Journal of Computational Science 7 (2015) 26–36

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

LCS Tool: A computational platform for Lagrangian coherent structures

K. Onu, F. Huhn, G. Haller*

Institute of Mechanical Systems, Department of Mechanical and Process Engineering, ETH Zurich, Switzerland

ARTICLE INFO

Article history:
Received 13 June 2014
Received in revised form
17 December 2014
Accepted 29 December 2014
Available online 7 January 2015

PACS:
47.10.Fg
47.27.De
47.11.−j
05.45.−a
47.51.+a
47.54.−r
92.10.hf
93.30.Qn
92.10.ab
92.10.Ty
92.10.ak
92.10.A−

MSC:
37N10
37B25
97N80
37C60
37B55
70H33
76F25

Keywords:
Lagrangian coherent structures
Mathematical software
Non-autonomous dynamical systems
Invariant manifolds
Mixing
Transport barriers
Fluid dynamics
Ocean surface flows

ABSTRACT

We give an algorithmic introduction to Lagrangian coherent structures (LCSs) using a newly developed computational engine, LCS Tool. LCSs are most repelling, attracting and shearing material lines that form the centrepieces of observed tracer patterns in two-dimensional unsteady dynamical systems. LCS Tool implements the latest geodesic theory of LCSs for two-dimensional flows, uncovering key transport barriers in unsteady flow velocity data as explicit solutions of differential equations. LCS Tool makes theoretical results accessible to the fluid mechanics community since implementing these results directly could be time consuming. After a review of the underlying theory, we explain the steps and numerical methods used by LCS Tool, and illustrate its capabilities on three unsteady fluid flow examples.

1. Introduction

Lagrangian coherent structures (LCSs) are evolving organizing centres of trajectory patterns in non-autonomous dynamical systems [1–3]. Applications of LCSs include oceanic and atmospheric flows [4,5], biological transport problems [6–8], aeronautics [9],

* Corresponding author. Tel.: +41 44 633 82 50.
E-mail address: georgehaller@ethz.ch (G. Haller).

http://dx.doi.org/10.1016/j.jocs.2014.12.002
1877-7503/© 2015 Elsevier B.V. All rights reserved.

celestial mechanics [10], crowd dynamics [11], and aperiodically forced mechanical oscillators [12].

Haller [13] proposed that ridges of the finite-time Lyapunov exponent (FTLE) are heuristic indicators of hyperbolic (i.e., repelling and attracting type) LCSs. A number of examples support this principle [2]. Equating FTLE ridges with LCSs, however, would create theoretical inconsistencies, as well as false positives and negatives in hyperbolic LCS detection [14,15]. In addition, the role of the FTLE field in the accurate detection of elliptic (vortex-type) and parabolic (jet-core type) LCSs has remained an open question (but see [16]).

More recent work has focused on an exact mathematical formulation of the properties defining LCSs [14,17–22]. In two-dimensional flows, hyperbolic and parabolic LCSs turn out to be stationary curves of the averaged material shear [23], whereas elliptic LCSs are stationary curves of the averaged strain [19,20]. These variational formulations lead to explicit solutions for LCSs as null geodesics of appropriate Lorentzian metrics.

We present an algorithmic introduction to geodesic LCS detection. We then review the implementation of this approach in a computational engine called LCS Tool.[1] This engine is a library of MATLAB functions that extract LCSs from two-dimensional unsteady flows. The examples we present form demonstration scripts distributed with LCS Tool.

Other software for LCS detection exists. ManGen [24] calculates the FTLE and advects material curves in two-dimensional velocity fields. It includes a graphical user interface and uses the Message Passing Interface standard for parallel calculations. Newman [25] calculates the FTLE using dimension independent code. It assists ridge extraction of FTLE fields and supports analytic and dataset velocity definitions. FlowTK [26] calculates the FTLE in two and three-dimensions on Cartesian and unstructured grids. The NVIDIA CUDA parallel computing platform is used for fast computations and Kitware's ParaView data analysis and visualization application is used for a user interface.

These packages generate FTLE plots to aid the visual assessment of hyperbolic LCSs. LCS Tool also has functions to generate FTLE plots, but its emphasis is to provide geodesic extraction of LCSs as parametrized material curves, and extend the scope of such extraction to elliptic LCSs.

2. Theory

We consider two-dimensional, finite-time, unsteady velocity fields of the form

$$\frac{dx}{dt} = v(x,t), \quad x \in U \subset \mathbf{R}^2, \quad t \in [t_-, t_+]. \tag{1}$$

Trajectories of Eq. (1) are denoted $x(t\,; t_0, x_0)$, with $x_0 \in U$ denoting their initial position in the open set U at an initial time $t_0 \in [t_-, t_+]$. The flow map is then defined as

$$F_{t_0}^t(x_0) \equiv x(t; t, x_0),$$

mapping initial positions to current positions at time t. The time interval $[t_-, t_+]$ is part of the definition of the finite-time dynamical system in Eq. (1). This interval may be a time scale of interest or the maximum interval over which velocity data is available from simulations or observations.

The right Cauchy-Green strain tensor associated with the flow map is defined as

$$C_{t_0}^t(x_0) = \left[\nabla F_{t_0}^t(x_0)\right]^T \nabla F_{t_0}^t(x_0), \tag{2}$$

measuring Lagrangian strain in the velocity field. This tensor is symmetric and positive definite [27]. We label the eigenvalues and eigenvectors of $C_{t_0}^t(x_0)$ as follows:

$$C_{t_0}^t \xi_i = \lambda_i \xi_i, \quad 0 < \lambda_1 \le \lambda_2, \quad i = 1, 2;$$

$$\left|\xi_i\right| = 1, \quad \xi_2 = \Omega\xi_1, \quad \Omega = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{3}$$

2.1. Elliptic LCSs

We seek positions of closed material lines at time t_0 that prevail as coherent Lagrangian vortex boundaries (or *elliptic LCSs*) over a time interval $[t_0, t] \subset [t_-, t_+]$. Haller and Beron-Vera [19,20] argue that such initial material line positions are closed stationary curves of the averaged strain functional

$$Q(\gamma) = \frac{1}{\sigma}\int_0^\sigma \frac{\sqrt{\langle r'(s), C_{t_0}^t(r(s))r'(s)\rangle}}{\sqrt{\langle r'(s), r'(s)\rangle}}ds,$$

obtained by averaging the tangential strain arising over $[t_0, t]$ along closed material lines parametrized as $r(s)$ with $s \in [0, \sigma]$. Solutions to this variational problem turn out to be closed orbits of one of two parametrized vector-field families

$$\eta_\pm^\lambda = \sqrt{\frac{\lambda_2 - \lambda^2}{\lambda_2 - \lambda_1}}\xi_1 \pm \sqrt{\frac{\lambda^2 - \lambda_1}{\lambda_2 - \lambda_1}}\xi_2, \tag{4}$$

with $\lambda > 0$ playing the role of a parameter. Such closed orbits satisfy the differential equation

$$r' = \eta_\pm^\lambda(r), \tag{5}$$

which coincide with null geodesics of the Lorentzian metric family

$$e_\lambda(u, v) = \langle u, \left[D_{t_0}^t(r) - \lambda^2 I\right] v\rangle.$$

For this reason, we refer to the computation of elliptic LCSs as limit cycles of Eq. (5) as *geodesic detection of elliptic LCSs*.

Any orbit of Eq. (5) turns out to stretch uniformly under the flow map $F_{t_0}^t$. Specifically, any subset of an orbit of Eq. (5) increases its arc length precisely by a factor of λ. For this reason, we refer to trajectories of Eq. (5) as λ-lines. Following Haller and Beron-Vera [19,20], we call the outermost member of a closed family of λ-lines a coherent Lagrangian vortex boundary.

2.2. Hyperbolic LCSs

Next we consider positions of material lines at time t_0 that prevail as most repelling or attracting material lines (or *hyperbolic LCSs*) over a time interval $[t_0, t] \subset [t_-, t_+]$. Farazmand et al. [23] argue that hyperbolic LCSs are stationary curves of the averaged shear functional

$$Q(\gamma) = \frac{1}{\sigma}\int_0^\sigma \frac{\langle r'(s), D_{t_0}^t(r(s))r'(s)\rangle}{\sqrt{\langle r'(s), C_{t_0}^t(r(s))r'(s)\rangle\langle r'(s), r'(s)\rangle}}ds,$$

$$D_{t_0}^t = \frac{1}{2}[C_{t_0}^t\Omega - \Omega C_{t_0}^t],$$

obtained by averaging the Lagrangian shear arising over $[t_0, t]$ along material lines parametrized as $r(s)$ with $s \in [0, \sigma]$. More precisely, hyperbolic LCSs are stationary curves of $Q(\gamma)$ with respect to fixed-endpoint perturbations. We note that parabolic LCSs (Lagrangian jet cores) are also stationary curves of $Q(\gamma)$, but under variable endpoint perturbations (cf. Farazmand et al. [23]).

[1] LCS Tool is available at: http://www.runmycode.org/companion/view/908.

Solutions to this variational problem turn out to be curves of the ξ_1 or ξ_2 eigenvector field. Repelling LCSs (*shrink lines*) are obtained as trajectories of the differential equation

$$r' = \xi_1(r), \tag{6}$$

and attracting LCSs (*stretch lines*) are obtained as trajectories of the differential equation

$$r' = \xi_2(r). \tag{7}$$

Shrink lines and stretch lines coincide with the null geodesics of the Lorentzian metric $h(u, v) = \langle u, D_{t_0}^t(r)v\rangle$. For this reason, we refer to the computation of hyperbolic LCSs as strongest normally-repelling or normally-attracting curves of Eq. (5) as *geodesic detection of hyperbolic LCSs*.

To compute the normal repulsion of shrink lines, we note that an infinitesimal normal perturbation to a shrink line γ at its point r grows under the flow map $F_{t_0}^t$ by a factor of $\lambda_2(r)$ in the direction normal to $F_{t_0}^t(\gamma)$. Similarly, small normal perturbations to a stretch line decay by a factor $\lambda_1(r)$ in the direction normal to the evolving stretch line.

3. Numerical methods

This section describes stepwise the numerical implementation of geodesic detection of elliptic and hyperbolic LCSs based on Eqs. 5, 6, 7. Table 1 gives an overview of the steps, functions and variable names used in LCS Tool to analyse a flow.

3.1. Computing the invariants of the Cauchy-Green strain tensor

The first step in calculating elliptic and hyperbolic LCSs is the computation of the Cauchy-Green strain tensor field $C_{t_0}^t(x_0)$, as defined in Eq. (2). The function performing this calculation in LCS Tool is `eig_cgStrain`. The main steps executed by this function are enumerated in Table 2, while Table 3 summarizes the syntax of `eig_cgStrain`.

The Cartesian grid of initial conditions mentioned in Table 2 is rectangular, with user-defined vertical and horizontal ranges and resolutions. The optimal resolution may be determined by a successive doubling of the initial resolution until convergence of the extracted LCSs is observed visually. If the domain of interest comprises only a few expected LCSs, e.g., one vortex, then a resolution of about 500 grid points along the longest axis usually gives good results. Otherwise, a higher resolution must be chosen.

The auxiliary grid (cf. Table 2) comprises four points placed symmetrically around each point of the Cartesian grid (Fig. 1). These points are used to achieve increased accuracy in the finite-difference approximation

$$\nabla F_{t_0}^t(x_0) \approx \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix}$$

$$\alpha_{ij} \equiv \frac{x_i(t; t_0, x_0 + \delta x_j) - x_i(t; t_0, x_0 - \delta x_j)}{2|\delta x_j|}$$

of $\nabla F_{t_0}^t$ at a point x_0 of the Cartesian grid. Here δx_j is a vector of length $|\delta x_j| > 0$ that points from the Cartesian grid-point x_0 in the jth coordinate direction (Fig. 1). Computational improvements arising from the use of the auxiliary grid over simply using the nearest points of the main grid were reported in Farazmand and Haller [17]. Experience suggests setting the auxiliary grid spacing to 1–10% of the main grid spacing.

The function `eig_cgStrain` of LCS Tool provides the option to calculate Cauchy-Green eigenvectors from the auxiliary grid using eigenvalues calculated from the main grid. We have found that for analytic flows, the eigenvalues can be calculated from the main grid. For dataset flows, using the auxiliary grid for eigenvalue calculations gives better results.

As stated above, a typical main grid for the Cauchy-Green strain tensor has 500×500 points. This means that after the addition of 4

Table 1
Overview of sequence of computations to detect LCSs with LCS Tool functions.

1.	Define a velocity field, `dx = derivative(t,x,p)`
2.	Compute Cauchy-Green strain tensor invariants, `[v,d]= eig_cgStrain(derivative)`
3.	Define a range of λ values, `lambda`
4.	Define Poincare sections, `ps`
5.	Detect elliptic LCSs, `closedLambdaLine = poincare_closed_orbit_range(v,d,lambda,ps)` and `ellipticLcs = elliptic_lcs(closedLambdaLine)`
6.	Define a local maximization/minimization distance for hyperbolic LCSs, `lmd`
7.	Detect hyperbolic LCSs, `hyperbolicLine = seed_curves_from_lambda_max(v,d,ellipticLcs,lmd)` and `hyperbolicLcs = remove_strain_in_elliptic(hyperbolicLine,ellipticLcs)`

Table 2
Algorithm to calculate the invariants of the Cauchy-Green strain tensor field.

1.	Define a Cartesian grid for initial conditions of trajectories. Define an auxiliary grid for differentiating with respect to initial conditions.
2.	Solve Eq. (1) starting from each grid point and auxiliary grid point over the time interval $[t_0, t]$. This gives a discrete approximation to the flow map $F_{t_0}^t(x_0)$.
3.	Use finite differencing over the auxiliary grid to compute numerically the derivative of the flow map $DF_{t_0}^t(x_0)$.
4.	Compute the Cauchy-Green strain tensor field $C_{t_0}^t(x_0) = \left(DF_{t_0}^t(x_0)\right)^T DF_{t_0}^t(x_0)$, its eigenvalue field $\lambda_{1,2}(x_0)$, and eigenvector fields $\xi_{1,2}(x_0)$ over the initial condition grid.

Table 3
Syntax of the function `eig_cgStrain`.

`[cgEigenvector,cgEigenvalue] = eig_cgStrain(derivative,domain,timespan,resolution)`	
`derivative`	function handle for flow velocity equations
`domain`	2×2 array to define flow domain
`timespan`	1×2 array to define flow timespan
`resolution`	1×2 array to define Cauchy-Green strain main grid resolution
`auxGridRelDelta`	optional scalar between 0 and 0.5 to specify auxiliary grid spacing. Default: 10^{-2}.
`eigenvalueFromMainGrid`	optional logical to control whether eigenvalues of Cauchy-Green strain are calculated from main or auxiliary grid. Default: `true`.
`incompressible`	optional logical to specify if incompressibility is imposed. Default: `false`.
`odeSolverOptions`	optional `odeset` structure to specify flow map integration parameters

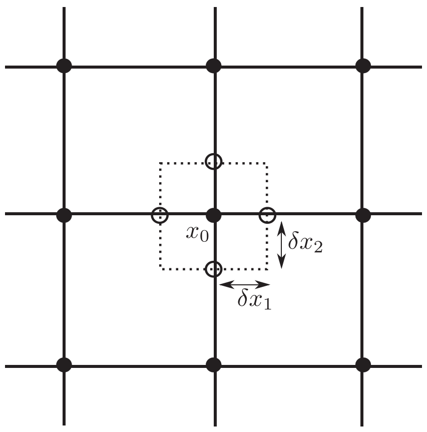

Fig. 1. Illustration of the main grid (filled circles) and the auxiliary grid (empty circles) used in the computation of the derivative of the flow map in Eq. (2) of the Cauchy-Green strain tensor. The variable `auxGridRelDelta` specifies the grid spacing of the auxiliary grid relative to the main grid spacing.

Table 4
Runtime and memory use associated with vector integration at different resolutions. Results are from the function `eig_cgStrain` applied to the double gyre presented in Section 4.1. Results obtained with an Intel Core i3-4130 3.4 GHz processor.

Resolution	Runtime (s)	Memory (MB)
500×251	29	55
750×376	63	123
1000×501	112	223

auxiliary grid points around each main grid point, Eq. (1) must be integrated over 1.25 million initial conditions.

To avoid excessive computational times in MATLAB, we vectorize Eq. (1), i.e., combine its right hand side evaluated over each initial point into a single system of equations. The resulting system is composed of independent blocks of two-dimensional first-order ordinary differential equations. We then use MATLAB's `ode45` function to perform trajectory integration from all grid points simultaneously. This calculation typically takes 5–10 min. A potential drawback of vector form integration is that memory requirements may become excessive at high resolutions. Table 4 lists runtime and memory use to integrate a flow map at different resolutions. Furthermore, writing the velocity function in vector form is more error-prone than the simpler two-dimensional form.

The Cauchy-Green strain tensor can have point singularities, i.e., points where $C_{t_0}^t(x_0)$ has repeated eigenvalues. At these points the eigenvectors $\xi_1(x_0)$ and $\xi_2(x_0)$ are no longer well-defined. This generically arises at a finite set of isolated points within the computational domain[28], and hence lie off the computational grid with probability one.

Table 5
Algorithm to calculate elliptic LCSs and coherent Lagrangian vortex boundaries.

1.	Position Poincare sections in flow domain to specify initial positions of lambda-lines
2.	Integrate λ-lines tangent to $\eta_\pm^\lambda$ (see Eq. (5))
3.	Calculate Poincare map
4.	Find closed orbits for fixed points of the Poincare map
5.	Identify outermost closed orbit on each Poincare section

3.1.1. *Special case: incompressible velocity fields*

Incompressible flows (i.e. those satisfying $\nabla \cdot v = 0$) satisfy the relation $\lambda_1(x_0)\lambda_2(x_0) = 1$ at all points of the computational domain [29]. Incompressibility can be computationally imposed by first calculating $\lambda_2(x_0)$, then setting $\lambda_1(x_0) = 1/\lambda_2(x_0)$ and calculating the strain eigenvectors ξ_2 from λ_2, then ξ_1 from the relationship in Eq. (3). Experience shows that computing λ_i in this order gives higher accuracy than in the reverse order [17].

At some grid points, $\lambda_2 < 1$ may occur due to numerical integration errors. By setting the integration tolerances to smaller values, the number of such grid points is reduced. Enforcing $\lambda_2 \geq 1$ everywhere, however, can incur excessive computational cost. To this end, the function `eig_cgStrain` records the number of points with $\lambda_2 < 1$, providing a measure for setting feasible integration tolerances.

3.1.2. *Special case: dataset velocity fields*

Velocity fields defined by datasets require preprocessing before they are used in the numerical integration of Eq. (1). This requires spatial and temporal interpolation that enables the evaluation of the velocity function at arbitrary points in U and at arbitrary times between t_- and t_+. In Section 4.3, we present an ocean dataset example with details of possible interpolation functions.

3.2. *Computing elliptic LCSs*

As discussed in Section 2.1, positions of elliptic LCSs at time t_0 are found as closed orbits of the $\eta_\pm^\lambda$ vector fields defined in Eq. (4). We find such orbits by integrating (5) from points of an appropriately chosen section (Poincare section), and evaluating the first return map (Poincare map) onto this section. A λ-line returning to its starting point is then an elliptic LCS. The outermost member of a family of closed λ-lines (obtained by varying λ) is a coherent Lagrangian vortex boundary [19,20].

The main steps in calculating elliptic LCSs are enumerated in Table 5 and described in further detail below. The syntax of elliptic LCS functions in LCS Tool is shown in Table 6.

The first step is to define the position of Poincare sections in regions where closed λ-lines are expected based on a visual analysis of the orbit structure of the $\eta_\pm^\lambda$ vector field. The Poincare section is to be oriented such that the first endpoint is close to the centre of the

Table 6
Syntax for LCS Tool elliptic LCS functions. Both functions are called by the function `poincare_closed_orbit_range`.

`[shearline.etaPos,shearline.etaNeg] = lambda_line(cgEigenvector,cgEigenvalue,lambda)`	
`cgEigenvector`	array of Cauchy-Green strain eigenvectors
`cgEigenvalue`	array of Cauchy-Green strain eigenvalues
`lambda`	scalar lambda value in Eq. 4
`[closedOrbits,orbits] = poincare_closed_orbit_multi(domain,resolution,shearline,PSList)`	
`domain`	array to define flow domain
`resolution`	1×2 array to define main grid resolution for Cauchy-Green strain tensor
`shearline`	structure of arrays of η_+ and η_- values on main grid
`PSList`	user-defined structure for Poincare section end-points, number of λ-lines launched from Poincare section, and maximum closed λ-line length
`nBisection`	optional number of bisection steps to refine zero crossings of Poincare map. Default: 5.
`dThresh`	optional threshold to discard discontinuous zero crossings of Poincare map. Default: 10^{-2}.
`odeSolverOptions`	optional `odeset` structure to specify λ-line integration parameters
`periodicBc`	optional 1×2 logical array to specify periodic boundary conditions. Default: `[false,false]`.

Table 7
Algorithm used for variable time step integration of λ-lines.

1.	Linearly interpolate vector field orientation at initial position.
2.	At next position, check whether vector field has rotated by over 90°, if yes, flip the vector field orientation by 180°.
3.	Stop integration when λ-line returns to Poincare section, λ-line reaches the domain boundary, or maximum integration length has been reached.

expected Lagrangian vortex, and the second endpoint is outside this vortex. Additionally, the number of lambda-lines launched from the Poincare section, `poincareSection.numPoints`, must be defined. A reasonable default value is 100.

The second step is to integrate the λ-lines starting from the Poincare section to obtain the corresponding Poincare map. Integration of λ-lines is performed using the $\eta_{\pm}^{\lambda}$ vector fields defined in Eq. (4) over the main grid. The underlying eigenvector fields, $\xi_i(x_0)$, have generic but removable orientation discontinuities, which require monitoring and local reorientation. This process is sketched in Table 7 and illustrated in Fig. 2. Linear interpolation is used in the interpolation of $\eta_{\pm}^{\lambda}$ within a grid element, since using higher-order interpolation would necessitate verifying that there are no orientation discontinuities beyond the four nearest grid points. We identify orientation discontinuities by checking the inner product of the $\eta_{\pm}^{\lambda}$ vectors at adjacent grid points. Rotations exceeding 90°, between two such neighbouring vectors are classified as orientation discontinuities and are corrected before linear interpolation. When setting the Cauchy-Green strain tensor main grid resolution, one may find it helpful to calculate a histogram of eigenvector field rotations to ensure that all rotations are well below 90° or almost 180°.

An example of a Poincare map produced from integration of the $\eta_{\pm}^{\lambda}$ field is shown in Fig. 3. Most orbits will return to the Poincare section and their integration will then be stopped using the ordinary differential equation event detection function of MATLAB. Some orbits may, however, deviate far from the Poincare section and do not return for any reasonable integration time. To control this behaviour, we specify a maximum orbit length, `poincareSection.orbitMaxLength`. In practice, viewing the Poincare section as the radius of a circle and setting the maximum λ-line integration length to twice the circumference gives good results.

In Fig. 3, circle markers indicate fixed points of the Poincare map, i.e., points where the distance between the final and the initial point of the orbit $P(s) - s$, is zero. The function `poincare_closed_orbit_multi` performs the computations. As seen in Fig. 3, not all zero crossings have circles. This is because LCS Tool uses a filtering parameter, `dThresh`, to discard sign changes of $P(s) - s$ that are likely due to numerical sensitivity or a

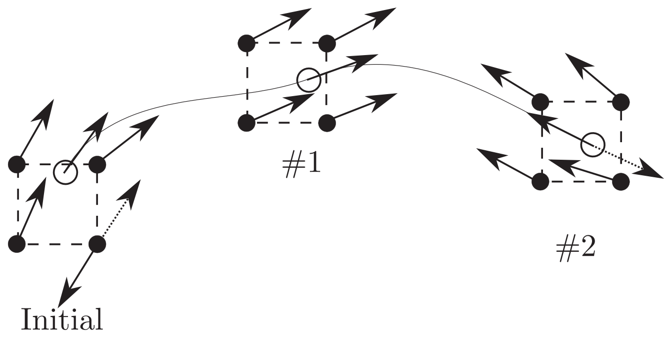

Fig. 2. Schematic illustration of the variable-time-step λ-line integration. At the initial point, there is an orientation discontinuity at the lower-right grid point that must be corrected prior to linear interpolation. At point #1, no orientation discontinuity is present. At point #2, $\eta_{\pm}^{\lambda}$ vectors must be rotated by 180° to match the orientation of the trajectory.

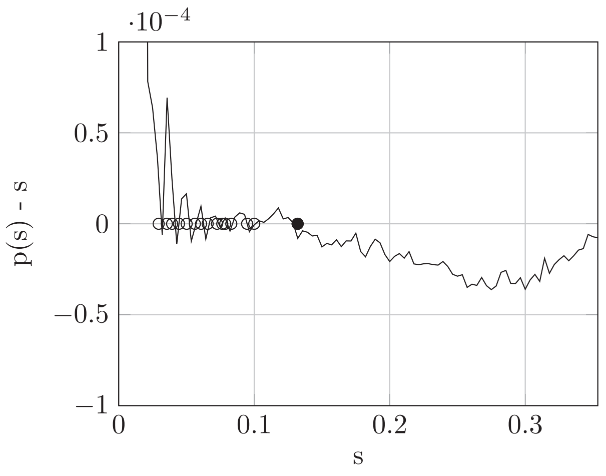

Fig. 3. Example of a Poincare map obtained for an elliptic LCS. The abscissa, s, represents distance along Poincare section. The ordinate, $p(s) - s$, represents distance of Poincare orbit return point from initial position. Circle markers indicate closed orbit positions. The filled circle indicates the outermost fixed point of the Poincare map, marking the intersection of a coherent Lagrangian vortex boundary with the Poincare section.

jump discontinuity of the Poincare map. Specifically, the location of each detected zero crossing is first refined by the bisection method. If after a predetermined number of iterations, `nBisection`, the two points around the zero crossing still have absolute values above `dThresh`, the zero crossing is discarded. Once all valid closed λ-lines have been located, the outermost closed λ-orbit associated with every Poincare section is identified as a coherent Lagrangian vortex boundary.

3.3. *Computing hyperbolic LCSs*

As discussed in Section 2.2, hyperbolic LCS positions at time t_0 are found as the strongest repelling curves of the vector field Eq. (6) (repelling LCSs), and strongest attracting curves of the vector field Eq. (7) (attracting LCSs). By repulsion and attraction we mean a property of the LCS (as an evolving material line) under the flow map $F_{t_0}^{t}$. We identify the strongest repelling shrink lines as those crossing a local maximum of the $\lambda_2(x_0)$ field. Similarly, we identify the strongest attracting stretch lines as those crossing a local minimum of the $\lambda_1(x_0)$ field. These local maxima and minima of the appropriate $\lambda_i(x_0)$ eigenvalue field can be thought of as the extensions of the concept of saddle points to the present finite-time, temporally aperiodic flow setting.

The main steps of hyperbolic LCS detection are enumerated in Table 8. The core function to compute hyperbolic LCSs is `seed_curves_from_lambda_max` and its syntax is given in Table 9.

4. Examples

This section presents the use of LCS Tool in three examples: a double gyre, a jet, and an oceanic geostrophic flow. The examples are available as scripts in the demo folder of LCS Tool. Executing these scripts demonstrates how to call LCS Tool functions. These functions are written to achieve balance between ease of use and performance. Examples of demo runtimes are given in Table 10.

4.1. *Double gyre*

The double gyre is a model for a time-dependent two gyre system observed in geophysical flows [30]. The model consists of two counter rotating sinusoidal vortices with a harmonically

Table 8
Algorithm to calculate initial positions of repelling LCSs at time t_0. The algorithm for attracting LCSs is similar.

1.	Define a local maximization distance.
2.	Find all points of the main grid that are local maxima of λ_2 within a circle whose radius is the local maximization distance.
3.	Define a maximum shrink line length.
4.	Integrate a shrink line forward and backward according to Eq. (6) and using the largest λ_2 local maximum as the initial position. Integrate until the shrink line has attained the maximum shrink line length, or until it has reached the domain boundary.
5.	Flag any remaining local maxima of λ_2 within the maximization distance of the shrink line as ineligible initial positions for subsequent shrink lines.
6.	Continue integrating shrink lines using local maxima of λ_2 as initial positions until no eligible local maxima of λ_2 remain.
7.	Remove all shrink line segments within elliptic LCSs.

Table 9
Syntax of the function `seed_curves_from_lambda_max`.

```
[curvePosition,curveInitialPosition] = seed_curves_from_lambda_max(distance,cgEigenvalue,cgEigenvector,flowDomain, flowResolution)
```

`distance`	threshold distance for placement of $\lambda_2(x_0)$ maxima
`cgEigenvalue`	array of Cauchy-Green strain eigenvalues
`cgEigenvector`	array of Cauchy-Green strain eigenvectors
`flowDomain`	2×2 array to define flow domain
`flowResolution`	1×2 array to define Cauchy-Green strain main grid resolution
`periodicBc`	optional 1×2 logical array to specify periodic boundary conditions. Default: `[false,false]`.
`nMaxCurves`	optional maximum number of curves (i.e. shrink lines or stretch lines) to generate. Default: `numel(cgEigenvalue)`.
`odeSolverOptions`	optional `odeset` structure to specify flow map integration parameters

Table 10
Runtime of some demo scripts. Results obtained with an Intel Core i3-4130 3.4 GHz processor with 8 GB of memory.

Demo	Runtime (min)
Double gyre	50
Bickley jet	25
Ocean dataset	127

Listing 1
Double gyre derivative function corresponding to Eq. (8).

```
function derivative_ = derivative(t,x,ϵ,A,ω)
...
a = ϵ*sin(ω*t);
b = 1 - 2*ϵ*sin(ω*t);
f = a*x(idx1).^2 + b*x(idx1);

derivative_(idx1) = -π*A*sin(π*f).*cos(π*x(idx2));
derivative_(idx2) = π
    *A*cos(π*f).*sin(π*x(idx2)).*(2*a*x(idx1) + b);
```

oscillating line in-between. Lagrangian particle motions satisfy the non-autonomous dynamical system

$$\begin{aligned}
\frac{dx}{dt} &= -\pi A \sin[\pi f(x,t)]\cos(\pi y), \\
\frac{dy}{dt} &= \pi A \cos[\pi f(x,t)]\sin(\pi y)\frac{\partial f(x,t)}{\partial x}, \\
f(x,t) &= \epsilon \sin(\omega t)x^2 + [1-2\epsilon\sin(\omega t)]x.
\end{aligned} \tag{8}$$

The MATLAB function describing this velocity field is given in Listing 1, specifying the right hand side of the particle ordinary differential equation in a way that supports vectorized integration.

In what follows, the parameter values are: $A = 0.1$, $\epsilon = 0.1$, $\omega = \pi/5$. The flow timespan is $t \in [0, 10]$ and the domain is $x \in [0, 2]$, $y \in [0, 1]$. By examining the FTLE field (which LCS Tool can provide), we position Poincare sections to capture elliptic LCSs. A script to perform this operation is given in Listing 2 where two Poincare sections are

Listing 2
LCS Tool commands for double gyre elliptic LCSs. Abridged subset of the LCS Tool script `demo/double_gyre/elliptic_hyperbolic_lcs.m`.

```
   %% Input parameters
 2 ...
   domain = [0,2;0,1];
 4 res = [500,250];
   time = [0,10];
 6
   %% Velocity definition
 8 lDerivative = @(t,x)derivative(t,x,ϵ,A,ω);

10 %% LCS parameters
   cgOptions = odeset('relTol',1e-5);
12
   % Lambda lines
14 ...
   ps(1).endPosition = [.55,.55;.1,.1];
16 ps(2).endPosition = [1.53,.45;1.95,.05];
   ...
18 lambda = .93:.01:1.07;
   llOptions = odeset('relTol',1e-6);
20
   %% Cauchy-Green strain eigenvalues and eigenvectors
22 [cgV,cgD] = eig_cgStrain(lDerivative,domain,res,time,
       'odeSolverOptions',cgOptions);

24 %% Elliptic LCSs
   [closedLlp,closedLln] =
       poincare_closed_orbit_range(domain,res,cgV,cgD,
       lambda,ps,'odeSolverOptions',llOptions);
26 ellipticLcs = elliptic_lcs(closedLlp);
   ellipticLcs = [ellipticLcs,elliptic_lcs(closedLln)];
```

defined (lines 15 and 16). The free stretching parameter λ (Eq. (5)) is varied over the range [0.93, 1.07] with increments of 0.01 (line 18). Closed orbits for all predefined λ values are computed and the outermost closed orbit is kept as the Lagrangian vortex boundary

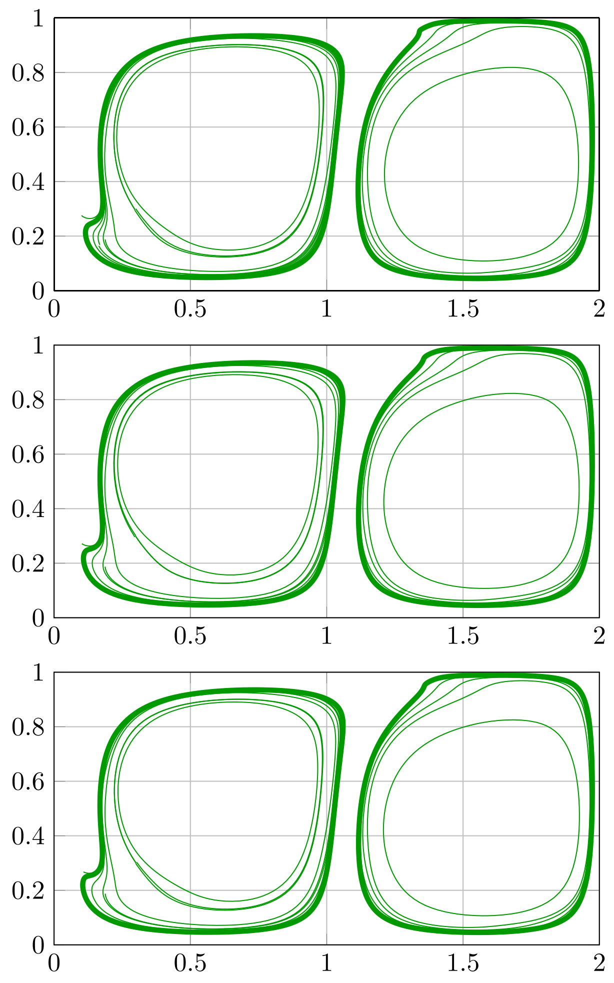

Fig. 4. Convergence of closed λ-lines for increasing main-grid resolution in the double-gyre. The outermost closed λ-line (bold green line) is the vortex boundary. Top: 500 × 250, middle: 750 × 375, bottom: 1000 × 500. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(lines 25–27). We have chosen small values for the error tolerances of the integration of the Cauchy-Green strain tensor (line 11) and λ-lines (line 19) to ensure convergence.

In Fig. 4, the resolution of the Cauchy-Green strain tensor is varied from 500 × 250 to 1000 × 500. The location of the outermost closed λ-line changes insignificantly, demonstrating convergence. For all tested resolutions, the λ values of the outermost closed orbits match. This also suggests that the lowest tested resolution, 500 × 250, is sufficient to identify elliptic LCSs.

Fig. 5 shows elliptic and hyperbolic LCSs for the double gyre at this resolution. Listing 3 gives the LCS Tool commands to compute hyperbolic LCSs. The maximum length of shrink lines and stretch lines, `shrinkLineMaxLength` and `stretchLineMaxLength`, is set to 20, a multiple of the domain size since the hyperbolic LCSs may wind around vortices several times.

As usual, the local maximization distance is set larger for stretch lines than for shrink lines (cf. line 3 and line 7). The purpose of the maximization distance is to obtain spatially separated LCSs and to avoid a dense tangle of lines that basically indicates the same hyperbolic LCS (cf. Table 8, item 5). Setting the local maximization distance for stretch lines larger than for shrink lines allows obtaining a comparable number of stretch lines and shrink lines overall in the flow domain (recall that hyperbolic LCS seed points are discarded if they are within the local maximization distance of an existing hyperbolic LCS). Shrink lines are locally tangent to ridges of

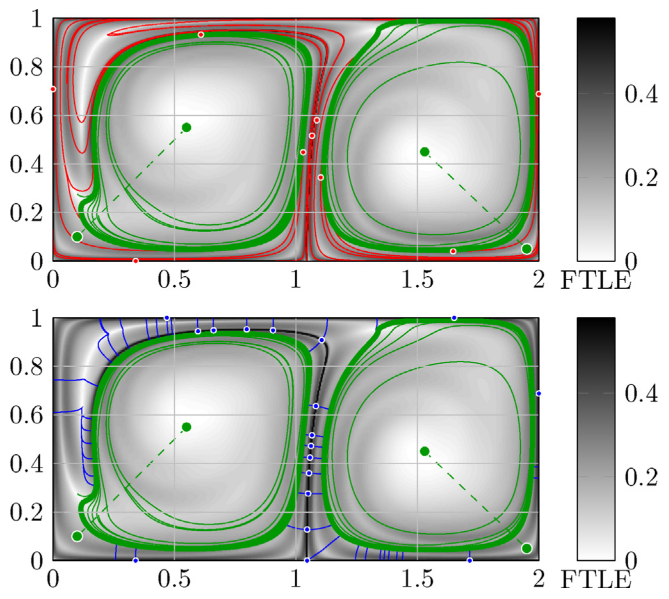

Fig. 5. LCSs in the double gyre. Resolution is 500 × 250. Elliptic LCSs are green, shrink line LCSs are red and stretch line LCSs are blue. White dots indicate λ_2 maxima for shrink lines and λ_1 minima stretch lines. FTLE shown in the background. $\lambda = 1.00$ and 1.04 for the left and the right gyre. $\lambda \in [0.93, 1.07]$, $\Delta\lambda = 0.01$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Listing 3
LCS Tool commands for double gyre hyperbolic LCSs. Abridged subset of the LCS Tool script `demo/double_gyre/elliptic_hyperbolic_lcs.m`.

```
   % Shrink lines
 2 shrinkLMaxLength = 20;
   shrinkLMaxDistance = 2*gridSpace;
 4
   % Stretch lines
 6 stretchLMaxLength = 20;
   stretchLMaxDistance = 10*gridSpace;
 8
   %% Hyperbolic shrink line LCSs
10 shrinkL = seed_curves_from_lambda_max(
       shrinkLMaxDistance,shrinkLMaxLength,cgV(:,2),
       cgD(:,1:2),domain,res);
   ...
12 %% Hyperbolic stretch line LCSs
   stretchL = seed_curves_from_lambda_max(
       stretchLMaxDistance,stretchLMaxLength,-cgV(:,1),
       cgD(:,3:4),domain,res);
14 ...
```

λ_2 maxima, whereas stretch lines are locally normal to these ridges. Setting the local maximization distance of stretch lines and shrink lines equal would therefore produce a greater number of stretch lines than shrink lines.

4.2. Bickley jet

The Bickley jet models a meandering zonal jet flanked above and below by counter rotating vortices. This is an idealized model of geophysical flows such as the Gulf Stream and the polar night jet perturbed by a Rossby wave [31,16].

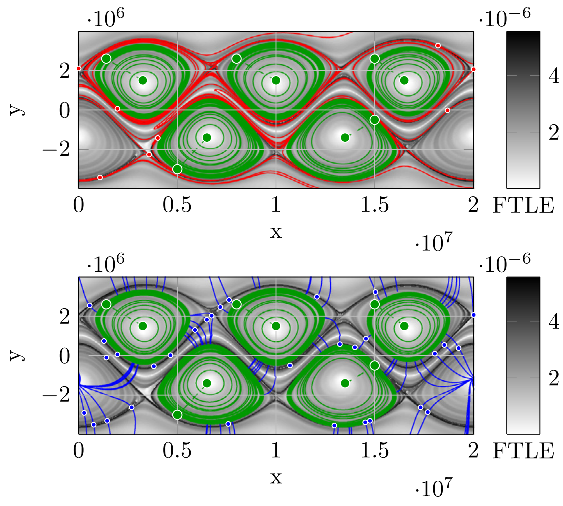

Fig. 6. LCSs in the Bickley jet. Resolution is 500 × 200. Elliptic LCSs are green, shrink line LCSs are red and stretch line LCSs are blue. White dots indicate λ_2 maxima for shrink lines and λ_1 minima stretch lines. FTLE shown in the background. λ values of elliptic LCSs from left to right are [0.95, 0.80, 0.94, 0.80, 0.94], $\lambda \in [0.80, 1.20]$, $\Delta\lambda = 0.01$.

The velocity is given by $v(x, y, t) = (-\partial_y \psi, \partial_x \psi)$ where

$$\psi(x, y, t) = \psi_0(x, y) + \psi_1(x, y, t),$$
$$\psi_0(x, y) = c_3 y - UL_y \tanh \frac{y}{L_y} + \epsilon_3 UL_y \operatorname{sech}^2 \frac{y}{L_y} \cos k_3 x,$$
$$\psi_1(x, y, t) = UL_y \operatorname{sech}^2 \frac{y}{L_y} \Re \left[\sum_{n=1}^{2} \epsilon_n f_n(t) e^{ik_n x} \right].$$

As a forcing function, we choose a solution running on the chaotic attractor of the damped and forced Duffing oscillator, specifically

$$\frac{d\phi_1}{dt} = \phi_2,$$
$$\frac{d\phi_2}{t} = -0.1\phi_2 - \phi_1^3 + 11\cos(t),$$
$$f_{1,2}(t) = 2.625 \times 10^{-2} \phi_1(t/6.238 \times 10^5)$$

The parameter values we use are: $U = 62.66$, $c_2 = 0.205U$, $c_3 = 0.461U$, $L_y = 1.77 \times 10^6$, $\epsilon_1 = 0.0075$, $\epsilon_2 = 0.04$, $\epsilon_3 = 0.3$, $L_x = 6.371 \times 10^6 \pi$, $k_n = 2n\pi/L_x$, $\sigma_1 = 0.5k_2(c_2 - c_3)$, $\sigma_2 = 2\sigma_1$.

The integration time is $T = 4L_x/U$, an integer multiple of the eddy turnover time. Listing 4 shows the LCS Tool commands in which the chaotically perturbed velocity is defined (line 9), periodic boundary conditions are imposed in the x-direction (line 10), and five Poincare sections are defined where we expect coherent vortices (lines 26–30). λ-values for closed orbit detection are varied over the range [0.80, 1.20] with a step of 0.01.

Fig. 6 shows elliptic and hyperbolic LCSs of the Bickley jet with the FTLE in the background.

Listing 4
LCS Tool commands for Bickley jet elliptic and hyperbolic LCSs. Abridged version of the LCS Tool script `demo/bickley_jet/elliptic_hyperbolic_lcs.m`.

```
   %% Input parameters
 2 ...
   domain = [0,Lx;[-1,1]*2.25*Ly];
 4 res = [500,276];
   time = [0,4*Lx/u];
 6
   %% Velocity definition
 8 ...
   lDerivative = @(t,x)derivative(t,x,u,Lx,Ly,ϵ,
         perturbationCase,phiSol,phi1Max);
10 pBc = [true,false];

12 %% LCS parameters
   % Lambda lines
14 lambda = .8:.01:1.1;
   ...
16 % Shrink lines
   shrinkLMaxLength = 1e8;
18 shrinkLMaxDistance = 8*gridSpace;
   ...
20 % Stretch lines
   stretchLMaxLength = 1e8;
22 stretchLMaxDistance = 4*gridSpace;
   ...
24 %% Lambda line LCSs
   ...
26 ps(1).endPosition = [3.25e6,1.5e6;1.4e6,2.6e6];
   ps(2).endPosition = [6.5e6,-1.4e6; 5.e6,-3.e6];
28 ps(3).endPosition = [1e7,1.5e6;8e6,2.6e6];
   ps(4).endPosition = [1.35e7,-1.4e6;1.5e7,-.5e6];
30 ps(5).endPosition = [1.65e7,1.5e6;1.5e7,2.6e6];
   ...
32 %% Cauchy-Green strain eigenvalues and eigenvectors
   [cgV,cgD] = eig_cgStrain(lDerivative,domain,res,time);
34 %% Elliptic LCSs
   [closedLlp,closedLln] = poincare_closed_orbit_range(
         domain,res,cgV,cgD,lambda,ps,'periodicBc',pBc);
36 ...
   %% Hyperbolic shrink line LCSs
38 shrinkL =
         seed_curves_from_lambda_max(shrinkLMaxDistance,
         shrinkLMaxLength,cgV(:,2),cgD(:,1:2),domain,res,
         'periodicBc',pBc);
   ...
40 %% Hyperbolic stretch line LCSs
   stretchL = seed_curves_from_lambda_max(
         stretchLMaxDistance,stretchLMaxLength,-cgV(:,1),
         cgD(:,3:4),domain,res,'periodicBc',pBc);
42 ...
```

4.3. Ocean velocity data from satellite altimetry

The final example demonstrates the use of LCS Tool on velocity data derived from satellite-observed sea-surface heights under the geostrophic approximation. In contrast to the previous two analytic examples, the velocity field is available only with discrete temporal and spatial resolution. Our region of interest is a small domain in the South Atlantic Ocean, where exceptionally coherent eddies, Agulhas rings, were recently found by Haller and Beron-Vera [19,20] using the theory we surveyed in Section 2.1.

In the geostrophic approximation, sea surface height (η) serves as a stream-function for surface velocities. In a longitude-latitude (φ, θ) coordinate system, the evolution of a fluid particle is given by

$$\frac{\partial\varphi(\varphi, \theta, t)}{\partial t} = -\frac{g}{R^2 f(\theta)\cos\theta} \frac{\partial\eta(\varphi, \theta, t)}{\partial\theta} \tag{9}$$

$$\frac{\partial\theta(\varphi, \theta, t)}{\partial t} = \frac{g}{R^2 f(\theta)\cos\theta} \frac{\partial\eta(\varphi, \theta, t)}{\partial\varphi} \tag{10}$$

where g is the constant of gravity, R is the mean radius of the Earth, and $f(\theta) \equiv 2\Omega \sin\theta$ is the Coriolis parameter, with Ω denoting the Earth's mean angular velocity.

Listing 5
LCS Tool commands for ocean data elliptic and hyperbolic LCSs. Abridged version of the LCS Tool script `demo/ocean_dataset/elliptic_hyperbolic_lcs.m`.

```
   % Input parameters
 2 domain = [0,6;-34,-28];
   res = [400,400];
 4 time = [100,130];

 6 % Velocity definition
   load('ocean_geostrophic_velocity.mat');
 8 ...
   intMet = 'spline';
10 vLonI = griddedInterpolant({time,lat,lon},vLon,intMet);
   vLatI = griddedInterpolant({time,lat,lon},vLat,intMet);
12 lDerivative =
       @(t,x,~)flowdata_derivative(t,x,vLonI,vLatI);

14 % LCS parameters
   % Cauchy-Green strain
16 cgVmg = false;
   cgAgd = .01;
18
   % Lambda lines
20 ...
   ps(1).endPosition = [3.3,-32.1;3.7,-31.6];
22 ...
   lambda = .9:.02:1.1;
24 llOptions = odeset('relTol',1e-6);
   ...
26 % Shrink lines
   shrinkLMaxLength = 20;
28 shrinkLMaxDistance = 2*gridSpace;
   ...
30 % Stretch lines
   stretchLMaxLength = 20;
32 stretchLMaxDistance = 4*gridSpace;
   ...
34 %% Cauchy-Green strain eigenvalues and eigenvectors
   [cgV,cgD] = eig_cgStrain(lDerivative,domain,res,time,
       'eigenvalueFromMainGrid',cgVmg,
       'auxGridRelDelta',cgAgd);
36 %% Elliptic LCSs
   [closedLlp,closedLln] =
       poincare_closed_orbit_range(domain,res,cgV,cgD,
       lambda,ps,'odeSolverOptions',llOptions);
38 ...
   % Hyperbolic shrink line LCSs
40 shrinkL = seed_curves_from_lambda_max(
       shrinkLMaxDistance,shrinkLMaxLength,cgV(:,2),
       cgD(:,1:2),domain,res);
   ...
42 % Hyperbolic stretch line LCSs
   stretchL = seed_curves_from_lambda_max(
       stretchLMaxDistance,stretchLMaxLength,-cgV(:,1),
       cgD(:,3:4),domain,res);
```

The data is given at a spatial resolution of 1/4° and a temporal resolution of 7 days. Due to the discrete data, defining the right hand side of Eqs. (9) and (10) involves spline interpolation in space and time. An interpolant is generated first, then the function `flowdata_derivative` evaluates the interpolants for the zonal and meridional velocity at the needed coordinates. Listing 5 shows the relevant part of the code in LCS Tool's ocean demo file `demo/ocean_dataset/elliptic_hyperbolic_lcs.m`. The commands for the interpolation of the velocity data are given in lines 9–12.

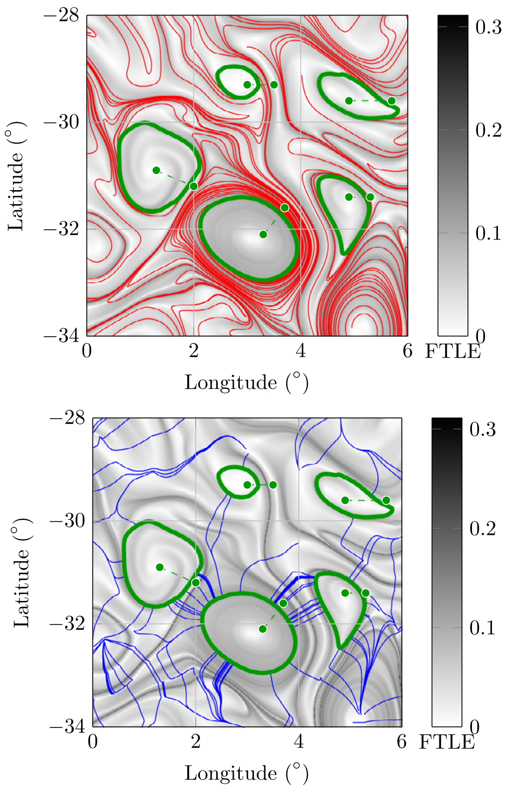

Fig. 7. LCSs in the ocean velocity data from satellite altimetry. Resolution is 400 × 400. Elliptic LCSs are green, shrink line LCSs are red and stretch line LCSs are blue. White dots indicate λ_2 maxima for shrink lines and λ_1 minima stretch lines. FTLE shown in the background. λ = 1.00, 1.08, 0.94, 0.90, 1.06.

We choose the integration time as T = 30 days (listing 5, line 4), which is larger than the eddy turnover time in this region. The resolution of the main computational grid for initial conditions is set to 400 × 400 (line 3). This corresponds to a resolution of roughly 0.015°. With this choice, the resolution of the tracer grid is 15 times higher than the resolution of the velocity field. The flow is integrated and the Cauchy-Green strain tensor is computed by the function `eig_cgStrain` (line 35). The auxiliary grid distance is set to 1% of the main grid distance (line 17), and eigenvalues are computed from the auxiliary grid (line 16). Elliptic LCSs are computed in line 37, after the Poincare sections have been set (line 21). λ-values are varied over a range of [0.90, 1.10] with a step of 0.02 (line 23). Hyperbolic LCSs are computed in lines 40 and 43.

Fig. 7 shows elliptic and hyperbolic LCSs on 22 November 2006, the same time as analysed in Haller and Beron-Vera [19,20], Beron-Vera et al. [4]. In those references, the integration time is 90 days. We use a shorter time to avoid tangling hyperbolic LCSs. Our analysis via LCS Tool reveals five coherent eddies. The largest, at (3, − 32), has a non-stretching boundary, i.e., λ = 1. It corresponds to eddy #2 in figure 3 of Beron-Vera et al. [4]. Four additional smaller coherent eddies are found. They do not stay coherent over 90 days. The hyperbolic LCSs determine the deformation of the fluid between coherent eddies.

5. Conclusions

We have described a computational toolbox, LCS Tool, that implements recent variational results for Lagrangian coherent structures (LCSs) in two-dimensional unsteady flows. We have also demonstrated the performance of LCS Tool on two analytic flow models and a geophysical velocity dataset. The publicly available software library producing these results enables the exploration of variational LCS methods without assuming a detailed knowledge of geodesic LCS theory.

LCS Tool leverages the capabilities of MATLAB. For FTLE based extraction of LCSs, computational performance has received considerable attention [32,33]. We think LCS Tool can facilitate similar computational advances for variational LCS methods. Optimizing computational performance will aid applications to large-scale forecasting applications, such as the tracking of environmental contaminants [34].

We think LCS Tool can serve as a foundation for the numerical implementation of recent theoretical advances. These include the geodesic theory of parabolic LCSs (jet cores) [23] and the variational theory of hyperbolic and elliptic LCSs for three-dimensional flows [22].

Acknowledgements

The altimeter products used in this work are produced by SSALTO/DUACS and distributed by AVISO, with support from CNES, http://www.aviso.oceanobs.com/duacs.

References

[1] G. Haller, G. Yuan, Lagrangian coherent structures and mixing in two-dimensional turbulence, Phys. D 147 (3–4) (2000) 352–370, http://dx.doi.org/10.1016/S0167-2789(00)00142-1.
[2] T. Peacock, G. Haller, Lagrangian coherent structures: the hidden skeleton of fluid flows, Phys. Today 66 (2) (2013) 41–47, http://dx.doi.org/10.1063/PT.3.1886.
[3] G. Haller, Lagrangian coherent structures, Annu. Rev. Fluid Mech. 47 (2015) 137–161, http://dx.doi.org/10.1146/annurev-fluid-010313-141322.
[4] F.J. Beron-Vera, Y. Wang, M.J. Olascoaga, G.J. Goni, G. Haller, Objective detection of oceanic eddies and the Agulhas leakage, J. Phys. Oceanogr. 43 (7) (2013) 1426–1438, http://dx.doi.org/10.1175/JPO-D-12-0171.1.
[5] T.-Y. Koh, B. Legras, Hyperbolic lines and the stratospheric polar vortex, Chaos 12 (2) (2002) 382–394, http://dx.doi.org/10.1063/1.1480442.
[6] M.M. Wilson, J. Peng, J.O. Dabiri, J.D. Eldredge, Lagrangian coherent structures in low Reynolds number swimming, J. Phys.: Condens. Matter 21 (20) (2009), http://dx.doi.org/10.1088/0953-8984/21/20/204105, 204105: 1–19.
[7] P. Tallapragada, S.D. Ross, D.G. Schmale, J. Burton III, Lagrangian coherent structures are associated with fluctuations in airborne microbial populations, Chaos 21 (3) (2011), http://dx.doi.org/10.1063/1.3624930, 33122: 1–16.
[8] F. Huhn, A. von Kameke, V. Pérez-Muñuzuri, M.J. Olascoaga, F.J. Beron-Vera, The impact of advective transport by the South Indian Ocean Countercurrent on the Madagascar plankton bloom, Geophys. Res. Lett. 39 (6) (2012), http://dx.doi.org/10.1029/2012GL051246, L06602: 1–6.
[9] W. Tang, P.W. Chan, G. Haller, Accurate extraction of Lagrangian coherent structures over finite domains with application to flight data analysis over Hong Kong International Airport, Chaos 20 (1) (2010), http://dx.doi.org/10.1063/1.3276061, 17502:1–8.
[10] E.S. Gawlik, J.E. Marsden, P.C. du Toit, S. Campagnola, Lagrangian coherent structures in the planar elliptic restricted three-body problem, Celest. Mech. Dyn. Astron. 103 (3) (2009) 227–249, http://dx.doi.org/10.1007/s10569-008-9180-3.
[11] S. Ali, M. Shah, A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis, IEEE Conf. Comput. Vis. Pattern Recognit. (2007) 1–6, http://dx.doi.org/10.1109/CVPR.2007.382977.
[12] A. Hadjighasem, M.M. Farazmand, G. Haller, Detecting invariant manifolds, attractors, and generalized KAM tori in aperiodically forced mechanical systems, Nonlinear Dyn. 73 (1–2) (2013) 689–704, http://dx.doi.org/10.1007/s11071-013-0823-x.
[13] G. Haller, Distinguished material surfaces and coherent structures in three-dimensional fluid flows, Phys. D 149 (4) (2001) 248–277, http://dx.doi.org/10.1016/S0167-2789(00)00199-8.
[14] G. Haller, A variational theory of hyperbolic Lagrangian coherent structures, Phys. D 240 (7) (2011) 574–598, http://dx.doi.org/10.1016/j.physd.2010.11.010.
[15] G. Norgard, P.-T. Bremer, Second derivative ridges are straight lines and the implications for computing Lagrangian coherent structures, Phys. D 241 (18) (2012) 1475–1476, http://dx.doi.org/10.1016/j.physd.2012.05.006.
[16] F.J. Beron-Vera, M.J. Olascoaga, M.G. Brown, H. Koçak, I.I. Rypina, Invariant-tori-like Lagrangian coherent structures in geophysical flows, Chaos 20 (1) (2010), http://dx.doi.org/10.1063/1.3271342, 17514:1–13.
[17] M.M. Farazmand, G. Haller, Computing Lagrangian coherent structures from their variational theory, Chaos 22 (1) (2012), http://dx.doi.org/10.1063/1.3690153, 13128:1–12.
[18] G. Haller, F.J. Beron-Vera, Geodesic theory of transport barriers in two-dimensional flows, Phys. D 241 (20) (2012) 1680–1702, http://dx.doi.org/10.1016/j.physd.2012.06.012.
[19] G. Haller, F.J. Beron-Vera, Coherent Lagrangian vortices: the black holes of turbulence, J. Fluid Mech. 731 (2013), http://dx.doi.org/10.1017/jfm.2013.391, R4:1–10.
[20] G. Haller, F.J. Beron-Vera, Addendum to 'Coherent Lagrangian vortices: the black holes of turbulence', J. Fluid Mech. 755 (2014), http://dx.doi.org/10.1017/jfm.2014.441, R3:1–4.
[21] M.M. Farazmand, G. Haller, Attracting and repelling Lagrangian coherent structures from a single computation, Chaos 23 (2) (2013), http://dx.doi.org/10.1063/1.4800210, 23101:1–11.
[22] D. Blazevski, G. Haller, Hyperbolic and elliptic transport barriers in three-dimensional unsteady flows, Phys. D 273–274 (2014) 46–62, http://dx.doi.org/10.1016/j.physd.2014.01.007.
[23] M.M. Farazmand, D. Blazevski, G. Haller, Shearless transport barriers in unsteady two-dimensional flows and maps, Phys. D 278–279 (2014) 44–57, http://dx.doi.org/10.1016/j.physd.2014.03.008.
[24] F. Lekien, Time-Dependent Dynamical Systems and Geophysical Flows (Ph.D. thesis), California Institute of Technology, 2003 http://resolver.caltech.edu/CaltechETD:etd-04082003-180353
[25] P.C. du Toit, Transport and Separatrices in Time-Dependent Flows (Ph.D. thesis), California Institute of Technology, 2010 http://resolver.caltech.edu/CaltechTHESIS:10072009-165901284
[26] S. Ameli, Y. Desai, S.C. Shadden, Development of an efficient and flexible pipeline for Lagrangian coherent structure computation, in: P.-T. Bremer, I. Hotz, V. Pascucci, R. Peikert (Eds.), Topological Methods in Data Analysis and Visualization, vol. 3, Springer, 2014, pp. 201–215, http://dx.doi.org/10.1007/978-3-319-04099-8_13.
[27] C.A. Truesdell, W. Noll, The Non-Linear Field Theories of Mechanics, 3rd ed., Springer, 2004, http://dx.doi.org/10.1007/978-3-662-13183-1.
[28] T. Delmarcelle, L. Hesselink, The topology of symmetric, second-order tensor fields, in: Proceedings of the IEEE Conference on Visualization '94, 1994, pp. 140–147, http://dx.doi.org/10.1109/VISUAL.1994.346326.
[29] V.I. Arnold, Mathematical Methods of Classical Mechanics Graduate Texts in Mathematics, vol. 60, Springer, 1989, http://dx.doi.org/10.1007/978-1-4757-2063-1.
[30] S.C. Shadden, F. Lekien, J.E. Marsden, Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows, Phys. D 212 (3–4) (2005) 271–304, http://dx.doi.org/10.1016/j.physd.2005.10.007.
[31] D. del Castillo-Negrete, P.J. Morrison, Chaotic transport by Rossby waves in shear flow, Phys. Fluids A 5 (4) (1993) 948–965, http://dx.doi.org/10.1063/1.858639.
[32] C. Conti, D. Rossinelli, P. Koumoutsakos, GPU and APU computations of finite time Lyapunov exponent fields, J. Comput. Phys. 231 (5) (2012) 2229–2244, http://dx.doi.org/10.1016/j.jcp.2011.10.032.
[33] P. Miron, J. Vétel, A. Garon, M. Delfour, M. El Hassan, Anisotropic mesh adaptation on Lagrangian coherent structures, J. Comput. Phys. 231 (19) (2012) 6419–6437, http://dx.doi.org/10.1016/jjcp.2012.06.015.
[34] M.J. Olascoaga, G. Haller, Forecasting sudden changes in environmental pollution patterns, Proc. Natl. Acad. Sci. U. S. A. 109 (13) (2012) 4738–4743, http://dx.doi.org/10.1073/pnas.1118574109.

Kristjan Onu was a postdoc researcher at McGill University, Canada. His research interests are fluid mechanics, numerical methods and stochastic dynamical systems.

Florian Huhn was a postdoc researcher at the Institute of Mechanical Systems at ETH Zurich, Switzerland. He is now at the Department of Experimental Methods, Institute of Aerodynamics and Flow Technology, German Aerospace Center (DLR), Göttingen, Germany. His main research interests are experimental fluid dynamics, reaction-diffusion advection systems, and transport in geophysical flows.

George Haller is Professor of Nonlinear Dynamics at ETH Zürich. He develops dynamical systems methods to handle complex problems in mechanics and fluid dynamics.