A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures

Doug Lipinski[1] and Kamran Mohseni[2,a)]
[1]*Department of Applied Mathematics, University of Colorado at Boulder, Boulder, Colorado 80309, USA*
[2]*Department of Aerospace Engineering Sciences, University of Colorado at Boulder, Boulder, Colorado 80309, USA*

(Received 10 September 2009; accepted 9 November 2009; published online 5 January 2010)

A ridge tracking algorithm for the computation and extraction of Lagrangian coherent structures (LCS) is developed. This algorithm takes advantage of the spatial coherence of LCS by tracking the ridges which form LCS to avoid unnecessary computations away from the ridges. We also make use of the temporal coherence of LCS by approximating the time dependent motion of the LCS with passive tracer particles. To justify this approximation, we provide an estimate of the difference between the motion of the LCS and that of tracer particles which begin on the LCS. In addition to the speedup in computational time, the ridge tracking algorithm uses less memory and results in smaller output files than the standard LCS algorithm. Finally, we apply our ridge tracking algorithm to two test cases, an analytically defined double gyre as well as the more complicated example of the numerical simulation of a swimming jellyfish. In our test cases, we find up to a 35 times speedup when compared with the standard LCS algorithm. © *2010 American Institute of Physics*. [doi:10.1063/1.3270049]

One of the biggest problems in fluid dynamics is analyzing data. As computational fluid dynamics (CFD) codes become more sophisticated and produce larger and more complex results for increasingly complicated flows, extracting meaningful results from the resulting fluid velocities becomes ever more challenging. Over the past several years, Lagrangian coherent structures (LCS) have emerged as an excellent way to visualize and analyze such flow fields. These structures represent barriers to transport and can be used to identify exact vortex boundaries, providing an unambiguous visual characterization of the flow field. They may also be used to extract quantitative measures of mixing and transport in the flow. However, the Lagrangian nature of LCS makes their computation extremely expensive, sometimes prohibitively so. Computations in two dimensions are extremely expensive and in three dimensions the problem is even worse. In this paper we present an algorithm which takes advantage of the temporal and spatial coherences of LCS to greatly speed computations. This algorithm tracks the ridges in the finite time Lyapunov exponent (FTLE) field at each time step, then approximates the location of the ridges at the next time step by advecting the LCS forward with the flow. A simple error estimate shows that the difference between the advected LCS and the actual LCS is typically very small. In the end, our algorithm proves to be very useful for extracting the major LCS present in a flow field and gives a speedup of up to 35 times.

[a)]Also at Department of Applied Mathematics, University of Colorado at Boulder, Boulder, Colorado 80309, USA. Electronic mail: mohseni@colorado.edu.

I. INTRODUCTION

LCS provide an effective way of visualizing many flow fields, both complex and simple, and have seen increasing use over the past several years. LCS were first proposed by Haller and Yuan[6] and their properties were further investigated by Shadden *et al.*[15] The well established properties of LCS as barriers to transport make them an excellent candidate for analyzing mixing and transport in fluid flows. They also establish unambiguous boundaries to vortices[2,14] and are relatively insensitive to small errors in the velocity field.[5] However, despite their increasing use by the fluid dynamics community, computational cost remains a significant barrier in many situations. The large cost of computing LCS is due to the Lagrangian nature of the structures. Following the method established by Shadden *et al.*,[15] computing the LCS requires advecting large numbers of particles at a high density in the flow to compute the FTLE field.

For completeness, we repeat several key definitions from Shadden *et al.*[15] here. The FTLE is defined as

$$\Phi_{t_0}^{t_0+T}(\mathbf{x}) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+T} \mathbf{v}(\mathbf{x}(t))dt, \tag{1}$$

$$\Delta = \left(\frac{d\Phi}{d\mathbf{x}}\right)^* \left(\frac{d\Phi}{d\mathbf{x}}\right), \tag{2}$$

$$\sigma_{t_0}^{T}(\mathbf{x}) = \frac{1}{|T|}\ln\sqrt{\lambda_{\max}(\Delta)}, \tag{3}$$

where T is the integration time, Φ is the flow map, Δ is the finite time Cauchy–Green deformation tensor, and σ is the FTLE.

The LCS are then defined as ridges in the FTLE field and may either be explicitly extracted or (more commonly) visualized by viewing a contour plot of the FTLE field. For our purposes, we will define a ridge as a second derivative ridge.

Definition: A second derivative ridge of σ is an injective curve $\mathbf{c}:(a,b)\rightarrow D$ satisfying the following conditions for all $s\in(a,b)$:

(1) The vectors $\mathbf{c}'(s)$ and $\nabla\sigma(\mathbf{c}(s))$ are parallel.
(2) $\mathrm{H}(\hat{\mathbf{n}},\hat{\mathbf{n}})=\min_{\|\mathbf{u}\|=1}\mathrm{H}(\mathbf{u},\mathbf{u})<0$, where $\hat{\mathbf{n}}$ is a unit normal vector to the curve $\mathbf{c}(s)$ and the Hessian of σ, H, is thought of as a bilinear form evaluated at the point $\mathbf{c}(s)$.

In the past, LCS have been used in many situations and computed from experimental data, CFD data, and analytical velocity fields. Examples of experimental data include high frequency radar data in Monterey Bay,[16] jellyfish swimming,[11] unsteady separation,[20] and two dimensional (2D) turbulence.[19] The output of CFD software has been used to compute LCS in turbulence,[4,10] vortex shedding behind an airfoil,[8] and jellyfish swimming.[9,21] Typically, in these examples it is desirable to compute the LCS at a few hundred time steps to visualize the time evolution of structures in the flow. Although LCS algorithms parallelize very well, the necessary computational time is often prohibitively long.

The nature of LCS, ridges in a field, makes this computation seem like a natural candidate for a more efficient algorithm. Since the ridges are the only part of the FTLE field which is of interest, any FTLE values which are computed away from the ridges are essentially wasted computational time. There are two broad classes of algorithms which seem well suited for this situation. The first is adaptive mesh refinement (AMR). By starting with a coarse grid of points on which the FTLE field is calculated and refining the grid only in areas where ridges are detected, it is possible to achieve very high resolution near the ridges with large computational savings when compared with a uniform mesh at the same resolution. There are many possible variations to this type of algorithm, but at its heart, it only requires a criterion to determine where the mesh should be refined. Two recent papers provide examples of such criteria.[3,12]

A second type of algorithm involves detecting and tracking a ridge in the FTLE field. In this type of algorithm, there must be some initial way to detect a point on the ridge, after which the ridge may be "grown" in either direction. This method has the advantage of only computing points along a ridge, but there are several drawbacks as well. Image processing techniques have been applied to tracking one dimensional (1D) ridges in a 2D field (see Tran and Lux[17] for example); however this does not readily generalize to higher dimensions.

In this paper, we focus on this second class of algorithm, ridge tracking. We present a ridge tracking algorithm for computing and extracting the LCS of a system. We use the fact that LCS are coherent in time as well as space to speed computations. In the case of a time dependent flow field, the LCS at one time step may be used to provide an excellent estimate of the location of the LCS at the next time step. In fact, Shadden *et al.*[15] showed that LCS are "nearly" material lines and that the fluid flux through LCS is usually on the order of numerical error. This means that, to a good approximation, LCS are simply advected with the flow. In the algorithm presented here, we take advantage of this property by using the LCS at time t to predict the location of LCS at time $t+\delta t$. We then make a small correction to ensure that the new points at time $t+\delta t$ are actually on the LCS and then extract the rest of the LCS via the ridge tracking algorithm. We also derive an estimate of the distance between a LCS and particles which begin on the LCS after some time δt has elapsed.

This manuscript is organized as follows. We first present and prove an estimate of the difference between the motion of LCS and that of a passive fluid particle. We then present a ridge tracking algorithm to compute LCS while taking advantage of spatial and temporal coherence of the structures. Finally, we present two examples using this algorithm, an analytically defined double gyre flow and the numerically computed flow created by a swimming jellyfish.

II. ESTIMATING THE DIFFERENCE BETWEEN LCS MOTION AND LAGRANGIAN PARTICLE MOTION

Shadden *et al.*[15] showed in their seminal paper that LCS are nearly material lines by providing an estimate of the flux through LCS. Here, we provide an estimate of the amount by which the time dependent motion of LCS differs from the motion of Lagrangian particles in the fluid. The relevant result from Shadden *et al.*[15] is

$$\left.\frac{dL(\mathbf{x},t)}{dt}\right|_{L=0}=\frac{\langle\hat{\mathbf{t}},\nabla\sigma\rangle}{\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle}\left\langle\hat{\mathbf{t}},\frac{\partial\hat{\mathbf{n}}}{\partial t}-J\hat{\mathbf{n}}\right\rangle+\mathrm{O}(1/|T|),\tag{4}$$

where L is the signed distance from the nearest LCS, σ is the FTLE field, H is the Hessian of the FTLE field, J is the Jacobian of the velocity field, and $\hat{\mathbf{n}}$ and $\hat{\mathbf{t}}$ are the unit normal and tangent vectors to the LCS.

Given this result, it is trivial to estimate the distance between the LCS and a fluid particle initially located on the LCS via a Taylor expansion

$$L(\mathbf{x}(t+\delta t),t+\delta t)=L(\mathbf{x}(t),t)+\left.\frac{dL(\mathbf{x},t)}{dt}\right|_{L=0}\delta t+\mathrm{O}(\delta t^2).\tag{5}$$

Since $L(\mathbf{x}(t),t)$ is chosen to be zero at time t, we have the following theorem:

Theorem II.1:

$$L(\mathbf{x},t)=\frac{\langle\hat{\mathbf{t}},\nabla\sigma\rangle}{\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle}\left\langle\hat{\mathbf{t}},\frac{\partial\hat{\mathbf{n}}}{\partial t}-J\hat{\mathbf{n}}\right\rangle\delta t+O(\delta t/|T|)+O(\delta t^2)$$

The algorithm we present in this paper relies heavily on this distance being small. Since this is such a key result, we now provide a sketch of an alternative proof of Theorem II.1. Our proof is formulated in a way which makes it clearer that we are estimating the difference between the motion of LCS and material particles. We first estimate the motion of a point on a LCS, then use a Taylor series approximation to the

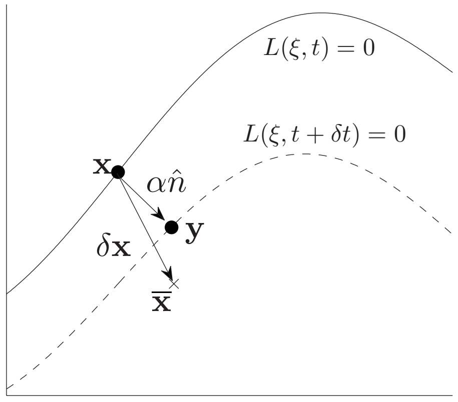


FIG. 1. $\mathbf{x}$, $\bar{\mathbf{x}}$, $\delta\mathbf{x}$, $\mathbf{y}$, and α are defined as shown here.

motion of a material point and calculate the difference between the two. This results in a slightly more direct proof than by going through the flux estimate provided in Shadden *et al.*[15]

A. Definitions

Our proof will use the following definitions as diagramed in Fig. 1. Let $\mathbf{x}$ be a point on the LCS at time t and let δt be a small increment in time. Then, let $\bar{\mathbf{x}}$ be the point to which $\mathbf{x}$ is advected [i.e., $\bar{\mathbf{x}}=\mathbf{x}+\int_t^{t+\delta t}\mathbf{v}(\mathbf{x}(\tau),\tau)d\tau$]. Let $\mathbf{y}$ be the intersection of the line through $\mathbf{x}$ normal to the LCS at time t and the LCS at time $t+\delta t$. Finally, α is the distance the LCS moves in the normal direction.

B. Alternative proof of the error estimate

We start by expanding $\nabla\sigma|_{y,t+\delta t}$ and $\nabla L|_{y,t+\delta t}$ about $\delta t=0$ and taking the inner product, $\langle\nabla L,\nabla\sigma\rangle|_{y,t+\delta t}$, as is done in[15]

$$\nabla\sigma|_{y,t+\delta t}=\nabla\sigma+\alpha\mathrm{H}\hat{\mathbf{n}}+\frac{\partial\nabla\sigma}{\partial t}\delta t+O(\delta t^2),$$

$$\nabla L|_{y,t+\delta t}=\nabla L+\frac{\partial\nabla L}{\partial t}\delta t+O(\delta t^2).$$

Taking the inner product gives

$$\langle\nabla L,\nabla\sigma\rangle|_{y,t+\delta t}=\langle\nabla L,\nabla\sigma\rangle+\langle\nabla L,\alpha\mathrm{H}\hat{\mathbf{n}}\rangle+\left(\left\langle\nabla L,\frac{\partial\nabla\sigma}{\partial t}\right\rangle+\left\langle\frac{\partial\nabla L}{\partial t},\nabla\sigma\right\rangle+\left\langle\frac{\partial\nabla L}{\partial t},\alpha\mathrm{H}\hat{\mathbf{n}}\right\rangle\right)+O(\delta t^2), \tag{6}$$

but since $\nabla L=\hat{\mathbf{n}}$ and $\nabla\sigma\|\hat{\mathbf{t}}$ on the LCS we know that $\langle\nabla L,\nabla\sigma\rangle|_{y,t+\delta t}=0$ and $\langle\nabla L,\nabla\sigma\rangle=0$.

Also, Corollary 4.1 of Shadden *et al.*[15] states that on the LCS, and for an arbitrary vector $\mathbf{u}$,

$$\langle\hat{\mathbf{n}},\mathrm{H}\mathbf{u}\rangle=\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle\langle\hat{\mathbf{n}},\mathbf{u}\rangle, \tag{7}$$

so

$$\left\langle\frac{\partial\nabla L}{\partial t},\alpha\mathrm{H}\hat{\mathbf{n}}\right\rangle=\alpha\left\langle\mathrm{H}\frac{\partial\hat{\mathbf{n}}}{\partial t},\hat{\mathbf{n}}\right\rangle=\alpha\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle\left\langle\hat{\mathbf{n}},\frac{\partial\hat{\mathbf{n}}}{\partial t}\right\rangle=0.$$

So Eq. (6) may be rearranged as

$$\alpha\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle=-\left(\left\langle\nabla\sigma,\frac{\partial\hat{\mathbf{n}}}{\partial t}\right\rangle+\left\langle\hat{\mathbf{n}},\frac{\partial\nabla\sigma}{\partial t}\right\rangle\right)\delta t+O(\delta t^2).$$

Finally, we use Corollary 3.1 of Shadden *et al.*,[15] which states

$$\frac{\partial\nabla\sigma}{\partial t}=-J\nabla\sigma-\mathrm{H}\mathbf{v}+O(1/|T|), \tag{8}$$

to arrive at

$$\alpha\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle=\left(\left\langle\nabla\sigma,J\hat{\mathbf{n}}-\frac{\partial\hat{\mathbf{n}}}{\partial t}\right\rangle+\langle\hat{\mathbf{n}},\mathrm{H}\mathbf{v}\rangle\right)\delta t+O(\delta/|T|)+O(\delta t^2). \tag{9}$$

Equation (9) gives an estimate for the motion of the LCS based only on properties of the system. However, this is not useful in practice due to the difficulty in computing these quantities. Instead, we use the motion of a Lagrangian fluid particle which begins on the LCS to approximate the motion. Taking the difference between the motion of the Lagrangian particle and the LCS gives us an error estimate.

A Taylor approximation gives the motion of the particle as

$$\delta\mathbf{x}=\mathbf{v}\delta t+O(\delta t^2).$$

We then apply the Hessian to this vector and take the inner product with the unit normal to the LCS,

$$\langle\hat{\mathbf{n}},\mathrm{H}\delta\mathbf{x}\rangle=\langle\hat{\mathbf{n}},\mathrm{H}\mathbf{v}\rangle\delta t+O(\delta t^2),$$

which, using Eq. (7), implies

$$\langle\delta\mathbf{x},\hat{\mathbf{n}}\rangle\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle=\langle\hat{\mathbf{n}},\mathrm{H}\mathbf{v}\rangle\delta t+O(\delta t^2). \tag{10}$$

Finally, subtracting Eq. (9) from Eq. (10) gives, after some algebra,

$$\alpha-\langle\delta\mathbf{x},\hat{\mathbf{n}}\rangle=\|\nabla\sigma\|\frac{\left\langle\hat{t},\frac{\partial\hat{\mathbf{n}}}{\partial t}-J\hat{\mathbf{n}}\right\rangle}{\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle}\delta t+O(\delta t/|T|)+O(\delta t^2), \tag{11}$$

which is the same result as Thm. II.1. It is worth noting that here, we have actually computed the difference between the motion of the LCS and the normal projection of the particle motion. However, to first order, the particle motion tangent to the LCS does not affect the distance of the particle from the LCS so, to first order, $L(\mathbf{x},t+\delta t)=\alpha-\langle\delta\mathbf{x},\hat{\mathbf{n}}\rangle$.

The most important characteristic of this estimate is that it is typically very small. For more details, see Shadden *et al.*,[15] but this term is typically of the same order as numerical errors. The term $\|\nabla\sigma\|$ is asymptotically zero for time independent systems and represents the rate at which the ridge is rising or falling. It is typically small along most of the ridge. The term $\langle\hat{\mathbf{n}},\mathrm{H}\hat{\mathbf{n}}\rangle$ appears in the denominator and is locally maximized in norm on the LCS (by the definition of LCS). Finally, the term $\langle\hat{t},\partial\hat{\mathbf{n}}/\partial t-J\hat{\mathbf{n}}\rangle$ represents the difference be-

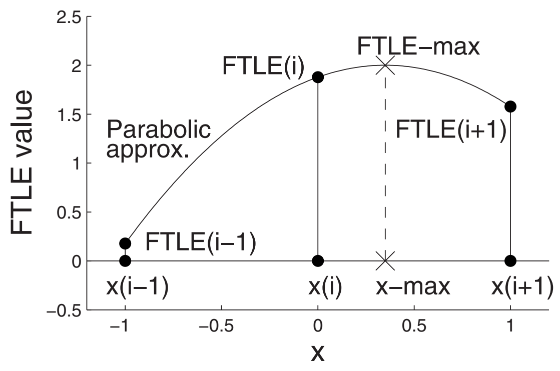


FIG. 2. The actual local maximum of the FTLE is estimated with a parabolic approximation of the ridge.

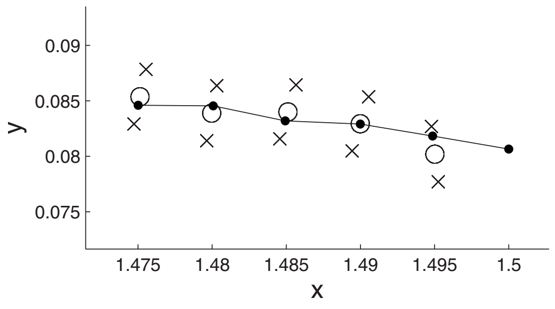


FIG. 3. The first several steps along a ridge beginning at the right end. The estimated next point based on the ridge trajectory is shown as a circle. The other points on the normal line are shown as ×'s and the actual trajectory is drawn as a line.

tween the rotation of the LCS and the rotation of vectors normal to the LCS by the Eulerian velocity field.

III. A RIDGE TRACKING ALGORITHM

In this investigation we will focus on the extraction of LCS via a ridge tracking algorithm. This algorithm has the advantage of computing the FTLE field at a minimum number of points, so it provides a very large speedup over methods which compute the FTLE field over the entire domain. Additionally, since fewer particles are used, less memory is required for the computations, and since the output consists only of the LCS lines rather than the entire FTLE field, the output files are much smaller as well.

A. The first time step

The first time step requires that we somehow find at least one point on the LCS to be extracted, as well as the orientation of the corresponding ridge in the FTLE field. We accomplish this by computing the FTLE value along lines which crisscross the domain. These lines of FTLE values then have local maxima where they cross a ridge in the FTLE field so we initialize the ridges at these points. To obtain a more accurate estimate, we use the locally maximum value, plus the value on either side to approximate the FTLE values around the ridge by a parabola and locate the maximum of this parabola (see Fig. 2). We also require that these points have a FTLE value above some threshold (experience shows that an appropriate threshold is usually in the range of 50%–80% of the maximum FTLE value) since we generally wish to extract only the "strongest" LCS. As a side note, for incompressible flows, the FTLE value must be non-negative. This has caused some confusion in the past since negative FTLE values have been reported for incompressible flow (for example, see Fig. 6 of Shadden *et al.*[15]). For a proof of this property, see the Appendix.

The next step is to find the orientation of the ridges in order to begin stepping along the ridge to extract the LCS. For each initial point, this is accomplished by computing the FTLE value of the eight points surrounding the point at a distance of one step size. We then find the adjacent point with the maximum FTLE value and again use a parabolic approximation to provide a better estimate of the location of the highest surrounding point. Finally, we note the orientation of the ridge as the vector from the initial point to this highest surrounding point. We then begin stepping along the ridge in both directions from this initial point.

B. Tracking the ridge

Given a point on the ridge and the orientation of the ridge, we step forward by taking a step in the direction of the given orientation, computing the FTLE value at three points on a line normal to the step direction, approximating the FTLE value by a parabola, and estimating the location and value of the true local max along this normal line (see Fig. 2). The first several steps along a ridge are shown in Fig. 3. To help prevent the trajectory from jumping from one side of the true ridge to the other, after the second step we update the trajectory based on the average of the last two steps. Experience also dictates that the spacing of the points on the normal line (used to compute the parabolic approximation) should be about 1/2 of the step size along the ridge. Finally, when making the parabolic approximation, one must ensure that the parabola has the appropriate concavity (concave down) and that the maximum occurs between the points on the normal line. If either of these tests fails, it is best to step to the point on the normal line with the maximum FTLE value. If these tests fail because the ridge has ended or the algorithm has lost the ridge, then the stopping criteria (presented below) will prevent the ridge from growing any further.

C. Stopping criteria

It is important to know when to stop tracking a ridge because either the ridge has ended or the algorithm has lost the ridge. We have selected three criteria which will cause the algorithm to stop tracking a ridge. The criteria and their results are as follows:

(1) The end of the ridge leaves the domain of the computations: stop tracking the ridge.
(2) The ridge hits the start or end of another ridge: join the two ridges into one.
(3) The FTLE value on the ridge falls below the threshold value [we use 0.8 max(FTLE)]: either the ridge is ending or the algorithm has lost the ridge so stop tracking the ridge.

D. Later time steps

Once all the ridges have ended by meeting one of the stopping criteria, we move on to compute the LCS at the next time step. To avoid the cost of initially detecting points on the ridges we simply advect some points on the ridge forward to the next time. For efficiency, we only advect every 15th and 16th point along the ridge, but this is a tunable parameter which may be used to help optimize the algorithm. We advect pairs of points so that we can easily determine the orientation of the new ridge and eliminate some error by taking their average position as the estimated new initial position on the ridge. We also compute two additional points on the line normal to the new ridge and use the parabolic approximation to more precisely locate the new ridge. Once these new initial points and orientations have been computed, we throw out any points below the threshold value and proceed exactly as for the first time step.

Even if a few of the advected points miss the new LCS ridges the other points on the ridge will grow the ridge to fill in the gap. In practice, this rarely happens and all the advected points lie very near the new ridge. Also, it is possible (and likely) that entirely new LCS will be created elsewhere in the flow domain. This algorithm will not detect these new ridges so it is necessary to occasionally repeat the initial detection part of the algorithm as performed in the first step. The frequency of repeating this step is entirely dependent on whether or not it is deemed acceptable to miss a newly created LCS for a few time steps. Also, if it is critical that all ridges are detected and a new ridge appears during this reinitialization, the previous time steps may be recomputed by the same method (only advecting the points backward in time) until the ridge is no longer present.

IV. RESULTS

To demonstrate the capabilities of this algorithm we present the results from two examples accompanied by the FTLE field over the entire domain. The first is an example which has become a standard test case for computing LCS, the time dependent double gyre. Second, we present the result of computing the LCS created by swimming jellyfish. This example was analyzed by our group and is further discussed in another paper.[9] In both examples, the ridge tracking algorithm represents a significant increase in performance.

A. The time dependent double gyre

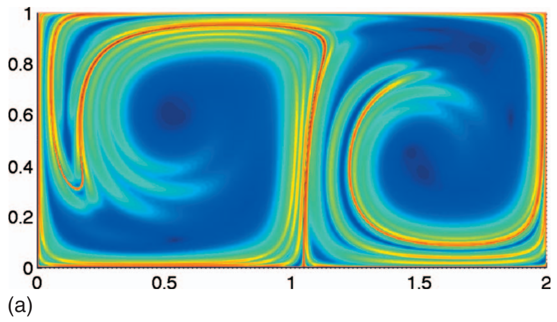
The time dependent double gyre is an oscillating perturbation to two counter-rotating gyres. The velocity field for this system is

$$u = -\pi A \sin(\pi f(x))\cos(\pi y), \tag{12}$$

$$v = \pi A \cos(\pi f(x))\sin(\pi y)\frac{\partial f}{\partial x}, \tag{13}$$

where

$$f(x,t) = a(t)x^2 + b(t)x, \tag{14}$$



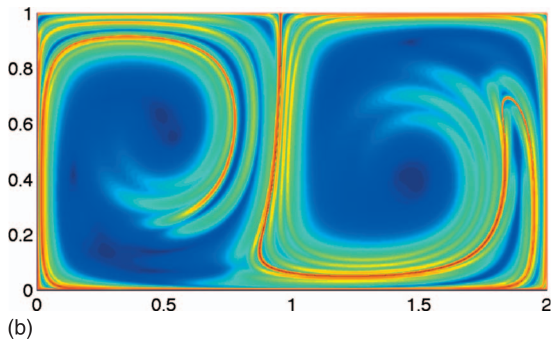


FIG. 4. (Color) The (a) forward and (b) backward FTLE fields for the time dependent double gyre system at time $t=0$ with $A=0.1$, $\epsilon=0.1$, $\omega=2\pi/10$, and $T=-15$.

$$a(t) = \epsilon \sin(\omega t), \tag{15}$$

$$b(t) = 1 - 2\epsilon \sin(\omega t). \tag{16}$$

For this investigation we choose the parameters

$$A = 0.1, \quad \epsilon = 0.1, \quad \omega = \frac{2\pi}{10}$$

and integration time $T = \pm 15$ which gives a system with an oscillation period of 10.

For reference we have computed the full FTLE field and the forward and backward FTLE fields are shown in Figs. 4(a) and 4(b), respectively. This field was computed with grid spacing of 0.0025, resulting in an 801×401 grid and the computation took 137.74 s per time step to run. FTLE values were computed using central differencing on this grid resulting in a finite difference spacing of $\Delta x=0.005$. This grid spacing was chosen to be 1/2 of the step size used in the ridge tracking. Since the ridge tracking algorithm can only correct steps along the ridge by up to 1/2 step size, we feel this grid spacing provides a fair comparison between the two methods.

The LCS extracted from this same system with the ridge tracking algorithm is shown in Fig. 5. The results are shown only for time $t=0$, but are typical of all times and are indistinguishable from the ridges seen in Figs. 4(a) and 4(b). For this example, we have chosen to use a step size along the ridges of 0.005, a spacing of points used to compute parabolic approximations of 0.0025, and FTLE values were computed with central differencing on a four point stencil with spacing $\Delta x=0.005$ (the same as the spacing used above to compute the full FTLE field). Finally, we used a threshold value of 80% of the maximum computed FTLE value.

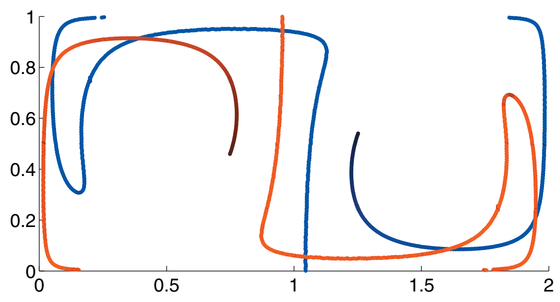


FIG. 5. (Color) The extracted LCS with the ridge tracking algorithm for the time dependent double gyre system at time $t=0$ with $A=0.1$, $\epsilon=0.1$, $\omega=2\pi/10$, and $T=\pm 15$. Forward LCS are shown in blue and backward LCS are shown in red. Also, the relative height of the ridge is indicated by the shade of the color. As the ridge height decreases to the threshold value, the color fades to black.

As mentioned previously, it is necessary to reinitialize the ridge tracking algorithm occasionally to ensure that any newly formed LCS is captured. For this system, we chose to compute the LCS at time steps of $\Delta t=0.02$ and to reinitialize ten times per period (at $t=0,1,2,3,\ldots$). Since the length of the LCS ridges present varies with time, so does the computation time. To account for this, we have taken the ten windows of 50 time steps each (which makes one period of the system) and averaged the time taken at each time step. The first time step takes 9.76 s while the 50 time step windows take an average of 4.28 s per time step. This represents a speedup of 2.3 times speedup from the first time step to subsequent step (due to the prediction step). The overall average time step for this run represents a 32.2 times speedup over the full FTLE calculation.

Finally, if we include the time necessary to write the data to a file, we see additional savings. The ridge tracking algorithm only needs to write the ridgelines to a file, while the other algorithm must write the FTLE values over the entire domain. This results in a total speedup of 35.0 times.

B. Jellyfish swimming

In our second example, we examine the flow created by a swimming jellyfish. The jellyfish is a specimen of *Sarsia tubulosa*. The flow around the swimming jellyfish is in the form of data files output from a CFD code. The deformation of the jellyfish body was obtained from videos of an actual swimming jellyfish and used as input to an arbitrary Lagrangian–Eulerian (ALE) CFD code. This code then computes the flow field around the swimming jellyfish on a moving, unstructured quadrilateral mesh. For additional details on the CFD process and the ALE method used, see Sahin and Mohseni.[13]

The CDF simulations were performed in axisymmetric, swirl free cylindrical coordinates and the FTLE computations were also performed in this coordinate system. This is only a minor change and corresponds to changing $d\Phi/d\mathbf{x}$ to

$$\frac{d\Phi}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial r_f}{\partial r_i} & 0 & \frac{\partial r_f}{\partial x_i} \\ 0 & \frac{r_f}{r_i} & 0 \\ \frac{\partial x_f}{\partial r_i} & 0 & \frac{\partial x_f}{\partial x_i} \end{pmatrix}, \tag{17}$$

where r_i, r_f, x_i, and x_f are the initial and final radial and axial coordinates of a particle in the flow. This particular jellyfish uses a jetting type of propulsion and ejects a very strong vortex during swimming.[9] We have chosen this example for two reasons, to test our algorithm on a discrete velocity field which is stored in data files and to examine the performance on a more realistic flow field with more complicated structures than the double gyre.

In cases where the velocity field is stored in data files instead of analytically defined, several additional considerations must be made to implement this ridge tracking algorithm. First and foremost, depending on the size and format of the velocity files, velocity readins may actually be a limiting factor for algorithm performance. It is very important to choose an efficient data format to minimize velocity readin time. In this case, we have chosen to use the NETCDF data format.[18] In our experience, this format has proven to be efficient and flexible.

Second, a ridge tracking algorithm requires computing FTLE values for each step along a ridge. If this is done by reading in the velocity for each time step as it is needed during the advection process, then the velocity files may each need to be read in hundreds of times. To avoid this, the ridge tracking algorithm should only be applied to situations where the velocity field for one complete period of integration can fit in memory.

Finally, since the mesh is unstructured, it is very important to use an efficient search algorithm to locate elements before interpolating the velocity during particle advections. In the case where the mesh is also moving, it is necessary to us a search algorithm which does not have a large setup cost since each time step will require a new setup. The most popular geometric search algorithms include nearest neighbor (NN), digital trees, and structured auxiliary mesh (SAM). Khoshniat *et al.* provide an excellent comparison of these three methods.[7] NN search scales as $O(n)$, tree searches scale as $O(\log_2(n))$ and SAM searches require large setup time. Due to low setup cost and good speed, we have chosen to use an Alternating Digital Tree.[1] This method creates a tree containing the elements to be searched and each level in the tree represents a partition in the x or y direction. If an element lies on the boundary of two partitions, we choose to add it to each. The setup cost and initial search combine to take only a few seconds for the full FTLE field calculations in this example. Once the particles have been located initially, search during the advection process is performed by checking whether or not a particle is in the same element as the previous time step. If it is not, the elements are traversed sequentially, based on the element connections and their location relative to the particle.

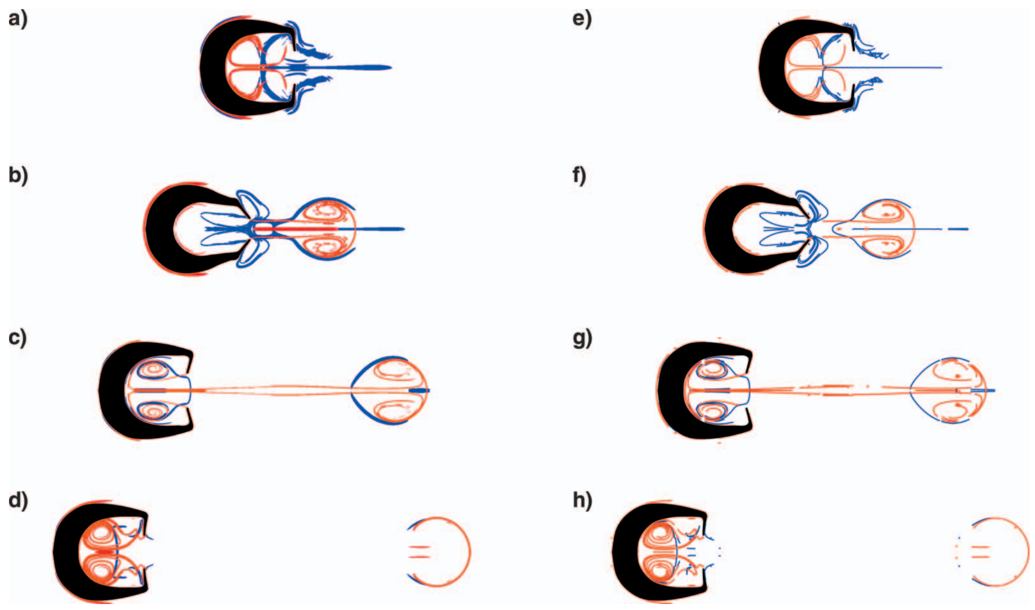


FIG. 6. (Color) The LCS for several time steps as computed with both the standard algorithm as well as the ridge tracking algorithm. Forward LCS are shown in blue and backward are red. The results of the standard algorithm are shown in (a)–(d) and the ridge tracking algorithm is show in (e)–(h). The four different time steps are evenly spaced at 0.25 s intervals and make up one complete swimming cycle (1 s).

For comparison, we have again computed the full FTLE field using the standard algorithm. The thresholded contour plots of the forward and backward FTLE fields for this jellyfish are presented in Figs. 6(a)–6(d). This computation was performed at mesh spacing of 0.0025 with an integration time of 0.40 s (40 time steps). On average, computing and writing the FTLE field for each time step took 874.6 s or about 14.6 min.

The resulting LCS from the ridge tracking algorithm are also shown in Figs. 6(e)–6(h). The ridge tracking algorithm took an average of 60.8 s per time step to run and captured all the major features of the LCS seen in the full FTLE field. This represents a speedup of 14.4 times, as the time to compute each time step was cut from 14.6 min to just over 1 min. While this is not as drastic a speedup as we saw for the double gyre example, it is still a substantial savings.

In both sets of figures, we can clearly see all the major LCS present at this time in the flow. The jellyfish begins contracting its bell to produce a vortex and propel itself forward [Figs. 6(a) and 6(e)]. In Figs. 6(b) and 6(f), the ejected vortex is moving away from the jellyfish and beginning to pinch off. We can also see lobes formed by the forward (blue) LCS around the velar opening. These lobes contain the fluid which will be drawn into the subumbrellar cavity as the jellyfish relaxes the bell.

Next, in Figs. 6(c) and 6(g), the forward LCS lobes near the bell opening have been drawn into the bell as the bell relaxes. The fluid contained in these lobes is exactly the fluid which is entrained into the bell. The vortex continues to move away from the jellyfish and vortex pinchoff is complete (note the forward LCS which closes the back of the vortex). Also, there is another vortex present in the bell of the vortex in both Figs. 6(c) and 6(g) and Figs. 6(d) and 6(h) as the expelled vortex continues to move away before the next contraction.

It is important to note that the major structures detected by the standard algorithm are also present in the ridge tracking algorithm's results. In fact, because the results of the standard LCS computation are often visualized with a thresholded contour plot, they often do not appear as clear lines where the ridges are present, but rather as narrow regions of high FTLE value. The ridge tracking algorithm suffers from no such drawback. Although there is a slight loss of detail in the vortex core seen in Figs. 6(f) and 6(g), we are able to extract a single, 1D line along each ridge. This is well demonstrated in Fig. 6. For example, the results of the ridge tracking algorithm display the structures in the bell in Figs. 6(h) more cleanly than the standard algorithm (Fig. 6(d)).

The structures discussed here exactly match the structures we have previously seen.[9] We are even able to observe the complicated structures within the bell which have not been previously observed. The characteristic swimming mechanism is also well shown (see Fig. 7). The jellyfish emits a single, highly energetic vortex ring during each contraction. This vortex ring separates from the jellyfish very rapidly at a Strouhal number (fL/v) of 0.1. This means that the vortices produced separate from the jellyfish at a rate of about 10 radii per swimming cycle. This gives very little opportunity for feeding during swimming since the fluid moves past the jellyfish so quickly. In contrast, various other jellyfish use a paddling type of swimming which actually complements feeding. For example, we have previously found the jellyfish *Aequorea victoria* to swim at a Strouhal number of 1.1. For a complete discussion of the swimming of this jellyfish, see Lipinski and Mohseni.[9]

This algorithm is very well suited for picking out the most major LCS. As the integration time is increased, the LCS that are revealed become increasingly complex and close together. The total length of LCS present in the domain

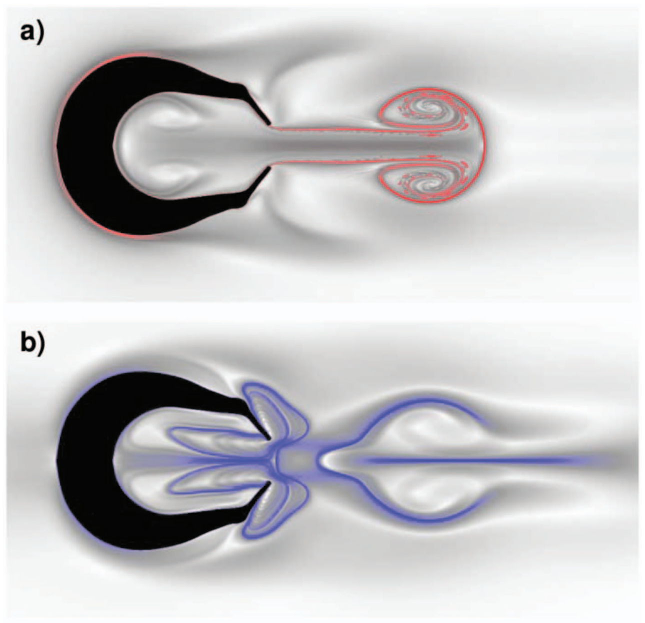


FIG. 7. (Color) Backward (a) and forward (b) LCS for *Sarsia tubulosa* as computed with the standard LCS algorithm.

increases as well. In many applications such as flow control and for identifying major flow structures, only the most major LCS needs to be computed. For these purposes, this algorithm provides a huge time savings over the standard algorithm.

V. CONCLUSIONS

Finding LCS is a computationally intensive process. To date, they have been used in many applications despite this cost due to their many useful properties for visualizing and understanding fluid flows. However, as the problems to which LCS are applied grow in size, the computational cost becomes prohibitive. To help solve this issue, fast algorithms are needed which can take advantage of the nature of LCS as ridges in the FTLE field and avoid computing the FTLE field at unnecessary points far from any ridges. Several AMR strategies have been proposed in the past. In this paper we fully outline a new ridge tracking algorithm which also takes advantage of the temporal coherence of LCS to further speed calculations after the first time step.

For an analytically defined velocity field, our algorithm gives a speedup of 35.0 times over the standard LCS algorithm. Due to the additional complexities of dealing with a discrete velocity field and the more complex LCS present in the example of the swimming jellyfish, we only see a 14.4 times speedup in this case, but we believe that future refinements of the algorithm may increase this speedup. Additionally, all the major LCS were successfully captured in each example. The speedup is achieved without a loss of detail or accuracy in finding the most significant LCS in a system. This ridge tracking algorithm also provides additional savings in terms of memory usage and size of the output files.

We believe that using the locations of LCS at a given time to predict the next time step will play a major role in future algorithms as well. This is a very general idea which could also be used as part of an AMR routine or other algorithms. For example, an AMR algorithm could use this information to immediately create a fine mesh only where the LCS are predicted to be located and skip all (or most) intermediate refinement steps in these areas.

In the future, we plan to implement AMR algorithms which also use the temporal coherence of LCS to speed computations. We would also like to continue to develop and refine the ridge tracking algorithm in two dimensions to improve the stability of the algorithm in areas of very complicated flow. Finally, computing LCS in three dimensions is an even more computationally demanding task. We would like to develop a similar algorithm which is able to track 2D LCS surfaces in a three dimensional domain. While it is more complicated to grow outward along a surface than a line, such an algorithm could provide tremendous computational savings.

ACKNOWLEDGMENTS

Financial support by the Office of Naval Research under the contract 09PR06473-00/1053746 is gratefully acknowledged.

APPENDIX: PROOF THAT $\sigma \geq 0$ IF $\nabla \cdot \mathbf{u} = 0$

For incompressible flows, the contraction of a fluid in one direction must be balanced by expansion in another direction. In terms of LCS, this translates to the following theorem:

Theorem A.1: *For incompressible flows, the FTLE field is non-negative,*

$$\nabla \cdot \mathbf{u} = 0 \Rightarrow \sigma_t^T(\mathbf{x}) \geq 0 \quad \forall\, \mathbf{x}, t, T.$$

This may be used as a simple check of the results obtained from FTLE calculations on an incompressible flow. Additionally, for compressible flows, if the FTLE values in some region are negative, this implies that the fluid is locally contracting in every direction over the chosen integration time.

Proof of Theorem A.1: The gradient of the flow field, commonly denoted as $d\Phi/d\mathbf{x}$, is the same as the Jacobi matrix $\mathbf{J}$, which appears in many Lagrangian fluid dynamics texts. It is a well known result that if the flow is incompressible the determinant of the Jacobi matrix is 1: $\det(J)=1$.

Also, for general matrices A and B,

$$\det(AB) = \det(A)\det(B),$$

$$\det(A^T) = \det(A),$$

$$\det(A) = \prod_{i=1}^{d} \lambda_i.$$

Then, using the definition of the FTLE field, we have

$$\Delta = J^* J,$$

$$\det(\Delta) = \det(J^*)\det(J) = 1 = \prod_{i=1}^{d} \lambda_i(\Delta).$$

Δ is positive definite since

$$\langle \mathbf{x}, \Delta \mathbf{x} \rangle = \langle \mathbf{x}, J^* J \mathbf{x} \rangle = \langle J\mathbf{x}, J\mathbf{x} \rangle = \|J\mathbf{x}\|^2 \geq 0$$

and

$$\det(\Delta) = 1.$$

So $\lambda_i(\Delta) > 0$ and

$$\lambda_{\max}(\Delta) = \max_i(\lambda_i(\Delta)) \geq 1$$

and

$$\sigma = \frac{1}{|T|} \ln \sqrt{\lambda_{\max}(\Delta)} \geq \frac{1}{|T|} \ln \sqrt{1} \geq 0.$$

[1]Bonet, J. and Peraire, J., "An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems," Int. J. Numer. Methods Eng. **31**, 1–17 (1991).
[2]Cardwell, B. and Mohseni, K., "Vortex shedding over a two-dimensional airfoil: Where the particles come from," AIAA J. **46**, 545–547 (2008).
[3]Garth, C., Gerhardt, F., Tricoche, X., and Hagen, H., "Efficient computation and visualization of coherent structures in fluid flow applications," IEEE Trans. Vis. Comput. Graph. **13**, 1464–1471 (2007).
[4]Green, M. A., Rowley, C. W., and Haller, G., "Detection of Lagrangian coherent structures in 3d turbulence," J. Fluid Mech. **572**, 111–120 (2007).
[5]Haller, G., "Lagrangian coherent structures from approximate velocity data," Phys. Fluids **14**, 1851–1861 (2002).
[6]Haller, G. and Yuan, G., "Lagrangian coherent structures and mixing in two-dimensional turbulence," Physica D **147**, 352–370 (2000).
[7]Khoshniat, M., Stuhne, G. R. , and Steinman, D. A., "Relative performance of geometric search algorithms for interpolating unstructured mesh data," Med. Image Comput. Comp. Assist. Interv. – MICCAI, Part 2 **2879**, 391–398 (2003).
[8]Lipinski, D., Cardwell, B., and Mohseni, K., "A Lagrangian analysis of a two-dimensional airfoil with vortex shedding," J. Phys. A **41**, 344011 (2008).
[9]Lipinski, D. and Mohseni, K., "Flow structures and fluid transport for the hydromedusae Sarsia tubulosa and Aequorea victoria," J. Exp. Biol. **212**, 2436–2447 (2009).
[10]Mathur, M., Haller, G., Peacock, T., Ruppert-Felsot, J. E., and Swinney, H. L., "Uncovering the Lagrangian skeleton of turbulence," Phys. Rev. Lett. **98**, 144502 (2007).
[11]Peng, J. and Dabiri, J. O., "Transport of inertial particles by Lagrangian coherent structures: Application to predator-prey interactions in jellyfish feeding," J. Fluid Mech. **623**, 75–84 (2009).
[12]Sadlo, F. and Peikert, R., "Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction," IEEE Trans. Vis. Comput. Graph. **13**, 1456–1463 (2007).
[13]Sahin, M. and Mohseni, K., "An arbitrary Lagrangian-Eulerian formulation for the numerical simulation of flow patterns generated by the hydromedusa Aequorea victoria," J. Comput. Phys. **228**, 4588–4605 (2009).
[14]Shadden, S. C., Dabiri, J. O., and Marsden, J. E., "Lagrangian analysis of fluid transport in empirical vortex ring flows," Phys. Fluids **18**, 047105 (2006).
[15]Shadden, S. C., Lekien, F., and Marsden, J. E., "Definition and properties of Lagrangian coherent structures," Physica D **212**, 271–304 (2005).
[16]Shadden, S. C., Lekien, F., Paduan, J. D., Chavez, F., and Marsden, J. E., "The correlation between surface drifters and coherent structures based on hf radar in Monterey Bay," Deep-Sea Res., Part II **56**, 161–172 (2009).
[17]Tran, T. T. H. and Lux, A., "A method for ridge extraction," Asian Conference on Computer Vision, Jeju, Korea, 2004 (unpublished), p. 960.
[18]Unidata. Netcdf (network common data form). http://www.unidata.ucar.edu/software/netcdf/, September 2009.
[19]Voth, G. A., Haller, G., and Gollub, J. P., "Experimental measurements of stretching fields in fluid mixing," Phys. Rev. Lett. **88**, 254501 (2002).
[20]Weldon, M., Peacock, T., Jacobs, G. B., Helu, M., and Haller, G., "Experimental and numerical investigation of the kinematic theory of unsteady separation," J. Fluid Mech. **611**, 1–11 (2008).
[21]Wilson, M., Peng, J., Dabiri, J. O., and Eldredge, J. D., "Lagrangian coherent structures in low Reynolds number swimming," J. Phys.: Condens. Matter **21**, 204105 (2009).