
DOPRI8	Double	1E-5	1E-5	1E-7	3.70×10^{-1}	21.176	15.57
DOPRI8	Double	1E-5	1E-6	1E-8	7.35×10^{-2}	4.213	34.44
DOPRI8	Double	1E-5	1E-7	1E-9	1.37×10^{-2}	0.785	82.89
DOPRI8	Double	1E-5	1E-8	1E-10	3.06×10^{-3}	0.176	178.53
DOPRI8	Double	1E-5	1E-9	1E-11	6.34×10^{-4}	0.036	313.74
DOPRI8	Double	1E-6	1E-3	1E-5	7.94×10^{-1}	45.500	4.05
DOPRI8	Double	1E-6	1E-4	1E-6	7.90×10^{-1}	45.291	7.38
DOPRI8	Double	1E-6	1E-5	1E-7	7.58×10^{-1}	43.438	16.16
DOPRI8	Double	1E-6	1E-6	1E-8	5.08×10^{-1}	29.132	33.57
DOPRI8	Double	1E-6	1E-7	1E-9	1.33×10^{-1}	7.608	87.56
DOPRI8	Double	1E-6	1E-8	1E-10	3.02×10^{-2}	1.729	181.07
DOPRI8	Double	1E-6	1E-9	1E-11	6.31×10^{-3}	0.362	310.72
DOPRI8	Double	1E-7	1E-3	1E-5	7.59×10^{-1}	43.480	3.87
DOPRI8	Double	1E-7	1E-4	1E-6	7.94×10^{-1}	45.521	7.37
DOPRI8	Double	1E-7	1E-5	1E-7	7.96×10^{-1}	45.604	15.80
DOPRI8	Double	1E-7	1E-6	1E-8	7.76×10^{-1}	44.484	33.27
DOPRI8	Double	1E-7	1E-7	1E-9	6.51×10^{-1}	37.310	84.86
DOPRI8	Double	1E-7	1E-8	1E-10	2.79×10^{-1}	15.971	180.58
DOPRI8	Double	1E-7	1E-9	1E-11	6.26×10^{-2}	3.588	307.43
DOPRI8	Double	1E-8	1E-3	1E-5	7.50×10^{-1}	42.947	3.74
DOPRI8	Double	1E-8	1E-4	1E-6	7.61×10^{-1}	43.608	7.53
DOPRI8	Double	1E-8	1E-5	1E-7	7.88×10^{-1}	45.175	15.56
DOPRI8	Double	1E-8	1E-6	1E-8	7.87×10^{-1}	45.113	34.39
DOPRI8	Double	1E-8	1E-7	1E-9	7.84×10^{-1}	44.910	84.91
DOPRI8	Double	1E-8	1E-8	1E-10	7.41×10^{-1}	42.431	177.86
DOPRI8	Double	1E-8	1E-9	1E-11	4.64×10^{-1}	26.596	312.36