Fast Computation of FTLE Fields for Unsteady Flows:
A Comparison of Methods

Steven L. Brunton & Clarence W. Rowley

This paper presents new, efficient methods for computing finite-time Lyapunov exponent (FTLE) fields in unsteady flows. The methods approximate the particle flow map to eliminate redundant particle integrations in neighboring flow map calculations. Two classes of flow map approximations are investigated based on composition of intermediate flow maps; unidirectional approximation constructs a time-$T$ map by composing a number of smaller time-$h$ maps, while bidirectional approximation constructs a flow map by composing both positive and negative-time maps. The unidirectional method is shown to be fast and accurate, although it is memory intensive. The bidirectional method is also fast and uses significantly less memory; however, it is prone to large error which is aligned with the opposite-time Lagrangian coherent structures. The algorithms are implemented and compared on three example fluid flows: the double gyre, a low Reynolds number pitching flat plate, and unsteady ABC flow.

Lagrangian coherent structures, or ridges of the field of finite-time Lyapunov exponents (FTLE), provide a useful analogue of invariant manifolds for unsteady flow fields. However, FTLE fields are expensive to compute due to the large number of particle trajectories which must be integrated to construct a particle flow map. Moreover, it is often necessary to compute a sequence of FTLE fields in time to visualize unsteady events. The methods presented here streamline the computation of a sequence of FTLE fields by removing redundant trajectory integrations between neighboring particle flow maps, which are necessary to compute the FTLE field. There are two categories of methods which approximate the particle flow map. The unidirectional method composes intermediate flow maps of the same time direction, and the bidirectional method composes intermediate flow maps of opposite time directions. It is shown that the unidirectional method is both fast and accurate, providing orders of magnitude computational savings over the standard method, when computing a sequence of FTLE fields in time to visualize the coherent structures of an unsteady flow.

I. INTRODUCTION

Coherent structures are important for understanding and modeling the underlying physical mechanisms of complex fluid flows [1]. In particular, Lagrangian coherent structures (LCS) are defined using particle trajectories and are Galilean-invariant, unlike Eulerian criteria [2]. LCS are defined as ridges of the finite-time Lyapunov exponent (FTLE) field and extend the notion of invariant manifolds from dynamical systems theory to unsteady flows. FTLE fields provide a measure of the stretching between nearby particles in a given flow and are important in determining transport mechanisms and separatrices in unsteady flows.

The theory and computation of finite-time Lyapunov exponents (FTLE), also known as direct Lyapunov exponents (DLE), is a relatively modern development [3, 4], with extensions to 3-dimensional [5, 6] and $n$-dimensional [7] flows. FTLE analysis has been widely applied in a number of branches of fluid mechanics, including fluid transport [8–10], bio-propulsion [11–13], flow over airfoils [14–16], plasmas [17], and geophysical flows [18, 19].

Because FTLE analysis is particularly useful for unsteady flows, it is often necessary to compute a sequence of FTLE fields in time to visualize an unsteady event. As flows become more complex, computations become increasingly expensive. In particular, FTLE calculations are expensive because a large number of particle trajectories must be integrated in order to obtain a particle flow map, often from stored velocity fields. When computing a sequence of FTLE fields in time, it is possible to speed up the computation considerably by eliminating redundant particle integrations. One approach that has been developed uses adaptive mesh refinement to reduce the number of integrations [20–23].

The approach here is to construct an approximate flow map by composing intermediate flow maps from FTLE field calculations at neighboring times. The first class of flow map approximation, denoted unidirectional composition, constructs a flow map by composing intermediate flow maps which are all aligned in the same time direction. The second class, denoted bidirectional composition, composes intermediate flow maps in both positive and negative-time. The methods are compared using analytic estimates for accumulated error and computation time as well as benchmarks on a number of example flows.

A. Main Results

In this paper we demonstrate that the unidirectional method is both fast and accurate, although it requires significantly more memory than the bidirectional method. Orders of magnitude speed-up may be achieved...
over the standard method, and computational improvement scales with the desired time resolution of the FTLE animation.

The bidirectional method suffers from significant error which is aligned with the opposite-time Lagrangian coherent structures. To understand this coherent error, we provide an error analysis for both methods, and uncover an important relationship between positive-time LCS (pLCS) and negative-time LCS (nLCS). In particular, in the neighborhood of a time-dependent saddle, particles near the pLCS flow into particles near the nLCS in positive time.

II. STANDARD COMPUTATION OF FTLE

Consider a time-dependent velocity field $\mathbf{u}$ on $\mathbb{R}^n$ and a particle trajectory $\mathbf{x}(t)$ which satisfies

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}(t), t).$$

The velocity field, $\mathbf{u}$, may be an unsteady solution of the Navier-Stokes equation, although it is only assumed that $\mathbf{u}$ is at least $C^0$ in time and $C^1$ in space. However, to extract Lagrangian coherent structures from the Hessian of the FTLE field, $\mathbf{u}$ must be $C^2$ in space [4]. The velocity field may be analytically defined, but is more often obtained from experiments or direct numerical simulation which produce velocity field data at discrete snapshots over a finite range of time. A method of computing finite-time Lyapunov exponents (FTLE) on a finite amount of discrete velocity field data has been developed [3, 4].

Computing an FTLE field typically involves four steps. First, a grid of particles $X_0 \subset \mathbb{R}^n$ is initialized over the domain of interest. The particles are advected (i.e., integrated) with the flow from initial time 0 to final time $T$, resulting in a time-$T$ particle flow map, $\Phi^T_0$, defined as:

$$\Phi^T_0 : \mathbb{R}^n \rightarrow \mathbb{R}^n; \quad \mathbf{x}(0) \rightarrow \mathbf{x}(0) + \int_0^T \mathbf{u}(\mathbf{x}(\tau), \tau) d\tau. \quad (2)$$

Next, the flow map Jacobian, $\mathbf{D}\Phi^T_0$ is computed, usually by finite-differencing, to obtain the Cauchy-Green deformation tensor,

$$\Delta = (\mathbf{D}\Phi^T_0)^* \mathbf{D}\Phi^T_0 \quad (3)$$

where $^*$ denotes transpose. Finally, the largest eigenvalue, $\lambda_{\text{max}}$, of this symmetric tensor is extracted and synthesized into an FTLE field:

$$\sigma(\Phi^T_0; \mathbf{x}) = \frac{1}{|T|} \log \sqrt[|\Phi^T_0; \mathbf{x}|]{\lambda_{\text{max}}(\Delta(\mathbf{x}))}. \quad (4)$$

The bottleneck in this procedure is the large number of particle integrations required to obtain the particle flow map, $\Phi^T_0$. Moreover, if the velocity field is time-varying, it is necessary to compute a sequence of FTLE fields in time to visualize unsteady events, as shown schematically in Fig. 1.

III. FLOW MAP APPROXIMATION

As seen in Fig. 1, the standard method of computing a sequence of FTLE fields involves inefficient re-integration of particles. The unidirectional and bidirectional methods outlined below streamline the computation of neighboring FTLE fields by approximating the time-$T$ flow map, $\Phi^T_{t_0}$, which can be written as:

$$\Phi^T_{t_0} = \Phi^T_{t_0} \circ \cdots \circ \Phi^T_{t_1} \circ \Phi^T_{t_0} \quad (5)$$

where $t_N = t_0 + T$.

Because the flow maps are obtained numerically on a discrete grid of points, $X_0 \subset \mathbb{R}^n$, it is necessary to interpolate the maps at points $\mathbf{x} \notin X_0$. Consider a flow map $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$, and the same flow map restricted to $X_0$, $\Phi|_{X_0} : X_0 \rightarrow \mathbb{R}^n$. The interpolation operator $I$ takes the discrete map $\Phi|_{X_0}$ and returns the interpolated map, $I\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$, which approximates $\Phi$ on $\mathbb{R}^n$:

$$I : \Phi|_{X_0} \mapsto I\Phi. \quad (6)$$

Here we use the shorthand $I\Phi \triangleq I(\Phi|_{X_0})$. We now obtain an approximation to the flow map in Eq. (5):

$$\tilde{\Phi}^T_{t_0} \circ \cdots \circ \tilde{\Phi}^T_{t_1} \circ \tilde{\Phi}^T_{t_0}(X_0) \approx \Phi^T_{t_0}(X_0) \quad (7)$$

The bidirectional method approximates the time-$T$ flow map $\Phi^T_{t_0}$ by first interpolating backward to a reference time, $t = 0$, then interpolating forward through a previously computed time-$T$ map, $\Phi^T_0$, and finally integrating forward to time $t_0 + T$. The unidirectional method approximates the time-$T$ flow map by composing a number of smaller time flow maps, $\Phi^T_{t_i}$, which all have the same time direction. Additionally, the chain rule may be applied to each of the methods, resulting in an approximation to the flow map Jacobian, $\mathbf{D}\Phi^T_{t_0}$.

A. Bidirectional Composition

Bidirectional approximation eliminates redundancy from neighboring FTLE field computations by using the

FIG. 1: The standard method for computing FTLE fields. Flow maps $\Phi_{t_0}^{h+T}$ for $k \in \{0, 1, 2, 3\}$ are shown (solid black arrow). Essential (blue) and redundant (red) particle integrations are outlined in dashed ovals.
information from a known flow map at a given time, \( \Phi_0^T \), to calculate an approximation to the flow map at future times, \( \Phi_{t_0}^{t_0+T} \). First, \( X_0 \) is integrated backward from \( t_0 \) to the reference time 0. The distorted grid \( \Phi_0^T(X_0) \) is then flowed forward through the interpolated map, \( \mathcal{I} \Phi_0^T \), and finally integrated forward an amount \( t_0 \) to the desired time \( t_0 + T \), as in Fig. 2:

\[
\Phi_{t_0}^{t_0+T} = \Phi_T^{t_0+T} \circ \mathcal{I} \Phi_0^T \circ \Phi_0^{t_0}, \tag{8}
\]

FIG. 2: Schematic for bidirectional method (a). Given a known flow map \( \Phi_0^T \) (solid black arrow), it is possible to approximate the flow map at future times \( \Phi_{kh}^{kh+T} \) (dashed black arrow) by integrating backward in time to \( t = 0 \) (red arrow), flowing forward through the interpolated map \( \mathcal{I} \Phi_0^T \) which was already computed (blue double arrow), and integrating trajectories forward to the correct final time (green arrow).

The flow \( \Phi_0^T \) is stored as a reference solution to compute an approximation to the flow map at later times

\[
\Phi_{kh}^{kh+T} \approx \Phi_{kh}^{kh+T} \text{ by } \Phi_{kh}^{kh+T} = \Phi_T^{kh+T} \circ \mathcal{I} \Phi_0^T \circ \Phi_0^{kh} \quad k \in \mathbb{Z} \tag{9}
\]

Instead of using \( \Phi_0^T \) as the reference solution for every future time, it is convenient to use the new approximate flow map \( \tilde{\Phi}_h^{kh+T} \) as the reference solution for the next iteration:

\[
\tilde{\Phi}_h^{2h+T} = \Phi_T^{2h+T} \circ \mathcal{I} \Phi_h^T \circ \Phi_0^{2h} \tag{10}
\]

This method may be continued, using \( \tilde{\Phi}_h^{kh+T} \) to approximate \( \Phi_{(k+1)h}^{(k+1)h+T} \):

\[
\tilde{\Phi}_{(k+1)h}^{(k+1)h+T} = \Phi_T^{(k+1)h+T} \circ \mathcal{I} \Phi_{(k+1)h}^T \circ \Phi_0^{(k+1)h} \tag{11}
\]

Errors will compound more quickly since approximate flow maps as used as the reference solutions for later approximations, as seen in Fig. 3.

B. Unidirectional Composition

The basis of the unidirectional method is to eliminate redundant particle integrations by only integrating particle trajectories through a given velocity field a single time. If a sequence of FTLE snapshots is desired at a time spacing of \( h \), for example as frames in an animation, then it is convenient to break up the time-\( T \) flow map into smaller time-\( h \) flow maps, where \( T = kh \):

\[
\tilde{\Phi}_0^{kh} = \mathcal{I} \Phi_{(k-1)h}^h \circ \cdots \circ \mathcal{I} \Phi_{2h}^h \circ \Phi_0^h \tag{12}
\]

This method is called unidirectional because particle flow maps of the same time direction are used, as opposed to the bidirectional method which composes both positive-time and negative-time flow maps.

The simplest approach is to compute a number of time-\( h \) flow maps and store them in memory. Then, to construct an approximate \( \Phi_{t_0}^{t_0+T} \), it remains only to compose the sequence of interpolated time-\( h \) flow maps. The next iteration involves integrating one more time-\( h \) flow map and composing the next sequence, as in Fig. 4.

To further improve efficiency by reducing the total number of flow map compositions, it is possible to construct a multi-tiered hierarchy of flow maps for reuse in neighboring flow map constructions. With enough memory, it is possible to reduce the number of interpolated compositions by increasing the number of tiers of flow maps, each tier being constructed as the composition of two of the flow maps in the next tier lower, as in Fig. 5.
Applied to the bidirectional methods, this yields:

\[ \Phi^{T}_{0} = \Phi^{T}_{T-h} \circ \cdots \circ \Phi^{2h}_{h} \circ \Phi^{h}_{0} \]

and applied to the unidirectional methods, this yields:

\[ \Phi^{T}_{0} = \Phi^{T}_{T-h} \circ \cdots \circ \Phi^{2h}_{h} \circ \Phi^{h}_{0} \]

\[ \implies D\Phi^{T}_{0}(x) = D\Phi^{T}_{T-h} \left( \Phi^{2h}_{0}(x) \right) \times \cdots \times D\Phi^{h}_{0}(x). \]

### IV. COMPARISON OF METHODS

Each method from Section III is implemented and tested on three example problems: the periodic double gyre, 2D flow over a pitching flat plate at Reynolds number 100, and 3D unsteady ABC flow. These examples are chosen because they cover a range of features including 2D and 3D vector fields, which are either defined analytically or obtained from data files from DNS on either open, closed, or periodic domains. Each example problem is discussed more in Appendix B, including details such as how the velocity field is defined, and on what domain. In the pitching plate example, velocity field snapshots are all loaded up-front before applying the methods.

Table I summarizes the results comparing each method on the three example fluid flows. In each comparison, the standard, unidirectional and bidirectional methods are used to compute a sequence of FTLE fields which are frames in an unsteady animation. The flow map duration used to compute an FTLE field is \( T \), and the time-spacing between neighboring FTLE fields is \( h \), so the number of animation frames per flow map duration is \( T/h \). As demonstrated in Section IV B, this is an upper bound on the speed-up of the unidirectional method.

In each comparison, the unidirectional method is accurate and offers the greatest speed-up over the standard method. However, it also requires more memory than any other method. The bidirectional method is fast and uses less memory than the unidirectional method, but is
(C1) Exact (C2) Unidirectional (C3) Bidirectional (C4) Exact (opposite-time)

(R1) Double Gyre

(R2) Pitching Plate

(R3) Pitching Plate

(R4) Unsteady ABC

FIG. 6: Graphical comparison of each method on four examples: (top row) positive-time FTLE of double gyre, (second row) positive-time FTLE of 2D pitching plate, (third row) negative-time FTLE of 2D pitching plate, (bottom row) negative-time FTLE of 3D ABC flow. Each figure shows the FTLE field after a number of iterations of the given method. The column of FTLE fields calculated using unidirectional composition, (C2), agree well with the exact FTLE fields computed using the standard method, (C1). The column of FTLE fields calculated using bidirectional composition, (C3) all have significant error which is aligned with the opposite-time coherent structures. The opposite-time FTLE fields are shown in the rightmost column, (C4), for comparison with the bidirectional method. FTLE fields computed for positive-time flow maps are blue and those computed for negative-time flow maps are red.

prone to large errors in the approximate flow map and does not accurately reproduce the FTLE field.

Contour plots of the FTLE fields computed after a number of iterations of each method are shown in Fig. 6. The FTLE fields computed with the unidirectional method agree with those computed using the standard method, as seen by comparing the first and second columns of Fig. 6. FTLE fields computed using the bidirectional method, shown in the third column, have large errors. It is interesting to note that these errors are aligned with coherent structures found in the opposite-time FTLE field, shown in the fourth column. An analysis of this coherent error is provided in Section V.

A. Example - Double Gyre

Figure 7 shows the $L_2$ and $L_\infty$ error of the forward-time FTLE field for the double gyre computed using the standard method with $T = 16$, as time-step $\Delta t$ and grid spacing $\Delta x$ are varied. At a given grid spacing, a reference FTLE field is computed using a sufficiently small
time-step, $\Delta t = 10^{-4}$, so that the FTLE field may be considered exact. For small enough time-step $\Delta t \approx .001$, the FTLE field error converges. All integrations are performed using a fixed time-step, fourth order Runge-Kutta scheme.

The flow map approximation methods are only faster than the standard method when used to compute a sequence of FTLE fields in time, as in the construction of frames for a movie. Figure 8 compares computation time and $L_2$ error vs. frame number (iteration #) for a sequence of FTLE fields of the double gyre, computed using the standard, unidirectional, and bidirectional methods. Each iteration produces an FTLE field which is a sequence of FTLE fields in time, as in the construction of frames. For small enough time-step $\Delta t$, the computation of FTLE field error converges. All integrations are performed using a fixed time-step, fourth order Runge-Kutta scheme.

In this example, the flow map duration is $T = 16$, the time spacing between each FTLE field is $h = 1$, and the time-step of integration is $\Delta t = .01$. The multi-tier unidirectional method uses four tiers. The first FTLE field takes roughly the same time to compute using each of the methods. However, for subsequent iterations, the unidirectional and bidirectional methods are significantly faster. The computation time of bidirectional method (a) increases with the number of iterations, $k$, because integrating back from $t = kh$ to the reference time $t = 0$ becomes more costly as $k$ increases, as seen in Fig. 3. After $T/2h = 8$ iterations of bidirectional method (a), it is advantageous to compute a new reference flow map using the standard method. This explains the breaks in the solid red curve in part (b) of Fig. 8, as the bidirectional method is exact at these iterations. Bidirectional method (b) overcomes this increasing cost vs. iteration by using the flow map from the current iteration as the reference flow map at the next iteration.

However, using an approximate flow map to compute the next approximation causes bidirectional method (b) to accumulate error more quickly than method (a). The unidirectional method is both the fastest and most accurate method in this comparison.

### B. Computational Resources

Again, consider a sequence of time-$T$ flow maps spaced $h$ apart, as might be required for an unsteady visualization. When there are many integration time-steps of size $\Delta t$ between each neighboring flow map, i.e. $\Delta t \ll h$, then the added cost of flow map composition becomes relatively small compared with the cost of integrating a time-$h$ flow map.

All methods take about the same amount of time to compute the first FTLE field in the sequence. For subsequent iterations, the standard method involves $(T/h) \times (h/\Delta t)$ integration steps for each new FTLE field, whereas the unidirectional method only requires $h/\Delta t$ integration steps, and the bidirectional method requires $2h/\Delta t$ integration steps. Assuming $\Delta t \ll h$, the speed-up of the unidirectional method over the standard method will increase as the number of frames in the animation per flow map duration, $T$. In other words, as $\Delta t/h \to 0$, the computation of $\Phi_{t_0+T}$ using the unidirectional method is $T/h$ times faster than using the standard method, and twice as fast as the bidirectional method.
In the examples above, all intermediate flow maps were stored in memory until no longer useful for future computations. Regardless of any parameters of the FTLE field animation, the standard and bidirectional methods must store a fixed number of flow maps. The standard method stores the single flow map $\Phi_{t_0}^{t_0+T}$, while the bidirectional method stores three maps: $\Phi_{t_0}^0$, $\Phi_T^0$, and $\Phi_{t_0}^{t_0+T}$. The unidirectional method, however, stores of every intermediate time-$h$ flow map $\Phi_{(k-1)h}$, of which there are $T/h$. Therefore, the memory requirement of the unidirectional method scales linearly with the upper-bound on its speed-up, $T/h$.

The memory usage of the unidirectional method scales with the dimension of the flow $D$, the spatial resolution $R$, and the possible computational speed up of the method $S$, given by $T/h$:

$$\text{Memory (GB)} \sim S \times D \times R^D$$  

$$= \frac{8 \text{ B/double}}{1024^3 \text{ B/GB}} \times \frac{T}{h} \times D \times R^D$$ (16) (17)

For example, a series of two dimensional, high-definition (1920 $\times$ 1080 resolution) FTLE fields may be computed using the unidirectional method with up to 100x speed up using approximately 3.1 GB of RAM. A three dimensional FTLE field with resolution $512 \times 256 \times 64$ may be computed with up to 100x speed up with approximately 19 GB of RAM.

In the pitching plate example, velocity fields are obtained from data files which are the output of a direct numerical simulation. Because loading velocity fields which are stored on disk is slow, it is important to minimize the number of file loads. In the pitching plate example, all of the velocity fields are loaded up-front and stored in memory throughout the computation. However, velocity fields are often too large to store them all in memory, for example in large 2D or 3D simulations, so that subsequent iterations of the methods require re-loading the same velocity field data from previous iterations. In practice, although loading data files is time consuming, it represents a fraction of the cost of particle integration.

The set $\Sigma_\alpha(\Phi)$, defined as the set of points $x$ with FTLE above a threshold value $\alpha$,

$$\Sigma_\alpha(\Phi) = \{x | \sigma(\Phi; x) > \alpha\},$$ (18)

is the collection of points where error will magnify the most through the map $\Phi$. The flow map approximations above all involve composing intermediate flow maps,

$$\Phi_2 \circ \Phi_1,$$ (19)

so it is important to know which points flow into $\Sigma_\alpha(\Phi_2)$ through the map $\Phi_1$. In other words, we want to describe the set $\Phi_1^{-1}(\Sigma_\alpha(\Phi_2)) = \{x | \Phi_1(x) \in \Sigma_\alpha(\Phi_2)\}$, and this is the subject of Section V A.

If the flow map $\Phi_2$ is defined on a regular grid $X_0$, it is necessary to pass trajectories of $\Phi_1$ through the interpolated map $\mathcal{I}(\Phi_2|_{X_0})$. This is the source of error in the flow map approximations, and this error is significant when the trajectories of $\Phi_1$ pass into the set $\Sigma_\alpha(\Phi_2)$, where FTLE is large. Using a nearest neighbor interpolation the interpolation error becomes particularly simple:

$$\Phi_2(\Phi_1(x)) \approx \mathcal{I}(\Phi_2|_{X_0})(\Phi_1(x)) = \Phi_2(\Phi_1(x) + \epsilon)$$ (20)

where $x \in X_0$, and $\epsilon$ is the difference between $\Phi_1(x)$ and its nearest neighbor in $X_0$. However, each approximate method has been tested with nearest neighbor, linear and bicubic spline interpolations with no significant qualitative change in results. The propagation of interpolation error using unidirectional and bidirectional composition is the subject of section V B.

### A. Accumulation of Particles

The main result of this section is that particles near the positive-time LCS (pLCS) flow into particles near the negative-time LCS (nLCS) in forward time, and vice-versa. This is observed in Figs. 9 and 10 for the pitching plate and double gyre examples.

Figure 9 shows particles in $\Sigma_{14}(\Phi_T)$ near the pLCS of the pitching plate example. As the particles convect downstream, they attract onto the nLCS. Compare with the first and last panel of the second row of Figure 6 to see what the pLCS and nLCS look like for this example. Similarly, Fig. 10 shows points in $\Sigma_{33}(\Phi_T)$ near the nLCS of the double gyre example being integrated in negative time until they attract onto the pLCS. Compare with the first and last panel of the first row of Fig. 6 to see the pLCS and nLCS of the double gyre.

The bottom panel of Fig. 10 is a zoom-in of the tangent of particles near a time-dependent saddle point at $T = -10$. A point $x(t)$ is a time-dependent saddle if it is at the transverse intersection of the pLCS and the nLCS. It is numerically observed that these saddles mediate transport of particles near the pLCS into particles near the nLCS in positive time.

Further, suppose that $x(t)$ persists as a time dependent saddle over a range of time $t \in (t_0 - T - \epsilon, t_0 + T + \epsilon)$.
The positive and negative-time FTLE properties of this point establish an exponential dichotomy which implies that $x(t)$ is a time-dependent hyperbolic trajectory [24]. This trajectory now carries with it all of the regular theory about saddles, including Hartman-Grobman and Stable/Unstable Manifold Theorems. In particular, we may consider the pLCS (resp. nLCS) to be the time-dependent stable (resp. unstable) manifold of $x(t)$.

Applying the Lambda lemma, it follows that a disk which intersects the pLCS transversely will attract arbitrarily close to a disk on the nLCS in positive-time, eventually. In the examples above, we observe a similar phenomena, namely, that in the neighborhood of a time-dependent saddle, the particles near the nLCS came from particles near the pLCS at an earlier time.

Similarly, it is possible to flow particles with large positive-time FTLE backward in time, and vice-versa, resulting in a set which resembles a positive-time LCS computed using a longer integration time. This is observed in Fig. 11, where particles in $\Sigma_\beta (\Phi_0^{15})$ are flowed forward along $\Phi_0^{15}$, resulting in a set which accumulates on the nLCS of the longer-time flow $\Phi_1^{15}$.

### B. Propagation of Interpolation Error

Consider the bidirectional composition of a positive-time flow map $\Phi^T$ with a negative-time flow map $\Phi^{-T}$, where error $\epsilon$ is introduced due to interpolation:

$$\Phi^T (\Phi^{-T}(x) + \epsilon) \approx \Phi^T (\Phi^{-T}(x)) + D\Phi^T (\Phi^{-T}(x)) \cdot \epsilon$$

$$= x + D\Phi^T (\Phi^{-T}(x)) \cdot \epsilon$$

(21)

The composition error is largest for points $x \in X_0$ where $D\Phi^T (\Phi^{-T}(x))$ is large. From Eqns. (3) and (4), we have the following relationship:

$$\|D\Phi^T(y)\| \geq e^{\alpha|T|} \text{ for } y \in \Sigma_\alpha (\Phi^T)$$

(22)

where $\|A\| = \max \frac{\|Ax\|_2}{\|x\|_2}$ is the maximum singular value of $A$. Thus, composition error is large at points $x$, where $y = \Phi^{-T}(x)$ is in the set $\Sigma_\alpha (\Phi^T)$, for large $\alpha$ and $T$.

Moreover, the results of the previous section indicate that points $x$ satisfying $\Phi^{-T}(x) \in \Sigma_\alpha (\Phi^T)$ originate in the set $\Sigma_\beta (\Phi^{-T})$ near the nLCS, in a neighborhood of a time-dependent saddle. Therefore, it is seen that the composition error will be largest at points $x \in \Sigma_\beta (\Phi^{-T})$, near the nLCS.

Now, consider the unidirectional composition of two positive-time flow maps, with interpolation error $\epsilon$:

$$\Phi^T (\Phi^T(x) + \epsilon) \approx \Phi^T (\Phi^T(x)) + D\Phi^T (\Phi^T(x)) \cdot \epsilon$$

$$= \Phi^{2T}(x) + D\Phi^T (\Phi^T(x)) \cdot \epsilon$$

(23)

Here the error is largest for points $x \in X_0$ where $D\Phi^T (\Phi^T(x))$ is large. Again, $D\Phi^T(y)$ is large when $y \in \Sigma_\alpha (\Phi^T)$, for sufficiently large $\alpha$ and $T$.

In unidirectional composition, because the pLCS is repelling in positive time, points $x \in X_0$ must be exactly in $\Phi^{-T} (\Sigma_\alpha (\Phi^T))$, or else they will repel away from the regions where error is magnified. Similarly for bidirectional composition, because the pLCS is attracting in negative time, points will attract toward the regions where error magnifies. For this reason, the unidirectional method is robust to interpolation error, while the bidirectional method amplifies this error.

FIG. 9: Particle trajectories of the set $\Sigma_{14}(\Phi_0^{15})$ for the pitching flat plate. Particles near the pLCS are integrated forward until they attract near the nLCS.
VI. CONCLUSIONS

A number of methods have been developed for the efficient computation of finite-time Lyapunov exponent (FTLE) fields in unsteady flows. In particular, the methods speed up the computation of a sequence of FTLE fields in time, used for frames of a movie, by approximating the particle flow map using information from neighboring times. The methods fall into two categories of flow map approximation based on composition of intermediate flow maps of the same time direction (unidirectional) or of both positive and negative-time directions (bidirectional). The main result is that the unidirectional method is both fast and accurate, and the computational savings over the standard method are proportional to the number of FTLE fields being computed per time \( T \). The unidirectional method provides one or two orders of magnitude computational savings over the standard method on the three example flows, as summarized in Table I.

The bidirectional methods are also fast, and use less memory than the unidirectional methods; however, bidirectional methods suffer from large errors which are concentrated along regions where the opposite-time FTLE field is large, in the vicinity of time-dependent saddle points. This coherent error was unexpected, but is explained by dynamical systems theory, since particles close
FIG. 11: Particle trajectories of the set $\Sigma_3(\Phi_0^{15})$ for the double gyre. Particles on the nLCS are flowed forward, shown in (a), resulting in a longer time nLCS, shown in (b).

to the pLCS near a time-dependent saddle will map into particles close to the nLCS in positive time. This result extends the relevance of Lagrangian coherent structure analysis to near identity particle maps in general.

The fast methods are implemented on three example velocity fields, chosen to represent typical fluid flows, and compared on the basis of computation time, accuracy and memory usage. The results of the method comparisons are summarized in Table I and Fig. 6. The unidirectional algorithm works well on 2D and 3D domains with either compact or spatially periodic domains. For open domains, as in the example of the pitching plate in a free stream velocity, the unidirectional method accurately computes the negative-time FTLE fields corresponding to the attracting set, or nLCS. However, error is introduced when computing the positive-time FTLE field, as particle trajectory information is lost downstream of the FTLE domain. This loss of information is not a problem when computing the nLCS because trajectory information upstream of the plate is well approximated using uniform flow. In experiments, however, velocity field data is often only available on a limited domain, which might correspond to the FTLE domain. In this case, the unidirectional and standard methods will produce matching positive-time FTLE fields.

There are a number of future directions which might arise from this work. First, FTLE algorithms lend themselves to parallelization, so it is conceivable that with further optimization, it will be possible to obtain real-time FTLE visualizations for interesting problems. It would also be interesting to extend the above methods to incorporate adaptive mesh refinement (AMR) as well as complex domain geometries. Additionally, it is important to more precisely determine how and when particles near the pLCS flow into particles near the nLCS in positive-time.

Appendix A: Notation

- $X_0 \subset \mathbb{R}^n$ - Discrete particle grid,
- $\Phi^b_{ab}$ - Particle flow map from time $t = a$ to time $t = b$,
- $\tilde{\Phi}$ - Approximation to the particle flow map $\Phi$,
- $\Phi|_{X_0}$ - Flow map restricted to a discrete grid $X_0$,
- $I(\Phi|_{X_0})$ - Interpolant of the flow map $\Phi|_{X_0}$, defined on a discrete grid $X_0$,
- $I\Phi \triangleq I(\Phi|_{X_0})$ - Shorthand for interpolation,
- $D\Phi$ - The Jacobian of the flow map $\Phi$,
- $\Delta$ - The Cauchy-Green deformation tensor,
- $\sigma(\Phi; x)$ - FTLE for flow $\Phi$ at the point $x$,
- pLCS - Positive-time Lagrangian coherent structure, which is a ridge of the FTLE field of $\Phi_{t_0}^{t_0+T}$; physically, this is a material line which attracts in negative time,
- nLCS - Negative-time Lagrangian coherent structure, which is a ridge of the FTLE field of $\Phi_{t_0}^{t_0-T}$; similarly, this is a material line which attracts in positive-time,
- $\Sigma_\alpha(\Phi) = \{x \mid \sigma(\Phi; x) > \alpha\}$ - Set of particles with FTLE above a given threshold $\alpha$; for $\alpha$ sufficiently large, this set represents a neighborhood of the dominant LCS ridges,
- $St$ - Strouhal number, defined as $St = \frac{fA}{U_\infty}$ where $f$ is frequency, $A$ is amplitude and $U_\infty$ is free stream velocity.

$\Sigma_3(\Phi_0^{15})$ 

(a) 

(b)
Appendix B: Example Velocity Fields

A summary of the attributes of each example velocity field is given in Table 2. Below is a description of how to compute the given velocity fields and an image of each corresponding FTLE field.

Double Gyre

The double gyre is an analytically defined velocity field which is time-periodic on the closed and bounded domain, \([0, 2] \times [0, 1]\). The stream-function is

\[
\psi(x, y, t) = A \sin(\pi f(x, t)) \sin(\pi y) \\
f(x, t) = \epsilon \sin(\omega t)x^2 + x - 2\epsilon \sin(\omega t)x
\]

which yields the following vector field

\[
u = -\frac{\partial \psi}{\partial y} = -\pi A \sin(\pi f(x)) \cos(\pi y) \\
v = \frac{\partial \psi}{\partial x} = \pi A \cos(\pi f(x)) \sin(\pi y) \frac{df}{dx}
\]

with maximum angle of attack, \(\alpha_{\text{max}} = 20^\circ\), and frequency \(f = .4\). The Strouhal number, \(St\), is a dimensionless pitching frequency given by:

\[
St = \frac{fA}{U_\infty} = .274 \tag{B4}
\]

where \(A = 2 \sin(20^\circ)\) is the amplitude of the plate's excursion, and \(U_\infty = 1\) is the free stream velocity of the uniform flow.

FIG. 12: FTLE field for double gyre, Eq. (B2), with \(A = .1, \omega = 2\pi/10, \epsilon = .25, T = 15\).

Pitching Flat Plate

The second example is the unsteady velocity field of a flat plate pitching in a uniform flow at low Reynolds number, \(Re = 100\). The plate pitches about its leading edge according to the following angle of attack motion:

\[
\alpha(t) = \alpha_{\text{max}} \sin(2\pi ft) \tag{B3}
\]

The motion of the plate is enforced using the multi-domain immersed boundary method of Taira & Colosnius [25], using a second-order Adams-Bashforth time-stepper. The output of the direct numerical simulation (DNS) is a time-sequence of velocity fields spaced .05 apart in non-dimensional time units. Each velocity field snapshot is defined on five nested grids. The finest grid covers a \(4 \times 4\) domain and the coarsest grid covers a \(64 \times 64\) domain, non-dimensionalized by chord length. Each grid has resolution \(200 \times 200\). This provides a large computational domain for integrating particle trajectories. Velocity fields from the DNS are stored on disk and are loaded for use in FTLE field computations.

Unsteady ABC Flow

The unsteady ABC flow is a 3D flow which is aperiodic in time, has spatially periodic boundary conditions, and whose velocity field is defined analytically as follows

\[
\dot{x} = (A + \frac{1}{2} t \sin(\pi t)) \sin z + C \cos y \tag{B5}
\]

\[
\dot{y} = B \sin x + (A + \frac{1}{2} t \sin(\pi t)) \cos z \tag{B6}
\]

\[
\dot{z} = C \sin y + B \cos x \tag{B7}
\]
All FTLE fields are computed on the periodic cube $X, Y, Z \in [0,1)$, where $x = 2\pi X$, $y = 2\pi Y$, and $z = 2\pi Z$.

FIG. 14: FTLE field for a unsteady ABC flow with $A = \sqrt{3}$, $B = \sqrt{2}$, $C = 1$ and $T = -8$. 


