An Eulerian Approach for Computing the Finite Time Lyapunov Exponent

Shingyu Leung[a]

[a] *Department of Mathematics,*
Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong.

Abstract

We propose efficient Eulerian methods for approximating the finite-time Lyapunov exponent (FTLE). The idea is to compute the related flow map using the level set method and the Liouville equation. There are several advantages of the proposed approach. Unlike the usual Lagrangian-type computations, the resulting method requires the velocity field defined only at discrete locations. No interpolation of the velocity field is needed. Also, the method automatically stops a particle trajectory in the case when the ray hits the boundary of the computational domain. The computational complexity of the algorithm is $O(\Delta x^{-(d+1)})$ with d the dimension of the physical space. Since there are the same number of mesh points in the x-t space, the computational complexity of the proposed Eulerian approach is optimal in the sense that each grid point is visited for only $O(1)$ time. We also extend the algorithm to compute the FTLE on a co-dimension one manifold. The resulting algorithm does not require computation on any local coordinate system and is simple to implement even for an evolving manifold.

Keywords: Eulerian method, Lagrangian coherent structure, Level Set Method, PDE on evolving surfaces
2000 MSC:

1. Introduction

Coherent structure is an important concept in understanding fluid motions. The goal is to segment the domain into different regions with similar behavior according to a quantity such as the strain, kinetic energy, or the vorticity. Since these mentioned quantities are measured locally at fixed given locations and also instantaneously at fixed given times, their corresponding coherent structures are classified as Eulerian coherent structure (ECS). Computationally, these coherent structures can be easily extracted from any of these Eulerian quantities. For example, vorticity can be computed from the velocity field defined on an underlying mesh. The coherent structure can then be identified by looking at the level lines of the resulting vorticity field. The implementation is straight-forward and the computation can be done systematically.

Another class of coherent structure is the Lagrangian coherent structure (LCS) which tries to partition the space-time domain into different regions according to a Lagrangian quantity advected along with passive tracers. Among many, a simple definition of LCS uses the finite-time Lyapunov exponent (FTLE) [15, 11, 12, 42, 21]. It measures the rate of separation between adjacent particles over a finite time interval with an infinitesimal perturbation in the initial location. For a given time, particles are first advected in the flow for a period of time to obtain the flow map which takes the initial particle location to it's arrival location. Mathematically, these particles in the extended phase space satisfy the following ordinary differential equation (ODE)

$$\dot{\mathbf{x}}(t) = \mathbf{u}(\mathbf{x}(t), t) \tag{1}$$

*Corresponding author.
Email address: masyleung@ust.hk (Shingyu Leung)

with the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$ and a Lipschitz velocity field $\mathbf{u} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$. We define the flow map $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ to be the mapping which takes the point $\mathbf{x}_0$ to the particle location at the final time $t = t_0 + T$, i.e $\Phi(\mathbf{x}_0; t_0, T) = \mathbf{x}(t_0 + T)$ with $\mathbf{x}(t)$ satisfies (1). Then the FTLE is computed using the norm of the Jacobian of this resulting flow map. Following the definition of Haller [11, 13, 14], one can see that LCS is closely related to the *ridges* of the FTLE fields. In particular, we refer any interested reader to a recent paper by Haller [14] for a careful study in the relationship between these two concepts. Even though efficient extraction of LCS from the FTLE is already an active area (see the adaptive mesh refinement (AMR) method for efficient extraction and visualization of the LCS in [39]), in this work we will concentrate only on developing efficient algorithms on calculating the FTLE. Moreover, the proposed algorithm may be applied to compute the FTLE for a wide range of time level or on a stationary or even an evolving manifold.

Recently there are vast research on applications of Lagrangian coherent structure based on the FTLE including fluid flow data in oceans [18, 42], hurricane structure visualization [40], flight data analysis [5, 45], gravity wave propagation [46] and some bio-inspired fluid flows [27, 10, 28], to name but a few recent examples. However, even with tremendous applications, there has not been much discussions on methods for efficient computations. Since LCS is long treated as a Lagrangian property of the fluid flow, majority of numerical methods are still based on Lagrangian ray tracing technique by solving (1) using a numerical integrator. For instance, to compute the FTLE for a given time $t_0 \in [T_s, T_f]$, one first uniformly discretizes the computational domain and then solves the above ray tracing system (1) with the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$ *forward* or *backward* in time for a period of time T which leads to the *forward* or *backward* FTLE. If the propagation time is long ($T \gg 1$) or the FTLE is required for large period of time ($|T_f - T_s| \gg 1$), the computational cost will be very expensive.

To reduce the computational cost, one way is to adaptively refine the computational mesh around the LCS so that most computational time is spent on resolving any fine feature. This reduces the computational cost in the spatial dimensions. Various adaptive methods have been recently proposed. For example, [7] has proposed adaptively refining the underlying Cartesian mesh locally so that the mesh size is $h/2^l$ for some adaptive level l with h is the grid size in the coarsest level. Adaptive method based on a triangular mesh has also been implemented in [20]. Graphics Processing Unit (GPU) has also been applied for such computations for efficient parallel computations [8]. However, all these methods are done on each individually time level. For the FTLE at a different time level, one has to shoot out another independent set of Lagrangian particles to compute the flow map.

An interesting implementation has been recently proposed in [3] which decomposes the flow map into a composition of maps of smaller time steps. This approach tries to reduce the computational cost in the time domain marching by recycling most of these maps in computing the FTLE at later time. However, such approach requires interpolation of these flow maps and it also needs to explicitly store all these intermediate maps. There could be problems in both the accuracy and the memory management.

In this paper, we first propose an Eulerian approach to compute the FTLE on a fixed Cartesian mesh. The idea is to determine the evolution using the Level Set Method [33] so that the flow map satisfies a Liouville equation. This resulting hyperbolic partial differential equation (PDE) can be solved by any well-established robust and high order accurate numerical methods.

There are several advantages of the proposed approach. The Lagrangian approach to FTLE requires velocity field defined not only on a given mesh points, but also at arbitrary locations in the computational domain. Indeed, high order reconstruction methods could be applied [19]. However, these methods could be expensive to use in practice. Moreover, since the velocity field might only be Lipschitz continuous, high order reconstruction might not be accurate at all. Our proposed approach on the other hand is similar to other Eulerian-type methods which require velocity field defined only on a fixed underlying mesh.

The second part of the algorithm provides an easy way to approximate the FTLE for long time propagation, i.e. $|T_f - T_s| \gg 1$. The computational cost is very cheap since the method requires solving only one single Liouville equation without constructing the flow map for later time at all. The overall computational complexity of the proposed Eulerian approach is of $O(\Delta x^{-d} \Delta t^{-1}) = O(\Delta x^{-(d+1)})$ with d is the dimension of the physical space. This is optimal in the sense that such order is the same as the total number of grid points in the extended phase space.

Another contribution of this paper is a simple way for computing the FTLE on a codimensional one

manifold $\mathcal{M} \subset \mathbb{R}^d$. The manifold itself could be stationary so that all particles stay on the same manifold for all time, or the hypersurface itself could evolve in time, i.e. $\mathcal{M} = \mathcal{M}(t)$. One possible way to determine the FTLE on a stationary manifold is proposed in [20] which explicitly parametrizes the surface $\mathcal{M}$ using unstructured mesh and then the FTLE is defined using the local coordinate system. In this paper, we propose a pure Eulerian way to compute the FTLE on an implicitly defined codimension one manifold. Using the Level Set Method, we implicitly represent the hypersurface and we derive a new algorithm to compute the FTLE without using any local coordinate system. All calculations will be done on the underlying Cartesian mesh. Several similar approaches have been introduced to solve various PDE's on surfaces. For example, Level Set Method was first introduced to solve elliptic problems on a manifold in [1]. The authors proposed a numerical projection operation to extend the diffusion operation from the interface to the whole computational space. [2] recently introduced a Level Set Method to determine eigenvalues of some elliptic operators on a codimensional one hypersurface. However, all these works concentrated only on solving an elliptic PDE on a fixed manifold. The current paper studies a Level Set Method for computing the FTLE on an evolving manifold.

The paper is organized as follow. In Section 2 we introduce the FTLE. We will also summarize typical Lagrangian-type methods for computing these quantities. Our corresponding Eulerian formulation is explained in Section 3. We then generalize the method to compute the FTLE on a codimension one manifold in section 5. Section 6 will show examples to demonstrate the efficiency of the proposed numerical approach.

2. Finite Time Lyapunov Exponent (FTLE)

2.1. Definition of FTLE

We consider a time dependent Lipschitz velocity field $\mathbf{u} : \Omega \times \mathbb{R} \to \mathbb{R}^d$ where $\Omega \subset \mathbb{R}^d$ is a bounded domain in the d-dimensional space. The trajectory of any particle in the phase space can be obtained by solving the following ordinary differential equation (ODE)

$$\mathbf{x}'(t; \mathbf{x}_0, t_0) = \mathbf{u}(\mathbf{x}(t; \mathbf{x}_0, t_0), t) \tag{2}$$

with the corresponding condition $\mathbf{x}(t_0; \mathbf{x}_0, t_0) = \mathbf{x}_0$. This condition could be an initial condition if we want to know the locations of the particle *forward* in time. We can also interpret it as a terminal condition if we trace the trajectory *backward* in time. t_0 in the above system is a reference time for the computation. Collecting the solutions to (2) at time $t = t_0 + T$ for all initial particle locations in Ω, we define the following flow map $\Phi : \Omega \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^d$,

$$\Phi(\mathbf{x}; t_0, T) = \mathbf{x}(T; \mathbf{x}_0, t_0) \tag{3}$$

which takes the particle from $\mathbf{x}(t_0; \mathbf{x}_0, t_0)$ to $\mathbf{x}(t_0 + T; \mathbf{x}_0, t_0)$ in a time period T.

To study the growth of an infinitesimal perturbation in an initial condition over a finite time period T, we have

$$\begin{aligned} \delta\mathbf{x}(T) &= \Phi(\mathbf{x} + \delta\mathbf{x}(0); t_0, T) - \Phi(\mathbf{x}; t_0, T) \\ &= \mathcal{D}\Phi(\mathbf{x}; t_0, T)\delta\mathbf{x}(0) + \text{ higher order terms} \end{aligned} \tag{4}$$

where $\mathcal{D}\Phi(\mathbf{x}; t_0, T)$ is the spatial gradient or the Jacobian of the flow map.

The leading order of the magnitude of this perturbation is given by

$$\|\delta\mathbf{x}(T)\| = \sqrt{\langle \delta\mathbf{x}(0), [\mathcal{D}\Phi(\mathbf{x}; t_0, T)]^* \mathcal{D}\Phi(\mathbf{x}; t_0, T)\delta\mathbf{x}(0) \rangle}\,, \tag{5}$$

where $[\cdot]^*$ denotes the adjoint or the transpose of a matrix. Denoting $\Delta(\mathbf{x}; t_0, T)$ the Cauchy-Green deformation tensor

$$\Delta(\mathbf{x}; t_0, T) = [\mathcal{D}\Phi(\mathbf{x}; t_0, T)]^* \mathcal{D}\Phi(\mathbf{x}; t_0, T)\,, \tag{6}$$

we obtain the largest strength deformation

$$\max_{\delta\mathbf{x}(0)} \|\delta\mathbf{x}(T)\| = \sqrt{\lambda_{\max}[\Delta(\mathbf{x}; t_0, T)]}\|\mathbf{e}(0)\| = \exp[\sigma^T(\mathbf{x}, t_0)|T|]\|\mathbf{e}(0)\|\,, \tag{7}$$

where $\mathbf{e}(0)$ aligns with the eigenvector associated with the largest eigenvalue of the deformation tensor $\lambda_{\max}[\Delta(\mathbf{x};t_0,T)]$. Using this quantity, we define the finite-time Lyapunov exponent (FTLE) using

$$\sigma^T(\mathbf{x},t_0)=\frac{1}{|T|}\ln\sqrt{\lambda_{\max}[\Delta(\mathbf{x};t_0,T)]}\,. \tag{8}$$

The absolute value of T reflects the fact that we can trace the particles both *forward* and *backward* in time. In the case when $T<0$, we are measuring the maximum stretch *backward* in time and this corresponding to the maximum compression *forward* in time. To distinguish different measures, we call $\sigma^T(\mathbf{x},t_0)$ the *forward* FTLE if $T>0$ and the *backward* FTLE if $T<0$. The value T in the definition of $\sigma^T(\mathbf{x},t_0)$ reflects the fact that the FTLE depends also on the time period T in computing the flow map. In the limit when $T\to\infty$, we recover the traditional definition of Lyapunov exponent.

The FTLE is time-dependent for general flows. If the velocity field is time-dependent, one has to recompute the flow map for different reference time t_0. Since one has to keep generating the flow maps for different t_k, this is why usual numerical approach is very time consuming.

2.2. Lagrangian approaches to the FTLE

Usual numerical methods in computing the flow map $\Phi(\mathbf{x};t_0,T)$ relies on ray tracing by solving (2) numerically on an initial Cartesian mesh at time $t=t_0$. For each grid point $\mathbf{x}=\mathbf{x}_i$, one solves (2) using high order ODE solvers for time up to $t=t_0+T$. After going through all mesh points in the computational domain, one constructs the deformation tensor (6) by finite differencing the *forward* flow map $\Phi(\mathbf{x}_i;t_0,T)$ on the initial Cartesian mesh at $t=t_0$. The *forward* FTLE can then be computed by the largest eigenvalue of the $d\times d$ matrix.

One difficulty is that the velocity field might only be discretely defined on a mesh. Since this Lagrangian approach follows ray paths by solving an ODE, typical implementation requires an extra high order interpolation routine applying to the discrete velocity field [19]. Despite the fact that this theoretically imposes a stronger regularity condition on the velocity field, this extra step might be very time-consuming.

The second concern is about the numerical accuracy when using a high order adaptive time integrator such as ode45 in MATLAB. Since the flow map at each grid locations are individually computed using the ODE solver, it is hard to reconcile the accuracies at different mesh points. When calculating the numerical derivatives and the resulting FTLE, error might be amplified and might seriously pollute in the final solution. We will demonstrate this behavior in the example section.

Another issue concerns about the computational efficiency. For a given $t=t_0$, one has to solve the ODE system for many different initial conditions in order to compute the flow map. For computing the FTLE at the next time step $t=t_1=t_0+\Delta t$ however, those rays obtained on the previous time step $t=t_0$ are all discarded. This process actually throws away many useful information on the flow map as described in [3]. To improve the computational efficiency, [3] has proposed an interesting implementation which tries to recycle part of those information obtained in earlier computations. The idea is to decompose the flow map operator $\Phi_{t_0}^{t_0+T}$ into a composition of maps

$$\Phi_{t_0}^{t_0+T}=\Phi_{t_{N-1}}^{t_N}\circ\cdots\circ\Phi_{t_1}^{t_2}\circ\Phi_{t_0}^{t_1}\,, \tag{9}$$

where $t_k=t_0+kT/N$ for $k=0,1,\cdots,N$. Numerically, since each map $\Phi_{t_k}^{t_{k+1}}$ is still computed using Lagrangian ray tracing, interpolation has to be done on each step to match ray trajectories at different time level. Defining the interpolation operator by $\mathcal{I}$, we have

$$\Phi_{t_0}^{t_0+T}=\Phi_{t_{N-1}}^{t_N}\circ\cdots\circ\mathcal{I}\Phi_{t_1}^{t_2}\circ\mathcal{I}\Phi_{t_0}^{t_1}\,. \tag{10}$$

This approach is computational efficient since the flow map $\Phi_{t_1}^{t_1+T}$ can be decomposed into

$$\Phi_{t_1}^{t_1+T}=\mathcal{I}\Phi_{t_N}^{t_{N+1}}\circ\cdots\circ\mathcal{I}\Phi_{t_2}^{t_3}\circ\Phi_{t_1}^{t_2}\,. \tag{11}$$

If the maps $\Phi_{t_k}^{t_{k+1}}$ for $k=1,\cdots,N$ are all stored once they are computed, we can form $\Phi_{t_1}^{t_1+T}$ by determining only $\Phi_{t_N}^{t_{N+1}}$. However, the memory requirement for such implementation could be extremely large if $N\gg 1$.

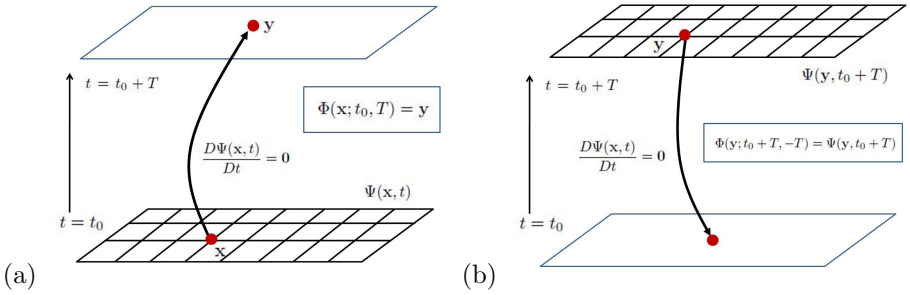

Figure 1: Lagrangian and Eulerian interpretations of the function Ψ. (a) Lagrangian ray tracing from a given grid location $\mathbf{x}$ at $t = t_0$. Note that $\mathbf{y}$ might be a non-grid point. (b) Eulerian values of Ψ at a given grid location $\mathbf{y}$ at $t = t_0 + T$ gives the corresponding take-off location at $t = t_0$. Note the take-off location might not be a mesh point.

This could be problematic in high dimensional high resolution simulations. We finally remark that if the flow is steady, i.e. time-independent, this method can be interpreted as a simple case of the phase flow method proposed in [4].

3. Eulerian formulation for FTLE

In this section, we will reformulate the above Lagrangian computations to an Eulerian formulation based on the Level Set Method [33] and Liouville equations. For simplicity, we will first concentrate on calculating the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0 + T)$ for some $T > 0$. The negative sign in this notation emphasizes that the flow map can be interpreted as tracing any particle in the negative time direction. The main idea is to introduce the Liouville equation to compute the *backward* flow map $\Phi(\mathbf{x}; t_0 + T, -T)$ on a uniform Cartesian mesh. Since the velocity field is required only on the mesh location, we do not need to interpolate the discrete velocity field. Similar technique has been widely use in the level set community [32] and has been successfully applied in a series of study in high frequency asymptotic solutions to the wave equation [36, 26, 25] and to the Schödinger equation [24].

The next step of the proposed algorithm is to propagate the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0 + T)$ *forward* in time from $t \in [t_0 + T, T_f]$ in order to obtain approximations to $\sigma^{-T}(\mathbf{x}, t)$ or $\sigma^{-(t-t_0)}(\mathbf{x}, t)$ for $t > t_0 + T$.

3.1. Constructing the backward flow map and the backward FTLE

We define a vectored value function $\Psi = (\Psi^1, \Psi^2, \cdots, \Psi^d) : \Omega \times \mathbb{R} \to \mathbb{R}^d$. At $t = t_0$, we initialize these functions by

$$\Psi(\mathbf{x}, t_0) = \mathbf{x} = (x^1, x^2, \cdots, x^d)\,. \tag{12}$$

These functions provide a labeling for any particle in the phase space at $t = t_0$. In particular, any particle initially located at $(\mathbf{x}_0, t_0) = (x_0^1, x_0^2, \cdots, x_0^d, t_0)$ in the extended phase space can be **implicitly** represented by the intersection of d codimension-1 surfaces represented by $\cap_{i=1}^d \{\Psi^i(\mathbf{x}, t_0) = x_0^i\}$ in $\mathbb{R}^d$. At the first glance this representation seems redundant. However, following the particle trajectory with $\mathbf{x} = \mathbf{x}_0$ as the initial condition in a given velocity field, any particle identity should be preserved in the Lagrangian framework and this implies that the material derivative of these level set functions is zero, i.e.

$$\frac{D\Psi(\mathbf{x}, t)}{Dt} = \mathbf{0}\,. \tag{13}$$

This implies the following level set equations, or the Liouville equations,

$$\frac{\partial \Psi(\mathbf{x}, t)}{\partial t} + (\mathbf{u} \cdot \nabla)\Psi(\mathbf{x}, t) = \mathbf{0} \tag{14}$$

with the initial condition (12). This **implicit** representation therefore embeds all path lines in the extended phase space. For instance, the trajectory of a particle initially located at $(\mathbf{x}_0, t_0)$ can be found by determining the intersection of d codimension-1 surfaces represented by $\cap_{i=1}^{d}\{\Psi^i(\mathbf{x}, t) = x_0^i\}$ in the extended phase space. Furthermore, this implicit representation provides a way to determine the *forward* flow map from $t = t_0$ to $t = t_0 + T$. In particular, the *forward* flow map at a grid location $\mathbf{x} = \mathbf{x}_0$ is given by

$$\Phi(\mathbf{x}_0; t_0, T) = \mathbf{y} \tag{15}$$

for $\mathbf{y}$ satisfies $\Psi(\mathbf{y}, t_0 + T) = \Psi(\mathbf{x}_0, t_0) \equiv \mathbf{x}_0$. Note that $\mathbf{y}$ in general is a non-mesh location. We have demonstrated a typical two dimensional scenario in figure 1 (a).

The solution to (14) contains much more information than what we have just interpreted above. We can use the solution $\Psi(\mathbf{x}, t)$ for $t \in \mathbb{R}$ to construct the flow map for **all** time, i.e. $\Phi(\mathbf{x}; t, T)$. For example, we assume that we have already solved the above Liouville equation to obtain the solutions at the time levels $t = t_1 = t_0 + \Delta t$ and $t = t_1 + T$. Since the points $(\Phi(\mathbf{x}_1; t_1, -\Delta t), t_0)$, $(\mathbf{x}_1, t_1)$ and $(\Phi(\mathbf{x}_1; t_1, T), t_1 + T)$ in the extended phase space are on the same ray trajectory, the function value of Ψ at these locations are the same. This means

$$\begin{aligned} \Psi(\mathbf{x}_1, t_1) &= \Psi(\Phi(\mathbf{x}_1; t_1, -\Delta t), t_0) \\ &= \Phi(\mathbf{x}_1; t_1, -\Delta t) \\ &= \Psi(\Phi(\mathbf{x}_1; t_1, T), t_1 + T)\,, \end{aligned} \tag{16}$$

and so the flow map can be obtained by

$$\Phi(\mathbf{x}_1; t_1, T) = \mathbf{y} \tag{17}$$

for $\mathbf{y}$ satisfies $\Psi(\mathbf{y}, t_1 + T) = \Psi(\mathbf{x}_1, t_1)$. This interpretation is in fact an Eulerian version of the approach proposed in [3]. However, since such Eulerian method still requires one to store the solution to (14) at **all** intermediate time, we will not pursue the computations in this direction in the current work.

Instead, we will interpret this level set implicit representation in the following way. Consider a given mesh location $\mathbf{y}$ in the phase space at time $t = t_0 + T$, as shown in figure 1 (b), i.e. $(\mathbf{y}, t_0 + T)$ in the extended phase space. Since the intersection $\cap_{i=1}^{d}\{\Psi^i(\mathbf{x}, t) = \Psi^i(\mathbf{y}, t_0 + T)\}$ represents the path line in the extended phase space passing through $(\mathbf{y}, t_0 + T)$, the level set functions $\Psi(\mathbf{y}, t_0 + T)$ represent the coordinates of the takeoff location at $t = t_0$ of a Lagrangian particle reaching $\mathbf{y}$ at $t = t_0 + T$. Therefore, these level set functions defined on a uniform Cartesian mesh in fact give the *backward* flow map from $t = t_0 + T$ to $t = t_0$, i.e.

$$\Phi(\mathbf{y}; t_0 + T, -T) = \Psi(\mathbf{y}, t_0 + T)\,. \tag{18}$$

Moreover, solution to the level set equations (14) for $t \in (t_0, t_0 + T)$ provides also backward flow maps for all intermediate times, i.e.

$$\Phi(\mathbf{y}; t, t - t_0) = \Psi(\mathbf{y}, t)\,. \tag{19}$$

With the *backward* flow map defined on a uniform mesh at $t = t_0 + T$, we can compute the deformation tensor and therefore the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0 + T)$ according to (6) and (8).

To summarize the procedure, we provide the following algorithm for computing the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0 + T)$ in the two-dimensional space $d = 2$.

Algorithm 1: Computing the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0 + T)$:

1. Discretize the computational domain

$$\begin{aligned} x_i = x_{\min} + (i-1)\Delta x \quad &, \quad \Delta x = \frac{x_{\max} - x_{\min}}{I - 1}\,, i = 1, 2, \cdots, I \\ y_j = y_{\min} + (j-1)\Delta y \quad &, \quad \Delta y = \frac{y_{\max} - y_{\min}}{J - 1}\,, j = 1, 2, \cdots, J \\ t_k = t_0 + k\Delta t \quad &, \quad \Delta t = \frac{T}{K}\,, k = 1, 2, \cdots, K\,. \end{aligned}$$

2. Initialize the level set functions

$$\begin{aligned}\Psi^1(x_i, y_j, t_0) &= x_i \\ \Psi^2(x_i, y_j, t_0) &= y_j\,.\end{aligned}$$

3. Solve the Liouville equations for each individual level set function

$$\begin{aligned}\frac{\partial \Psi^1}{\partial t} + (\mathbf{u}\cdot\nabla)\Psi^1 &= 0 \\ \frac{\partial \Psi^2}{\partial t} + (\mathbf{u}\cdot\nabla)\Psi^2 &= 0\end{aligned} \tag{20}$$

   for up to $t = t_K = t_0 + T$.
4. Compute the Cauchy-Green deformation tensor

$$\Delta(x_i, y_j; t_K, -T) = [\mathcal{D}\Psi(x_i, y_j, t_K)]^* \mathcal{D}\Psi(x_i, y_j, t_K)\,, \tag{21}$$

   with $\Psi = (\Psi^1, \Psi^2)^T$.
5. Determine the *backward* FTLE at $t = t_0 + T$ by computing the largest eigenvalue of the deformation tensor at each grid point (x_i, y_j)

$$\sigma^{-T}(x_i, y_j, t_K) = \frac{1}{T} \ln \sqrt{\lambda_{\max}[\Delta(x_i, y_j; t_K, -T)]}\,. \tag{22}$$

The Liouville equation in step 3 can be solved efficiently using any well-developed high order numerical methods like WENO5-TVDRK2 [34, 16]. Indeed, there is a higher order total variation diminishing Runge-Kutta (TVDRK) solver for time integration (TVDRK3). However such numerical scheme requires that the velocity field defined at non-mesh point in the time-direction and therefore we do not implement it here.

Now we discuss the boundary conditions for (20). For practical implementations, we might obtain the velocity field only within certain computational domain. In the Lagrangian implementation, one might simply stop the trajectory at the boundary. However, this approach is not trivial to perform in the Lagrangian implementation. If one has naively implemented this constraint by imposing a zero velocity field outside the computational domain, the resulting method might extend the velocity field discontinuously and this will introduce serious error in the integration. Another method is to interpolate the Lagrangian trajectory in time to obtain the location where the characteristic intersects with the computational domain.

Our Eulerian approach on the other hand can automatically impose such a boundary condition according to the direction of the flow. In particular, we impose the following boundary conditions for $t > t_0$,

$$\Psi(\mathbf{x}, t)|_{\mathbf{x}\in\partial\Omega} = \Psi(\mathbf{x}, t_0)|_{\mathbf{x}\in\partial\Omega} = \mathbf{x} \tag{23}$$

if $\mathbf{n}\cdot\mathbf{u} < 0$ where $\mathbf{n}$ is the outward normal of the boundary, and

$$\mathbf{n}\cdot\nabla\Psi^i(\mathbf{x}, t)|_{\mathbf{x}\in\partial\Omega} = 0 \tag{24}$$

for $i = 1, \cdots, d$ if $\mathbf{n}\cdot\mathbf{u} > 0$. The first constraint imposes the inflow condition for which all incoming information will carry the location where the corresponding characteristics enter the computational domain. Consider an interior grid point (x_i, y_j, t_k) on a characteristic hitting the computational boundary at $t \in [t_0, t_k)$ when tracing backward in time. The corresponding value of $\Psi(x_i, y_j, t_k)$ will contain the precise location where the particle entered the domain.

On the boundary with $\mathbf{n}\cdot\mathbf{u} > 0$, information is out-going from the interior and no boundary condition is required. For the ease of numerical implementations, we simply impose a non-reflective boundary condition so that no information on this portion of the boundary will interfere with function values at the interior.

When computing the Jacobian of the flow map on those mesh locations next to computational domain, we will use one-sided differencing away from the boundary to avoid influence by such boundary condition.

Indeed, as demonstrated in [45], such boundary conditions might seriously influence the FTLE. Stopping particle trajectories at the in-flowing boundary will turn boundaries into repelling invariant manifold, while stopping particle trajectories at the out-going boundary will turn boundaries into attracting invariant manifold. We refer interested readers to [45] for a better velocity extension which can eliminate any artificial LCS due to the boundary condition.

3.2. Constructing the forward flow map and the forward FTLE

For the *forward* FTLE $\sigma^T(\mathbf{x}, t_0)$, equation (15) indeed provides an algorithm to compute the *forward* flow map $\Phi(\mathbf{x}; t_0, T)$. However, the corresponding implementation requires a further interpolation and root-finding process to determine a non-mesh location $\mathbf{y}$ representing the particle arrival location at $t = T$. Here, we propose another way to determine the *forward* flow map by reversing the above process for the *backward* flow map. We initialize the level set functions at $t = t_0 + T$ by

$$\Psi(\mathbf{x}, t_0 + T) = \mathbf{x}\,, \tag{25}$$

and we solve the corresponding level set equations (14) *backward* in time. The *forward* FTLE at $t = t_0$ is determined by first determining the Jacobian of the resulting flow map and then computing the largest eigenvalue of the deformation tensor $\Delta(\mathbf{x}; t_0, T)$. Then the *forward* FTLE is formed by

$$\sigma^T(\mathbf{x}, t_0) = \frac{1}{T} \ln \sqrt{\lambda_{\max}[\Delta(\mathbf{x}; t_0, T)]}\,. \tag{26}$$

Note that in the original Lagrangian formulation, the *attracting* LCS or the *unstable* manifold is obtained by *backward* time tracing, while the *repelling* LCS or the *stable* manifold is computed by *forward* time integration. In this current Eulerian formulation, on the other hand, *forward* time integration of the Liouville equations gives the *attracting* LCS and *backward* time marching provides the *repelling* LCS.

3.3. Evolution of the FTLE in time

In the previous sections, we have discussed our Eulerian approach in computing both the *backward* flow map $\Phi^{-T}(\mathbf{x}; t_0+T)$ for some time interval $T > 0$ and also the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0+T)$. Generalization to the *forward* flow map and the corresponding *forward* FTLE have also been described. In this section, we will explain the way how we approximate $\sigma^{-T}(\mathbf{x}, t)$ or $\sigma^{-(t-t_0)}(\mathbf{x}, t)$ for an arbitrary $t > t_0 + T$ using $\sigma^{-T}(\mathbf{x}, t_0 + T)$.

Our approach is motivated by the following observation proven in [42] which is essentially a finite time approximation to the Oseledec's multiplicative ergodic theorem under some more restrictive assumptions.

Theorem 3.1 (Theorem 3.1 in [42]). *The traditional (forward) Lyapunov exponent is constant along trajectories. Also, the (forward) finite-time Lyapunov exponent becomes constant along trajectories for large integration times T.*

In the proof, the authors have estimated the difference between $\sigma^T(\mathbf{x}, t_0)$ and $\sigma^T(\Phi^s(\mathbf{x}; t_0), t_0 + s)$ for some arbitrary but fixed s,

$$|\sigma^T(\mathbf{x}, t_0) - \sigma^T(\Phi^s(\mathbf{x}; t_0), t_0 + s)| \le \frac{2|s|}{|T|} \max_{t^*} \sigma^{t^*-t_0}(\mathbf{x}, t_0) = O(|T|^{-1})\,. \tag{27}$$

This implies that along any particle trajectory, the material derivative of the (*forward*) FTLE is bounded by

$$\left|\frac{D\sigma^T(\mathbf{x}, t)}{Dt}\right| = \lim_{s\to 0} \frac{|\sigma^T(\Phi^s(\mathbf{x}; t), t + s) - \sigma^T(\mathbf{x}, t)|}{|s|} \le \frac{2}{|T|} \max_{t^*} \sigma^{t^*-t_0}(\mathbf{x}, t_0) = O(|T|^{-1}) \tag{28}$$

If T is large enough, we approximate the right hand side of the inequality by zero and so

$$\frac{D\sigma^T(\mathbf{x},t)}{Dt} = 0\,. \tag{29}$$

This means that the (*forward*) FTLE constructed by a **fixed** map of size T is almost constant along any Lagrangian particle trajectory, and the traditional Lyapunov exponent (for $T \to \infty$) is constant along any ray trajectory.

In the previous section, we have introduced an Eulerian approach to compute the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0+T)$ on the time level $t = t_0 + T$. The above theorem implies also that this *backward* FTLE is almost constant along a Lagrangian particle trajectory. Such condition provides a way to approximate the FTLE. In particular, if T is large enough, we can approximate the solution $\sigma^{-T}(\mathbf{x}, t)$ for $t > t_0 + T$ by simply solving the following Liouville equation

$$\frac{\partial\sigma^{-T}(\mathbf{x},t)}{\partial t} + \mathbf{u}\cdot\nabla\sigma^{-T}(\mathbf{x},t) = 0\,, \tag{30}$$

with the initial condition at $t = t_0 + T$ obtained by the algorithm derived in the previous section. Indeed, this Liouville equation has already been derived in [42] (equation (33)). However, such relation was derived solely to explain the Lagrangian property of the LCS and was used only to estimate the flux across it. To the best of our knowledge, no numerical approach has been developed to compute the FTLE based on this Eulerian formulation.

Using (27), we can estimate the error we made in approximating $\sigma^{-T}(\mathbf{x}, t)$ by solving the Liouville equation (30). Let S be the time difference from $t = T$. Since

$$\left|\sigma^{-T}(\Phi^S(\mathbf{x};t_0+T),t_0+T+S) - \sigma^{-T}(\mathbf{x},t_0+T)\right| \le \frac{2S}{T}\max_{t^*}\sigma^{t^*}(\mathbf{x},T) = O\left(\frac{S}{T}\right)\,, \tag{31}$$

the error in the approximation is linear in S.

There is a second interpretation of the solution from the Liouville equation (30). We can also treat the solution $\sigma(\mathbf{x}, t)$ as an approximation to the FTLE constructed by the flow map $\Phi(\mathbf{x}; t, -(t-t_0))$ which maps a point $(\mathbf{x}, t)$ in the extended phase space to it's takeoff location at time $t = t_0$. This means that the solution from (30) can be used to approximate $\sigma^{-(t-t_0)}(\mathbf{x}, t)$.

Corollary 3.2. *Let*

$$\tilde{\sigma}^{-T}(\mathbf{x},t) = \ln\lambda_{\max}[\Delta(\mathbf{x};t,-T)] \tag{32}$$

be a scaled backward FTLE so that

$$\sigma^{-T}(\mathbf{x},t) = \frac{1}{2|T|}\tilde{\sigma}^{-T}(\mathbf{x},t)\,. \tag{33}$$

Along any Lagrangian particle trajectory, the scaled backward FTLE $\tilde{\sigma}^{-(t-t_0)}(\mathbf{y}, t)$ satisfies

$$\left|\tilde{\sigma}^{-(t-t_0)}(\mathbf{y},t) - \tilde{\sigma}^{-T}(\mathbf{x},t_0+T)\right| = O(t-t_0-T) \tag{34}$$

for $\mathbf{y} = \Phi(\mathbf{x}; t_0 + T, t - t_0 - T)$ and for $t > t_0 + T$.

This implies that $\tilde{\sigma}^{-T}(\mathbf{x}, t_0+T)$ is a linear approximation to the scaled FTLE $\tilde{\sigma}^{-T-S}(\mathbf{y}, t_0+T+S)$ with the error depends linearly on the time difference S. To approximate the *backward* FTLE for $t > t_0 + T$, we only need to divide it by a factor of $1/2(T + S)$, i.e. we can approximate

$$\sigma^{-T-S}(\mathbf{y},t_0+T+S) = \frac{1}{2(T+S)}\tilde{\sigma}^{-T}(\mathbf{x},t_0+T) + O\left(\frac{S}{T+S}\right)\,. \tag{35}$$

We note that when T is large enough, both of these approaches provide good approximations to the FTLE. However, the error in the second approximation (34) is significantly smaller than that in the first approximation given in (31).

These two properties lead to the following algorithm for propagating the *backward* FTLE from $t = t_0+T$ *forward* in time.

Algorithm 2: Computing both the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0 + T + S)$ and $\sigma^{-T-S}(\mathbf{x}, t_0 + T + S)$ using $\sigma^{-T}(\mathbf{x}, t_0 + T)$ according to algorithm 1:

1. Discretize the computational domain

$$\begin{aligned} x_i &= x_{\min} + (i-1)\Delta x \quad , \quad \Delta x = \frac{x_{\max} - x_{\min}}{I-1}, i = 1, 2, \cdots, I \\ y_j &= y_{\min} + (j-1)\Delta y \quad , \quad \Delta y = \frac{y_{\max} - y_{\min}}{J-1}, j = 1, 2, \cdots, J \\ s_k &= t_0 + T + k\Delta s \quad , \quad \Delta s = \frac{T_f - t_0 - T}{K'}, k = 1, 2, \cdots, K' . \end{aligned}$$

2. Follow Algorithm 1 and obtain $\tilde{\sigma}(x_i, y_j, s_0)$ according to (32).
3. Solve the Liouville equations for the scaled *backward* FTLE

$$\frac{\partial \tilde{\sigma}}{\partial t} + (\mathbf{u} \cdot \nabla)\tilde{\sigma} = 0 \tag{36}$$

   using the initial condition $\tilde{\sigma}(x_i, j_j, s_0)$.
4. Scale the solution appropriately to approximate the *backward* FTLE using one of the following

$$\begin{aligned} \sigma^{-T}(x_i, y_j, s_k) &= \frac{1}{2T}\tilde{\sigma}(x_i, y_j, s_k) \\ \sigma^{t_0 - s_k}(x_i, y_j, s_k) &= \frac{1}{2(t_0 - s_k)}\tilde{\sigma}(x_i, y_j, s_k) . \end{aligned} \tag{37}$$

---

4. Advantages and limitations of the Eulerian formulation

In this section, we will discuss several properties of the proposed Eulerian formulation for computing the FTLE. Limitations of the proposed approach will also be described.

First we will compare the computational complexity of the proposed algorithms with a typical Lagrangian-type computation. Let d be the dimension of the domain, $N = O(\Delta x^{-1})$ be the number of mesh points in each spacial dimension, $M = T/\Delta t = O(\Delta t^{-1})$ be the number of time steps to march for $t = T$ and $K = (T_f - t_0)/\Delta t = O(\Delta t^{-1})$ be the total number of time steps to march until $t = T_f$. To compute the flow map $\Phi(\mathbf{x}; t_k, t_k + T)$ for each t_k for $k = 1, \cdots, K$, usual Lagrangian-type method requires shooting rays from N^d locations and solving each of these ODE system for a period of time T. The computational complexity for this step is $O(N^d M)$. Computing the FTLE from the flow map requires $O(N^d)$ operations. To compute the FTLE for all time steps will therefore need $O(N^d MK) = O(\Delta x^{-d}\Delta t^{-2})$ operations.

Now for our Eulerian formulation, we first solve d level set equations in the d-dimensional space for a period of time T. This takes $O(dN^d M)$ operations. Once we have obtained the solutions to these level set functions at $t = t_0 + T$, we can construct the FTLE at $t = t_0 + T$ (*backward* FTLE) and this step requires $O(N^d)$ computations. The next step in our algorithm is to propagate the FTLE according to the Liouville equation for a time period $t = T_f - t_0 - T$. The computational complexity for this step is $O(N^d(K - M))$. Since there is a Courant-Friedrichs-Lewy (CFL) condition associated to the numerical solution to a hyperbolic equation, we have $\Delta t = O(\Delta x)$. Therefore, the complexity of the overall algorithm is $O(dN^d M + N^d + N^d(K - M)) = O(N^d[(d-1)M + K + 1]) = O(\Delta x^{-(d+1)})$. Note that there are $O(N^d K)$ mesh points in the extended phase space $\Omega \times [t_0, T_f]$, our Eulerian approach is actually optimal in the sense that each point in the computational domain is visited for only $O(1)$ time.

One can compare the computational complexities of these approaches. The Eulerian algorithm is more efficient if the time-marching size in the Lagrangian method is chosen so that $\Delta t^{-2} > O(\Delta x^{-1})$, i.e. $\Delta t < O(\sqrt{\Delta x})$. Indeed, one might pick a large $\Delta t > O(\sqrt{\Delta x})$ in the Lagrangian approach since the timestep in an ODE solver is not restricted by the mesh size but the required accuracy in the solution. However, this means that Δt is controlled by the stiffness of the ODE system, which is unfortunately flow-dependent.

Indeed, the current Eulerian approach by solving PDE's requires that Δt is restricted by a CFL condition of $O(\Delta x)$. This seems to be problematic in practice. However, if the velocity field is obtained by a CFD solver, the same stability condition has already been imposed on the time marching step. Therefore, solving these extra linear PDE's will not significantly increase the overall computational time. Of course, if one has already obtained and has stored the velocity field, the Lagrangian approach might compute the FTLE solution in a shorter time. Since the ODE timestep is restricted solely by the stiffness of the ODE system based on the velocity field, one might be able to apply a much larger ODE timestep $\Delta t \gg O(\sqrt{\Delta x}) > O(\Delta x)$.

If the velocity field is not determined by a CFD solver but is obtained from real life measurements, Eulerian approaches proposed in this work might also require interpolation procedure as in the typical Lagrangian approach. However, unlike the Lagrangian approach where interpolation is done along the characteristics, these Eulerian approaches require interpolating the velocity field only on a uniform Cartesian mesh.

Next we consider the memory issues. Assuming that we have already obtained the velocity field from a computational fluid dynamic (CFD) code. Typical Lagrangian approaches to FTLE will require first to load and store the velocity field at all mesh points at the beginning of the FTLE computations because one would trace various characteristics in $\mathbb{R}^d$ by interpolating the stored velocity field. The memory requirement therefore depends on the resolution of the flow field and is given by $O(dP^{d+1})$ with d is the dimension of the physical space, and P is the number of mesh points in the time and also each spatial dimension. In the current proposed Eulerian approach on the other hand, one can directly incorporate the Liouville equation (14) into the CFL code such that the flow map is obtained at the same time when one is generating the velocity field. We do not have to obtain and store the whole flow field at all time levels ahead of the FTLE computations but only at the current and the next time levels. Therefore, we need only an extra $O(dN^d)$ memory with N is the number of mesh point in each spatial dimension.

Concerning the storage of the computed FTLE, typical Lagrangian approach can output the solution right away which implies a memory requirement of only $O(d)$. The Eulerian approach, on the other hand, requires an allocation of $O(dN^d)$ variables for storing the flow map $\Psi(\mathbf{x})$.

Numerical dissipation in the PDE formulation might cause inaccuracy. Since the flow map and the FTLE have shape features near LCS, the numerical dissipation in the PDE solver might smooth out such structure. In general such Eulerian approach will require high order numerical schemes to accurately capture the solution or fine mesh near such feature. We will demonstrate this effect in Section 6.1. Adaptive mesh refinement [29] based on tree data structure might help to further improve the computational efficiency.

For the approximation we made in Section 3.3, an order of $O(S/T)$ error is introduced in the computations. Clearly, such error is not desirable for large S since newly created LCS may be missing in the resulting FTLE. This may give misleading results especially in aperiod flows. Therefore, it is advised that results from algorithm 2 should be carefully interpreted when used. One possible improvement is to reapply algorithm 1 once every period of $S = O(T)$ so that the error we introduce in algorithm 2 is bounded. This will be further studied in the future.

5. FTLE on an evolving codimensional one manifold

In this section, we will discuss the computation of FTLE on a codimensional one manifold. An approach in this direction was proposed in [20] in which the manifold was parametrized by an adaptive triangular mesh. To determine the deformation matrix defined on the manifold, particles were seeded on the interface and the Jacobian was formed by using a local coordinate system. This is unfortunately difficult to do in practice, especially in high dimensions. Furthermore, if the particles does not stay on the same manifold in time, i.e. the manifold is time-dependent, the approach proposed in [20] could be very expensive since one

might need to reparametrize the manifold at some later time. In this paper, we consider the case where the manifold itself moves along with the flow. The proposed approach can naturally compute the Jacobian of the flow on an evolving manifold and it does not require a local coordinate system to compute the deformation matrix.

We consider a codimension one manifold $\mathcal{M}(t)$ in $\mathbf{R}^d$ evolving in time under the velocity field $\tilde{\mathbf{u}} : \mathcal{M} \times t \to \mathbb{R}^d$. Using the level set method [33], we represent the codimension one manifold implicitly by the zero level set of a level set function $\phi(\mathbf{x}, t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$, i.e.

$$\mathcal{M}(t) = \{\mathbf{x} : \phi(\mathbf{x}, t) = 0\}\,. \tag{38}$$

In this Eulerian framework, the evolution of the manifold satisfies the level set equation

$$\phi_t + \mathbf{u} \cdot \nabla\phi = 0 \tag{39}$$

where $\mathbf{u}$ is the velocity field extended off the manifold $\mathcal{M}$ at each time step by

$$\mathbf{u}_\tau + \mathrm{sgn}(\phi)(\mathbf{n} \cdot \nabla)\mathbf{u} = \mathbf{0} \tag{40}$$

with the boundary condition $\mathbf{u}(\mathcal{M}) = \tilde{\mathbf{u}}(\mathcal{M})$, $\mathrm{sgn}(x)$ is the signum function and $\mathbf{n} = \nabla\phi/|\nabla\phi|$ is the normal of the manifold $\mathcal{M}$ [33, 41, 32].

To avoid explicit computing the local coordinate systems for each point on the manifold $\mathcal{M}$, we propose the following strategy which involves computation in the whole $\mathbb{R}^d$ space rather than in a local coordinate system of $\mathcal{M}$. A local computation is possible but will not be fully discussed here. The idea is to introduce a normal extension of the flow map and an extra singular value to the deformation matrix. For simplicity, we first consider computing the *backward* FTLE $\sigma^{-T}(\mathbf{x}, t_0)$ for $\mathbf{x} \in \mathcal{M}(t_0 + T)$.

Following the previous approach, we obtain the flow map by solving the following Liouville equation for

$$\frac{\partial\Psi(\mathbf{x}, t)}{\partial t} + (\mathbf{u} \cdot \nabla)\Psi(\mathbf{x}, t) = \mathbf{0}\,, \tag{41}$$

with the initial condition $\Psi(\mathbf{x}, t_0)|_{\phi^{-1}(0)} = \mathbf{x}$ and $\mathbf{u}$ is the extended velocity field we defined above. Note that the PDE is solved in the whole computational space and therefore it requires an initial condition defined not only on $\{\phi = 0\}$ but also everywhere in the whole computational domain. There are various ways to extend the initial condition on the manifold to the whole computational domain. One trivial way is to simply use $\Psi(\mathbf{x}, t_0) = \mathbf{x}$. Once we obtain the flow map at time $t = t_0 + T$, we could extract the zero level set which gives $\mathcal{M}$ and also the take-off location of each particle on the interface. To determine the FTLE however, one has to extract the flow map defined on $\mathcal{M}$ and then calculate any necessary component on a local coordinate system.

In this paper, however, we propose another extension so that we have an easier way for constructing the deformation matrix. To define Ψ away from the manifold $\mathcal{M}$ at $t = t_0$, we require $\mathbf{n} \cdot \nabla\Psi = 0$ with $\mathbf{n} = \nabla\phi/|\nabla\phi|$ is the normal of the manifold.

To illustrate the idea, we first consider the following simple two dimensional case where the manifold is a circle of radius r_0 centered at the origin. At $t = 0$, the deformation tensor defined **on** the one-dimensional manifold should be identity. Now, using the proposed normal extension, we obtain the following initial condition $\Psi(x, y, 0) = (\Psi^1(x, y, 0), \Psi^2(x, y, 0))$

$$\begin{aligned} \Psi^1(x, y, 0) &= \frac{r_0 x}{\sqrt{x^2 + y^2}} \\ \Psi^2(x, y, 0) &= \frac{r_0 y}{\sqrt{x^2 + y^2}}\,. \end{aligned} \tag{42}$$

The corresponding Jacobian is

$$\mathcal{D}\Psi = \frac{r_0}{(x^2 + y^2)^{3/2}} \begin{pmatrix} y^2 & -xy \\ -xy & x^2 \end{pmatrix} \tag{43}$$

and the eigenvalues of the resulting deformation matrix on the manifold $\phi = 0$ are therefore $\lambda_0 = 0$ and $\lambda_1 = 1$ with their corresponding eigenvectors $\mathbf{w}_0 = (x, y)^T$ and $\mathbf{w}_1 = (y, -x)^T$. The largest eigenvalue is λ_1 which gives the correct FTLE. We note that the other eigenvalue is zero. This implies that any initial location perturbation in the normal direction of the manifold, i.e. $\mathbf{n} = (x, y)^T = \mathbf{w}_0$, has no contribution to the deformation matrix and therefore the information in the normal direction will not contribute to the FTLE. For a general manifold, we obtain such an initial condition by solving

$$\Psi_\tau + \text{sgn}(\phi)(\mathbf{n} \cdot \nabla)\Psi = \mathbf{0} \tag{44}$$

with the initial condition at $\tau = 0$

$$\Psi(\mathbf{x}, \tau = 0) = \mathbf{x}\,. \tag{45}$$

Since we require the solution in a local neighborhood of the manifold, we do not need to solve this PDE until we obtain the steady state solution. Numerically, we can simply solve the above hyperbolic PDE for several iterations in the τ-direction.

Now we consider a later time $t = t_0 + T$. If we follow the approach we have introduced above and simply solve the flow map everywhere in the computational domain, the perturbation in the normal direction of the manifold $\mathcal{M}(t_0 + T)$ will unfortunately contribute to the deformation matrix for general flows since $(\mathbf{n} \cdot \nabla)\Phi$ is in general non-zero. Therefore at each time step, we first extend the flow map off from the manifold by solving (44) so that the map is constant in the normal direction of the manifold. Once again, this normal extension implies a zero singular value in this normal direction of the manifold. When we compute the deformation tensor, this flow map extension will guarantee that any perturbation in the normal direction introduces only a zero singular value and it has no contribution to the all other singular values of the Jacobian. In particular, without lost of generality we assume the local coordinate system $\{\mathbf{e}_1, \cdots, \mathbf{e}_{d-1}\}$ for $\mathcal{M}$ at a given point $\mathbf{x}_0$. The Jacobian matrix of the map Ψ_{loc} at $\mathbf{x} = \mathbf{x}_0 \in \mathcal{M}$ has the following singular value decomposition (SVD),

$$\mathcal{D}\Psi_{\text{loc}} = U\Sigma V^* \tag{46}$$

with $\Sigma = \text{diag}(\sigma_1, \cdots, \sigma_{d-1})$ is a diagonal matrix, $U = (\mathbf{u}_1 \cdots \mathbf{u}_{d-1})$ and $V = (\mathbf{v}_1 \cdots \mathbf{v}_{d-1})$ are some orthogonal matrices, $(\cdot)^*$ is the vector or matrix transpose and $\mathbf{u}_i, \mathbf{v}_i \in \mathbf{R}^{d-1}$ are vectors in the local coordinates system at $\mathbf{x}_0$. Now, if we extend the flow map away from $\mathcal{M}$ in the normal direction according to (44), i.e. adding $\mathbf{e}_d = \mathbf{n}$ to the local coordinate system, the Jacobian matrix of the flow map in the whole $\mathbb{R}^d$ space has the following SVD decomposition

$$\mathcal{D}\Psi = \left(\begin{array}{c|c} \mathcal{D}\Psi_{\text{loc}} & \mathbf{0} \\ \hline \mathbf{0}^* & 0 \end{array}\right) = \tilde{U}\tilde{\Sigma}\tilde{V}^* \tag{47}$$

with

$$\tilde{U} = \left(\begin{array}{c|c} U & \mathbf{0} \\ \hline \mathbf{0}^* & 1 \end{array}\right), \; \tilde{V} = \left(\begin{array}{c|c} V & \mathbf{0} \\ \hline \mathbf{0}^* & 1 \end{array}\right) \tag{48}$$

and $\tilde{\Sigma} = \text{diag}(\sigma_1, \cdots, \sigma_{d-1}, 0)$. Therefore, by extending the map from the manifold $\mathcal{M}$ in the normal direction, we introduce only one zero singular value to the new Jacobian $\mathcal{D}\Psi$. The largest singular value remains unchanged and it is the same as that computed on the manifold using the local coordinates system.

Now, once we have obtained this *backward* flow map, we compute the *backward* FTLE at $t = t_0 + T$. For time $t > t_0 + T$, we follow the same procedure as above and propagate it *forward* in time according to the Liouville equation (30).

To compute the *forward* FTLE, we follow a similar procedure. However, since we require an implicit representation of the manifold on the final time $t = T_f$ given the level set function at $t = t_0$, we will first solve the level set equation for ϕ forward in time for $t \in (t_0, T_f]$. If we directly reverse the above procedure, we will back propagate this level set function from $t = T_f$ to $t = t_0$ which is unnecessary. Instead, we will store the level set function at various times $t = \tilde{t}_{\tilde{k}}$ when we want to visualize the FTLE. We summarize the procedure here.

Algorithm 3: Computing the *forward* FTLE $\sigma^T(\mathbf{x}, t_0)$ for $\mathbf{x} \in \mathcal{M}(t_0)$ using a pure PDE approach:

1. Discretize the computational domain

$$x_i = x_{\min} + (i-1)\Delta x \quad , \quad \Delta x = \frac{x_{\max} - x_{\min}}{I-1}\,, i = 1, 2, \cdots, I$$
$$y_j = y_{\min} + (j-1)\Delta y \quad , \quad \Delta y = \frac{y_{\max} - y_{\min}}{J-1}\,, j = 1, 2, \cdots, J.$$

2. Initialize the level set function $\phi(x_i, y_j, t_0)$ at $t = t_0$ so that $\mathcal{M} = \{\phi = 0\}$.
3. Solve the level set equation

$$\frac{\partial \phi}{\partial t} + (\mathbf{u} \cdot \nabla)\phi = 0 \tag{49}$$

for up to $t = t_0 + T$.
4. Initialize the level set functions at $t = t_0 + T$

$$\begin{aligned} \Psi^1(x_i, y_j, T_f) &= x_i \\ \Psi^2(x_i, y_j, T_f) &= y_j\,. \end{aligned}$$

5. Extend Ψ^1 and Ψ^2 away from the manifold by

$$\begin{aligned} \frac{\partial \Psi^1}{\partial \tau} + \mathsf{sgn}(\phi)(\mathbf{n} \cdot \nabla)\Psi^1 &= 0 \\ \frac{\partial \Psi^2}{\partial \tau} + \mathsf{sgn}(\phi)(\mathbf{n} \cdot \nabla)\Psi^2 &= 0 \end{aligned} \tag{50}$$

with $\mathbf{n} = \nabla\phi/|\nabla\phi|$.
6. Solve the Liouville equations for each individual level set function backward in time

$$\begin{aligned} \frac{\partial \Psi^1}{\partial t} + (\mathbf{u} \cdot \nabla)\Psi^1 &= 0 \\ \frac{\partial \Psi^2}{\partial t} + (\mathbf{u} \cdot \nabla)\Psi^2 &= 0 \end{aligned} \tag{51}$$

from $t = t_0 + T$ to $t = t_0$.
7. Extend the map off the manifold at $t = t_0$ by

$$\begin{aligned} \frac{\partial \Psi^1}{\partial \tau} + \mathsf{sgn}(\phi)(\mathbf{n} \cdot \nabla)\Psi^1 &= 0 \\ \frac{\partial \Psi^2}{\partial \tau} + \mathsf{sgn}(\phi)(\mathbf{n} \cdot \nabla)\Psi^2 &= 0 \end{aligned} \tag{52}$$

with $\mathbf{n} = \nabla\phi/|\nabla\phi|$.
8. Compute the Cauchy-Green deformation tensor at $t = t_0$

$$\Delta(x_i, y_j; t_0, T) = [\mathcal{D}\Psi(x_i, y_j, t_0)]^* \mathcal{D}\Psi(x_i, y_j, t_0)\,, \tag{53}$$

with $\mathcal{D}\Phi$ computed using the Cartesian coordinates.
9. Determine the *forward* FTLE at $t = t_0$ by computing the largest eigenvalue of the deformation tensor at each grid point (x_i, y_j)

$$\tilde{\sigma}^T(x_i, y_j, t_0) = \ln \lambda_{\max}[\Delta(x_i, y_j; t_0, T)]\,. \tag{54}$$

In this algorithm, the computation is done on the whole $\mathbb{R}^d$ space rather than on the $d-1$ dimensional manifold. The overall computational complexity is indeed of order $O(\Delta x^{-(d+1)})$ as in the algorithm we proposed earlier in Section 3. Note however that since we care about the solution in a small neighborhood

of the manifold $\mathcal{M}(t)$, we can speed up the whole computation by applying the local level set technique as in [35]. The idea is to concentrate the computational power in a local neighborhood of the evolving manifold $\mathcal{M}(t)$ by computing any solution only if $|\phi(\mathbf{x},t)| < \epsilon\Delta x$. By this local level set procedure, the overall computational complexity can be reduced to $O(\Delta x^{-d})$.

Indeed, one might also consider the following Lagrangian tracing technique on an Eulerian mesh for computing the *forward* FTLE. There is a very important difference in this algorithm from the approach in [20]. We do not parametrize the manifold explicitly. Instead of shooting Lagrangian particles from the manifold at the initial time, we discretize the underlying mesh uniformly and then using those grid locations near the manifold as the initial condition in the ray tracing system. We do not require a complicated triangulation of the surface. Once we have obtained the flow map in a neighborhood of the interface, we follow the above approach to extend the flow map from the implicit interface. Note however that the main purpose of the current paper is to propose pure Eulerian formulations for computing the FTLE, we will not concentrate on this Lagrangian formulation but will simply state it here for completeness.

Algorithm 4: Computing the *forward* FTLE $\sigma^T(\mathbf{x},t_0)$ for $\mathbf{x} \in \mathcal{M}(t_0)$ with the surface implicitly represented by a level set function and the flow map computed by Lagrangian ray tracing:

1. Discretize the computational domain

$$x_i = x_{\min} + (i-1)\Delta x \quad , \quad \Delta x = \frac{x_{\max}-x_{\min}}{I-1}, i = 1,2,\cdots,I$$
$$y_j = y_{\min} + (j-1)\Delta y \quad , \quad \Delta y = \frac{y_{\max}-y_{\min}}{J-1}, j = 1,2,\cdots,J.$$

2. Initialize the level set function $\phi(x_i,y_j,t_0)$ at $t=t_0$ so that $\mathcal{M} = \{\phi = 0\}$.
3. Solve the Lagrangian tracing system

$$\begin{aligned} x'(t) &= u \\ y'(t) &= v \end{aligned} \tag{55}$$

   for up to $t = t_0 + T$ with the initial conditions $\mathbf{x}(t_0) = (x(t_0),y(t_0))^T = (x_i,y_j)^T$ if $|\phi(x_i,y_j)| < \epsilon$.
4. Construct the flow map $\Psi(x_i,y_j;t_0,T) = \mathbf{x}(t_0+T)$.
5. Extend Ψ^1 and Ψ^2 away from the manifold by

$$\begin{aligned} \frac{\partial \Psi^1}{\partial \tau} + \mathrm{sgn}(\phi)(\mathbf{n}\cdot\nabla)\Psi^1 &= 0 \\ \frac{\partial \Psi^2}{\partial \tau} + \mathrm{sgn}(\phi)(\mathbf{n}\cdot\nabla)\Psi^2 &= 0 \end{aligned} \tag{56}$$

   with $\mathbf{n} = \nabla\phi/|\nabla\phi|$.
6. Compute the Cauchy-Green deformation tensor at $t=t_0$

$$\Delta(x_i,y_j;t_0,T) = [\mathcal{D}\Psi(x_i,y_j,t_0)]^* \mathcal{D}\Psi(x_i,y_j,t_0)\,, \tag{57}$$

   with $\mathcal{D}\Psi$ computed using the Cartesian coordinates.
7. Determine the *forward* FTLE at $t=t_0$ by computing the largest eigenvalue of the deformation tensor at each grid point (x_i,y_j)

$$\tilde{\sigma}^T(x_i,y_j,t_0) = \ln \lambda_{\max}[\Delta(x_i,y_j;t_0,T)]\,. \tag{58}$$

In the special case where the manifold is time-independent, i.e. $\mathbf{u}\cdot\mathbf{n} = 0$ and $\mathcal{M}(t) \equiv \mathcal{M}(t=t_0)$, we do not need to solve the level set equation (49). The rest of the algorithm follows.

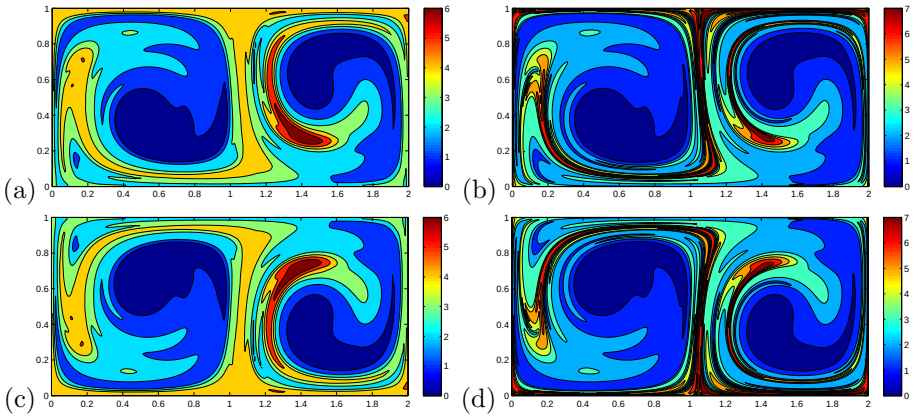

Figure 2: (Example 6.1) (a) The *backward* FTLE using $\Delta x = \Delta y = 1/128$ at $t = 15$, $T = 5$ and $S = 10$. (b) The *backward* FTLE using $\Delta x = \Delta y = 1/512$ at $t = 15$, $T = 5$ and $S = 10$. (c) The *forward* FTLE using $\Delta x = \Delta y = 1/128$ at $t = 0$, $T = 5$ and $S = 10$. (d) The *forward* FTLE using $\Delta x = \Delta y = 1/512$ at $t = 0$, $T = 5$ and $S = 10$.

6. Numerical Examples

In this section we will apply our Eulerian formulation to demonstrate the computational efficiency of the method. Indeed the original definition of the FTLE is given by (8),

$$\sigma^T(\mathbf{x}, t_0) = \frac{1}{2|T|} \ln \lambda_{\max}[\Delta(\mathbf{x}; t_0, T)] \tag{59}$$

we will show only the scaled version

$$\tilde{\sigma}(\mathbf{x}, t_0) = \ln \lambda_{\max}[\Delta(\mathbf{x}; t_0, T)] \tag{60}$$

by ignoring the scaling factor $1/2|T|$. The main reason is that our *backward* FTLE algorithm can in fact compute all $\sigma^{-T}(\mathbf{x}, T)$ and $\sigma^{-T}(\mathbf{x}, t)$ and/or $\sigma^{-(t-t_0)}(\mathbf{x}, t)$ for $t > t_0 + T$. It will be confusing to normalize the FTLE by a fixed factor of $1/|T|$ in both time regimes.

In those three dimensional cases, we will extract the zero level set using the MATLAB functions isosurface and patch. To graph the FTLE on the zero level set, we will also use the functions isonormals and isocolors.

6.1. Double-Gyre flow

This example is taken from [42] to describe a periodically varying double-gyre. The flow is modeled by the following stream-function

$$\psi(x, y, t) = A \sin[\pi f(x, t)] \sin(\pi y)\,, \tag{61}$$

where

$$\begin{aligned} f(x,t) &= a(t)x^2 + b(t)x\,, \\ a(t) &= \epsilon \sin(\omega t)\,, \\ b(t) &= 1 - 2\epsilon \sin(\omega t)\,. \end{aligned} \tag{62}$$

The velocity field can be obtained by $u(x, y) = -\partial\psi/\partial y$ and $v(x, y) = \partial\psi/\partial x$. In this example, we follow [42] and use $A = 0.1$, $\omega = 2\pi/10$ and $\epsilon = 0.1$.

In figure 2 (a), we discretize the domain $[0, 2] \times [0, 1]$ using 257 grid points in the x-direction and 129 grid points in the y-direction. This gives $\Delta x = \Delta y = 1/128$. We first solve the level set equations from $t = t_0 = 0$ to $t = T = 5$ and then construct the scaled *backward* FTLE, $\tilde{\sigma}(\mathbf{x}, T)$. Once we have obtained this quantity, we switch to another Liouville equation and propagate $\tilde{\sigma}(\mathbf{x}, t)$ up to $T_f = 15$. Since we are

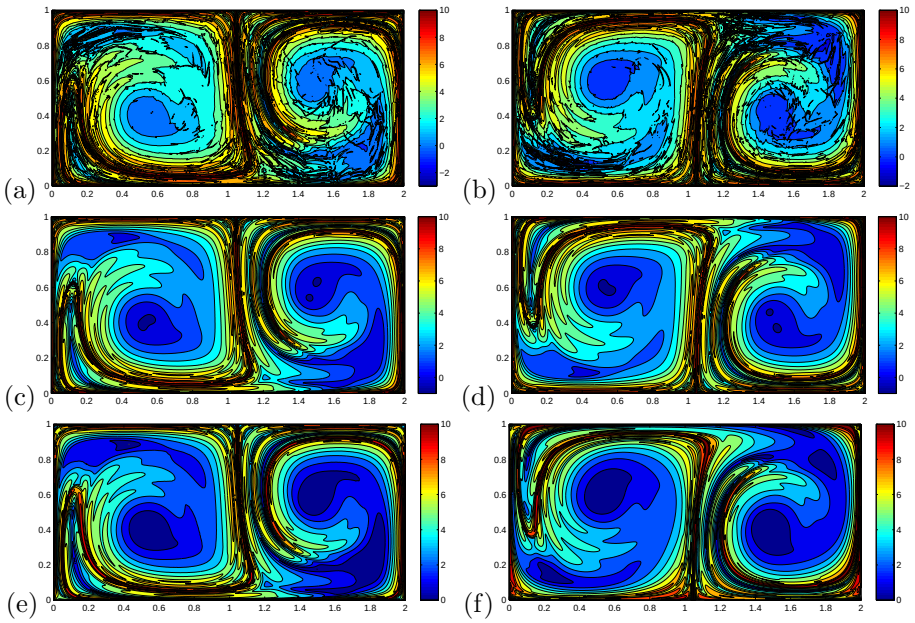

Figure 3: (Example 6.1) (a) The *backward* FTLE with $\Delta x = \Delta y = 1/512$ at $t = 15.0$ using the Lagrangian approach (MATLAB function ode45) with $T = 15$. (b) the *forward* FTLE with $\Delta x = \Delta y = 1/512$ at $t = 0.0$ using the Lagrangian approach (MATLAB function ode45) with $T = 15$. (c) The *backward* FTLE with $\Delta x = \Delta y = 1/512$ at $t = 15.0$ using the Lagrangian approach (RK4 with a fixed timestep) with $T = 15$. (d) The *forward* FTLE with $\Delta x = \Delta y = 1/512$ at $t = 0.0$ using the Lagrangian approach (RK4 with a fixed timestep) with $T = 15$. (e) The *backward* FTLE with $\Delta x = \Delta y = 1/512$ at $t = 15.0$ using the proposed Eulerian approach with $T = 15$. (f) The *forward* FTLE with $\Delta x = \Delta y = 1/512$ at $t = 0.0$ using the proposed Eulerian approach with $T = 15$.

advecting the level set functions *forward* in time, these figures are actually showing the scaled *backward* FTLE. We have also used some finer meshes for computing the level set functions. Figure 2 (b) shows the computed FTLE at $t = 15$ and $T = 5$ using a mesh of $\Delta x = \Delta y = 1/512$. As we can see, we recover the finer features in the FTLE field by refining the underlying computational mesh.

Next we compute the scaled *forward* FTLE. We solve the level set equation *backward* in time from $t = T_f = 15$ for a time of $T = 5$, i.e. we obtain the solutions to $(\Psi^1(x, y, t), \Psi^2(x, y, t))$ for $t \in [10, 15]$ using the initial condition $\Psi^1(x, y, T_f) = x$ and $\Psi^2(x, y, T_f) = y$. Once we get $\tilde{\sigma}(\mathbf{x}, T_f - T)$, we back propagate it by solving another Liouville equation until $t = 0$. Figures 2 (c) and (d) show the contour plot of this *forward* FTLE. Figure 2 (c) shows the FTLE using an underlying Cartesian mesh of size $\Delta x = \Delta y = 1/128$. Similar to figures 2 (a) and (b), we have refined the computation mesh to $\Delta x = \Delta y = 1/512$ in figure 2 (d). Similar to the *backward* FTLE, we can recover the fine feature in $\sigma^{-T}(x, y, t)$ when we refine the computational mesh.

In figure 3, we compare our solutions with the Lagrangian solutions. Figure 3 (a), (c) and (e) show the solutions to the scaled *backward* FTLE at $t = T = 15$. Figure 3 (b), (d) and (f) plot the scaled *forward* FTLE at $t = 0$. All these solutions are computed on the same underlying mesh of 1025×513 so that $\Delta x = \Delta y = 1/512$. The first row (figures (a) and (b)) shows the solutions computed by the Lagrangian approach using the MATLAB function ode45 which automatically chooses an adaptive timestep in the program. As we have mentioned before, it is difficult to reconcile the accuracies at different mesh points. When we compute the resulting FTLE by finite differencing the map, error will amplify and this explains the tiny oscillations in the solutions. When applying a fourth order Runge-Kutta solver with a fixed timestep, these oscillations are removed and we have shown the results on the second row (figures (c) and (d)). The third row is the Eulerian solutions computed by the proposed method. The solutions are clean and the fine structures in the FTLE are well-captured. Note that those solutions in figure 2 are obtained by simply propagating the scaled FTLE obtained by $T = 5$ for a time difference $S = 10$. Even though the estimate (34) says the error in this scaled FTLE might grow at worst of $O(S)$, the error in the FTLE is actually of $O(S/(T + S))$ and is only of $O(10^{-1})$.

To confirm the computational complexities we discussed in Section 4, we have recorded the CPU times for the proposed Eulerian approach and also the Lagrangian formulation (with two different ways to choose Δt) using different Δx's. Table 1 shows these data measured in seconds. We have coded in MATLAB R2010a and these programs are executed in a laptop computer with a Intel Core 2 Duo 2.8GHz processor. Note that we are not claiming that the Eulerian approach always gives more accurate solutions. In fact, the Eulerian solutions from this table provide only approximations to the true FTLE as discussed earlier. Moreover, none of these codes has been optimized. In particular, the Lagrangian code was written with for-loops over all particles. One should be able to significantly improve these CPU times if the computational codes are properly vectorized/optimized. Therefore, one should use these CPU times to confirm the computational complexity only, but not to conclude that the Eulerian computation is faster than the Lagrangian computation. The second column in the table shows the computational times (in seconds) for using the proposed Eulerian approach with $T = 5$ and $S = 10$. We have computed the rate r in the computational complexity $O(\Delta x^{-r})$ using the CPU time measurements in the second column. These numerical values suggest that the computational complexity is roughly $O(\Delta x^{-3})$ which matched well with the estimate we have stated in Section 4. Because the solution from the code contains the FTLE on a mesh with $O(\Delta x^{-3})$ points in the $x - y - t$ space, each grid point is visited for only $O(1)$ time. For the Lagrangian formulation, we have computed the flow map using the fourth order Runge-Kutta method with a fixed timestep. The estimate in Section 4 gives $O(N^d MK)$ with N is the number of mesh points in each spatial direction, M is the number of time marching steps in the RK4 solver and K is the number of points we discretize the time domain of the FTLE solution. Even though we are showing only the FTLE at $t = 0$ or $t = 15$ in figure 3, similar to the Eulerian approach we compute the FTLE also at all time steps, i.e. on all $O(N^d K)$ mesh points. The measured CPU times are shown on the fourth and the sixth columns. On these two columns, we have used $N = O(\Delta x^{-1})$ and $K = O(\Delta x^{-1})$. On the fourth column where $\Delta t = 1$, i.e. $M = O(1)$, we obtain the numerical complexity of $O(N^d MK) = O(\Delta x^{-3})$ on the fifth column. For the sixth column, we have $M = O(\Delta x^{-1})$ and so $O(N^d MK) = O(\Delta x^{-4})$. All CPU times observed from the table also match well with the estimate $O(N^d MK)$.

Δx	Eulerian	Rate	Lagrangian with $\Delta t = 1$	Rate	Lagrangian with $\Delta t = \Delta x$	Rate
1/64	22.87	-	1085	-	4591	-
1/128	110.9	2.28	8424	2.95	71440	3.96
1/256	1021	3.20	66030	2.97	1123000	3.97
1/512	8859	3.11	525700	2.99	-	-

Table 1: (Example 6.1) CPU times (in seconds) for the Eulerian approach with $T = 5$ and $S = 10$, and the Lagrangian approach with $\Delta t = 1$ and $\Delta t = \Delta x$. The computational complexity for the proposed Eulerian approach is approximately $O(\Delta x^{-3})$, while the complexity for the Lagrangian approach is roughly $O(\Delta x^{-3}\Delta t^{-1})$. Note that both the Eulerian code and the Lagrangian code are not optimized. One should be able to significantly improve these CPU times if the computational codes are properly vectorized/optimized.

6.2. A simple analytic field

In this example, we consider a simple analytical velocity field [45] given by

$$u = x - y^2 \;\text{ and }\; v = -y + x^2\,. \tag{63}$$

The computational domain is $[-6,6]^2$ and we assume that we do not have any information on the velocity outside this domain. Unlike usual Lagrangian approach where the ray trajectories have to be artificially stopped at the boundary of the computational domain, we can naturally impose the fixed inflow boundary condition and the non-reflective outflow boundary condition.

We consider first solving the level set equations *forward* in time from $t = 0$ to $t = T = 1.5$. Then we calculate the FTLE $\tilde{\sigma}(\mathbf{x}, T)$ on the fixed Cartesian mesh points. Once we have obtained this quantity, we switch to another Liouville equation and propagate it up to $T_f = 5$, i.e. $S = 3.5$. The solution is plotted in figure 4 (a). Since we are advecting the level set functions *forward* in time, these figures are actually showing the *backward* FTLE.

Next we solve the level set equations backward in time from $t = T_f = 5$ up to $t = T_f - T = 3.5$ with the initial condition $\Psi^1(x, y, T_f) = x$ and $\Psi^2(x, y, T_f) = y$. Using these level set functions at $t = T_f - T = 3.5$, we compute the *forward* FTLE and then back propagate it until $t = 0$. The solutions at different time using 257 mesh points are shown in figure 4 (b). In figure 4 (c) and (d), we repeat the computations but using $T = 5$. As we can see, we can well-approximate the solutions by simply advecting the FTLE along Lagrangian trajectories. We have also shown the solutions using the usual Lagrangian approach in figure 4 (e) and (f). One can clearly observe the artificial LCS's due to the naive treatment of the particle trajectory by imposing $u = v = 0$ outside the computational domain. A velocity extension algorithm has been proposed in [45] to remove such LCS. We do not repeat the computations here but refer any interested reader to the cited reference. Comparing our solutions to the typical Lagrangian approach, we do not observe any interference from the boundary conditions. Both our computed *forward* and *backward* FTLE's are clean away from the true LCS.

6.3. Rayleigh-Bénard convection

In this section, we consider a three-dimension example for which the velocity field is given by [21]

$$\begin{aligned} u &= \frac{A}{k}\xi \sin kr \cos z\,, \\ v &= \frac{A}{k} y \sin kr \cos z\,, \\ w &= -A \sin z \left(r \cos kr + \frac{2}{k} \sin kr \right)\,, \end{aligned} \tag{64}$$

where $A = 0.24$, $k = 2$, $r^2 = \xi^2 + y^2$ and $\xi = x - g(t)$. Unlike [21] where the paper has chosen a random forcing $g(t)$, we simply use $g(t) = 0.5\cos(2\pi t)$. In figure 5 (a), we have shown different isosurfaces of the *backward* FTLE computed at $t = 50$ using $T = 12$ and $S = 38$, i.e. we have computed $(\Psi^1(x, y, z, t), \Psi^2(x, y, z, t), \Psi^3(x, y, z, t))$ for $(x, y, z) \in [-2, 2]^3$ and $t \in [0, T = 12]$ forward in time by solving their corresponding level set equations with the initial conditions

$$\begin{aligned} \Psi^1(x, y, z, 0) &= x \\ \Psi^2(x, y, z, 0) &= y \\ \Psi^3(x, y, z, 0) &= z\,. \end{aligned} \tag{65}$$

We have also plotted the isosurfaces of the *forward* FTLE $\sigma^T(\mathbf{x}, t)$ at $t = 0$, $T = 12$ and $S = 38$ using the same underlying Cartesian mesh in figure 5 (b).

To compare our solutions with the typical Lagrangian approach, we have shown both the *forward* and the *backward* FTLEs on the cross section $z = 1$ in figure 6. The Lagrangian solutions are computed on the same mesh with $\Delta x = \Delta y = \Delta z = 1/32$ using RK4 with a fixed time-marching step $\Delta t = 1/16$. Similar to the previous example, we stop any ray trajectory if it hits the boundary by imposing $u = v = w = 0$ outside the computational domain $[-2, 2]^3$. Our proposed Eulerian computations match with the Lagrangian solutions very well even for such a relatively coarse mesh.

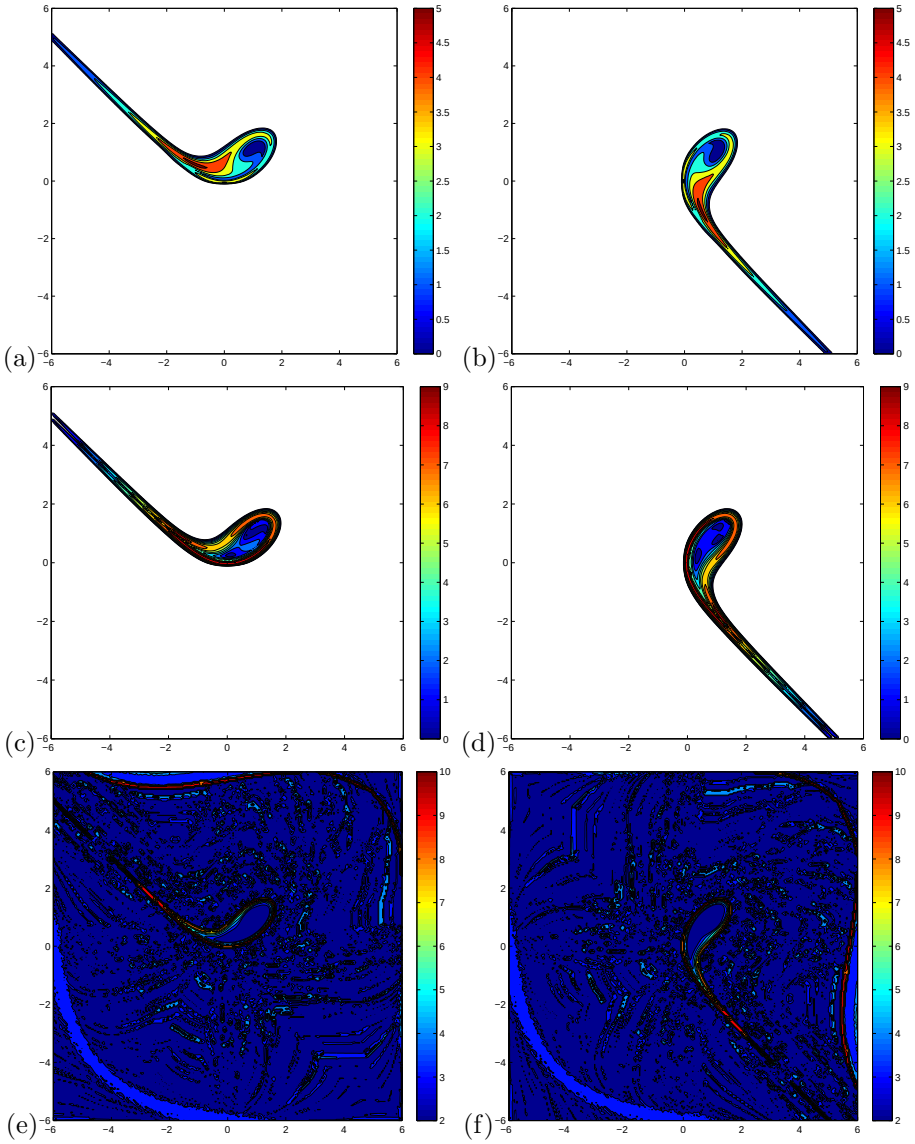

Figure 4: (Example 6.2) (a) The *backward* FTLE using the proposed Eulerian approach with $\Delta x = \Delta y = 6/256$, $t = 5.0$, $T = 1.5$ and $S = 3.5$. (b) The *forward* FTLE using the proposed Eulerian approach with $\Delta x = \Delta y = 6/256$, $t = 0.0$, $T = 1.5$ and $S = 3.5$. (c) The *backward* FTLE using the proposed Eulerian approach with $\Delta x = \Delta y = 6/256$, $t = 5.0$ and $T = 5.0$. (d) The *forward* FTLE using the proposed Eulerian approach with $\Delta x = \Delta y = 6/256$, $t = 0.0$ and $T = 5.0$. (e) The *backward* FTLE using the usual Lagrangian approach with $\Delta x = \Delta y = 6/256$, $t = 5.0$ and $T = 5$. (b) The *forward* FTLE using the usual Lagrangian approach with $\Delta x = \Delta y = 6/256$, $t = 0.0$ and $T = 5$.

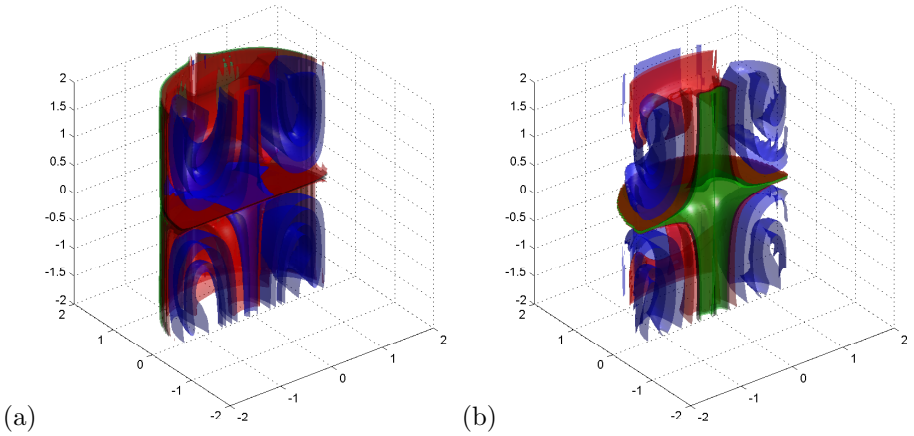

Figure 5: (Example 6.3) (a) The *backward* FTLE using $\Delta x = \Delta y = 1/32$ at $t = 50$, $T = 12$ and $S = 38$. (b) The *forward* FTLE using $\Delta x = \Delta y = 1/32$ at $t = 0$, $T = 12$ and $S = 38$.

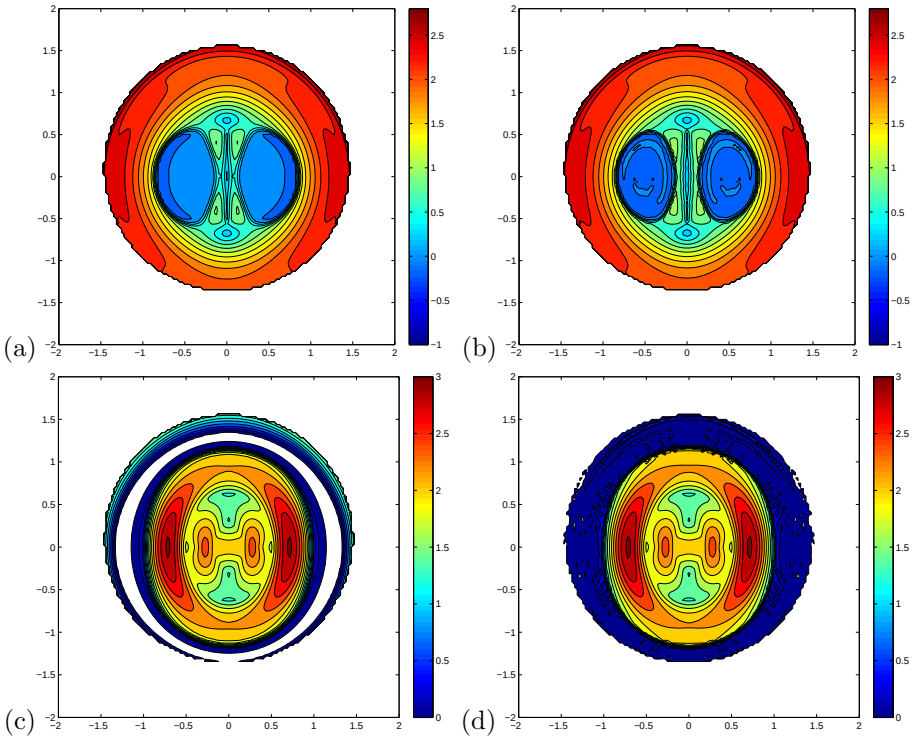

Figure 6: (Example 6.3) (a) The *backward* FTLE on the cross section $z = 1$ using the proposed Eulerian approach with $\Delta x = \Delta y = 1/32$, $t = 5$, $T = 5$. (b) The *backward* FTLE on the cross section $z = 1$ using the usual Lagrangian approach with $\Delta x = \Delta y = 1/32$, $t = 5$ and $T = 5$. (c) The *forward* FTLE on the cross section $z = 1$ using the proposed Eulerian approach with $\Delta x = \Delta y = 1/32$, $t = 40$, $T = 10$. (d) The *forward* FTLE on the cross section $z = 1$ using the usual Lagrangian approach with $\Delta x = \Delta y = 1/32$, $t = 40$, $T = 10$.

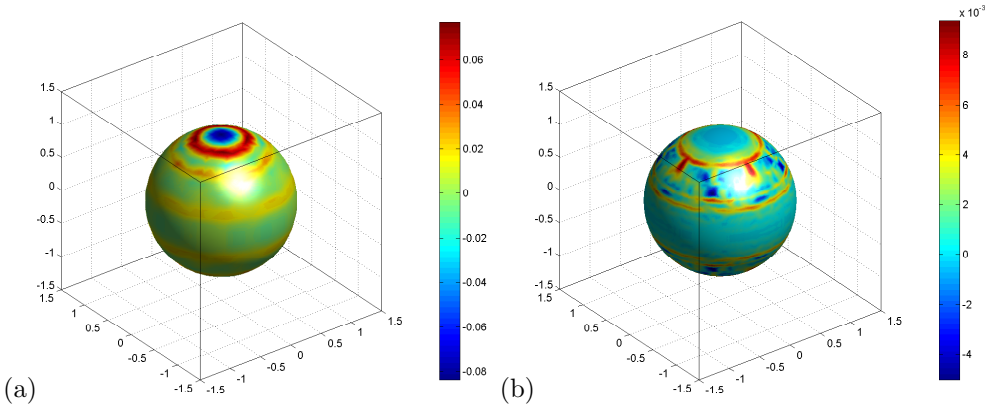

Figure 7: (Example 6.4) The *forward* FTLE on a sphere under the rigid body rotation. The exact solution is zero everywhere on the sphere. The underlying mesh size is (a) $\Delta x = 3/32$ and (b) $\Delta x = 3/128$.

6.4. FTLE on a sphere

In this example, we consider several simple three dimensional cases of a sphere to check the convergence and the accuracy of the proposed Eulerian approach. The first example is a rotating sphere under the rigid motion given by $u(x, y) = y$, $v(x, y) = -x$ and $w(x, y) = 0$. Figure 7 shows the FTLE on the sphere at $t = \pi$ using mesh of size $\Delta x = \Delta y = \Delta z = 3/32$ and $3/128$, respectively. Since the rotation is rigid, the Jacobian is identity and therefore the FTLE is zero everywhere on the manifold. There are various sources of error in the algorithm. One comes from the implicit representation of the interface. Since MATLAB extracts the zero level set using linear interpolations, the location of the interface may be off by an error of order $O(\Delta x)$. Numerical solutions to the level set equations or the Liouville equations also contribute error to the FTLE. However, as we can see from these solutions, the error does decrease as we refine the underlying mesh. Few errors from grid effects might still be observable, but these errors are of order of 10^{-3} using only $\Delta x = 3/128$.

In the next example we consider the expansion of a sphere in the normal direction. The initial radius of the sphere is $r_0 = 0.75$. The velocity of the sphere is chosen to be $v_n = 1$ and we are interested in the solution of the *backward* FTLE at $t = T = 0.5$. The exact solution can be found in this simple case and it is given by

$$\tilde{\sigma}^{-T} = \ln\left(\frac{r_0}{r_0 + T}\right)^2 = 2\ln\left(\frac{3}{5}\right) \simeq -1.021651248\,. \tag{66}$$

In figure 8 we showed the errors in the *backard* FTLE on an expanding sphere at $t = T = 0.5$. Similar to previous calculations, the error in the solution drops as we refined the underlying mesh. We have also computed the *forward* FTLE at $t = 0$ with $T = 0.5$

$$\tilde{\sigma}^{T} = \ln\left(\frac{r_0 + T}{r_0}\right)^2 = 2\ln\left(\frac{5}{3}\right) \simeq 1.021651248\,. \tag{67}$$

The solutions using a coarse and a fine computation mesh are plotted in figure 8 (c) and (d). Note, however, that the errors in these solutions are significant larger that in the *backward* FTLE. The main reason is that the *forward* FTLE computations require an extra step to solve the level set equation from $t = 0$ to $t = T$, i.e. step 3 in Algorithm 3. This introduces extra perturbations in extracting the FTLE on the final manifold.

In this last example from this section, we consider an advection motion in the field of four point vortices on a sphere [30, 31]. The velocity of any marker particle satisfies the motion

$$\mathbf{x}' = \frac{1}{2\pi}\sum_{i=1}^{4}\frac{\mathbf{x}_i \times \mathbf{x}}{2(1 - \mathbf{x}\cdot\mathbf{x}_i)}\,, \tag{68}$$

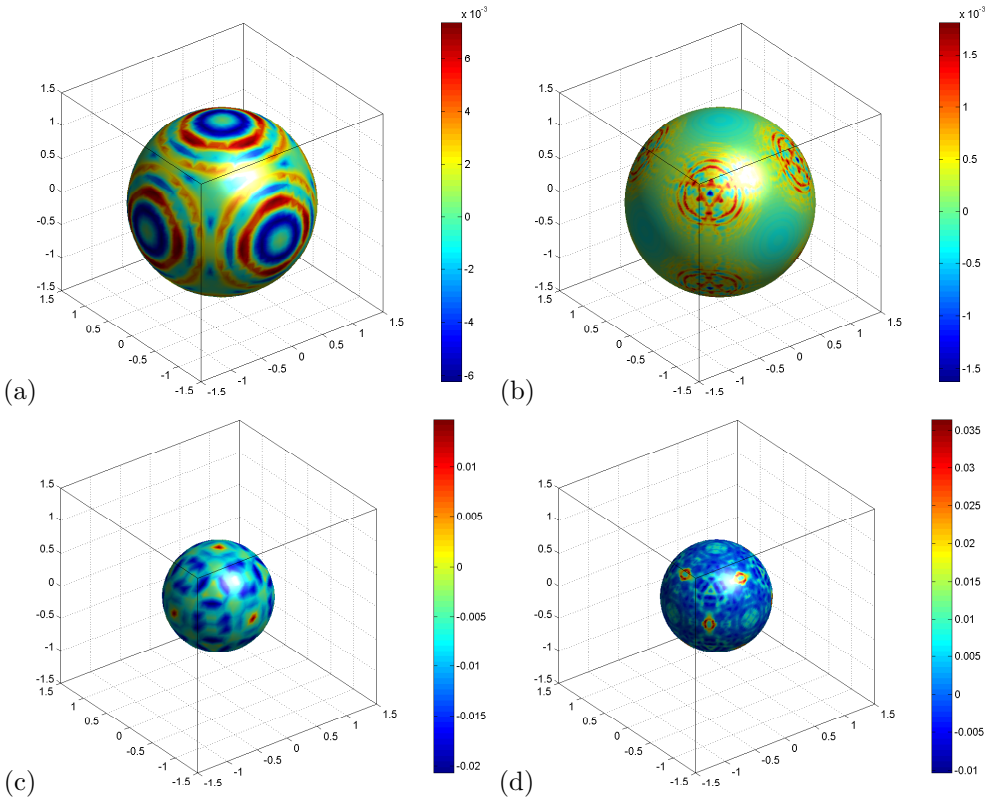

Figure 8: (Example 6.4) The error in the *backward* FTLE on a sphere under the motion in the normal direction. The underlying mesh size is (a) $\Delta x = 3/32$ and (b) $\Delta x = 3/128$. The error in the *forward* FTLE on a sphere under the motion in the normal direction. The underlying mesh size is (c) $\Delta x = 3/32$ and (d) $\Delta x = 3/128$.

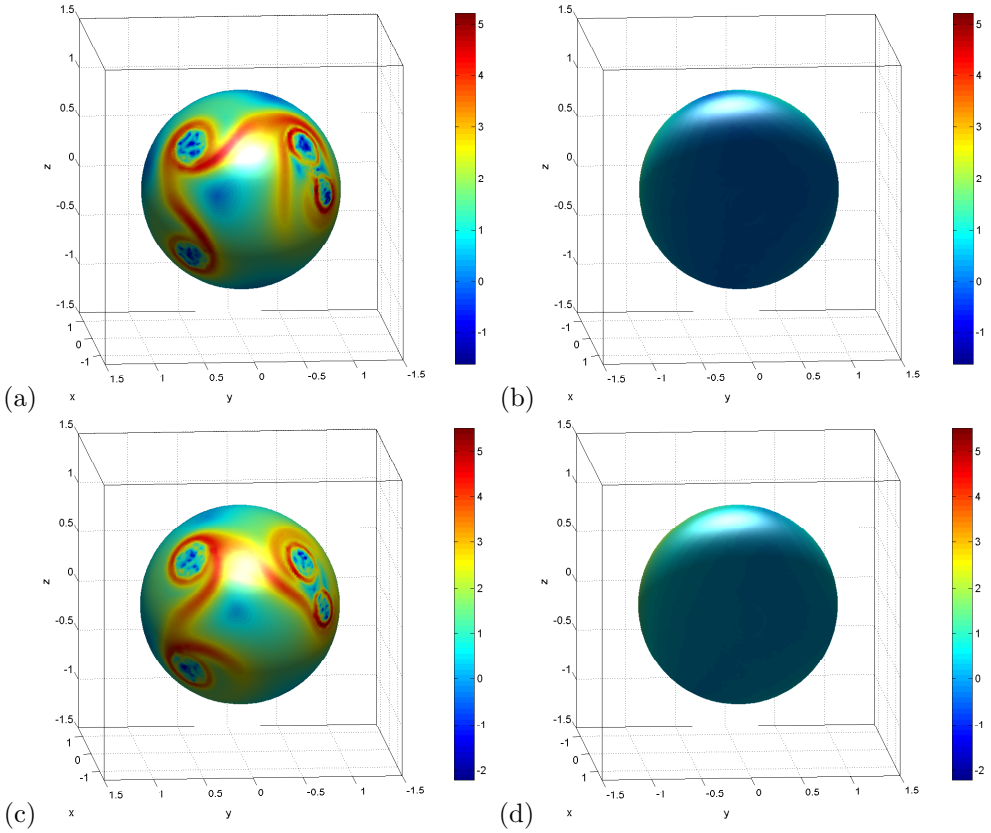

Figure 9: (Example 6.4) (a) and (b) The *backward* FTLE on a sphere under 4 point vortices at $t = 2$ with $T = 2$. (c) and (d) The corresponding *forward* FTLE at $t = 0.0$ with $T = 2$.

with the point vortices centered at $(1/\sqrt{3}, -1/\sqrt{3}, 1/\sqrt{3})$, $(1/\sqrt{3}, -1/\sqrt{3}, -1/\sqrt{3})$, $(-\sqrt{2/3}, 1/\sqrt{3}, 0)$ and $(-1/\sqrt{3}, -1/\sqrt{3}, 1/\sqrt{3})$. In figure 9, we have shown both the *backward* and the *forward* FTLE on the sphere of radius $r_0 = 1$ from two different angles. The mesh size is $\Delta x = \Delta y = \Delta z = 3/128$.

6.5. FTLE on an evolving manifold

In this last section, we approximate the FTLE on an evolving manifold. In this first example, we consider an important flow in designing numerical methods for interface evolutions where an initial sphere is moving along with a vortex flow. In the second example, we propose to apply the techniques we have developed to study the so-called paraxial assumption which is widely used in the seismic community.

6.5.1. Sphere under vortex flow

In the first example, we consider the motion of a sphere under the vortex motion [6] governed by

$$\begin{aligned} u &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z) \\ v &= -\sin^2(\pi y)\sin(2\pi x)\sin(2\pi z) \\ w &= -\sin^2(\pi z)\sin(2\pi y)\sin(2\pi x)\,. \end{aligned} \tag{69}$$

This challenging flow is widely used as a test case in various numerical methods for interface evolutions. It was originally proposed in [6] to test if a numerical method is able to resolve very thin structures. The initial sphere is centered at $(0.35, 0.35, 0.35)$ with radius 0.15. Figure 10 shows both the *backward* and the *forward* FTLE using $T = 0.5$ from two different angles. In figure 10 (a) and (b) we plot the *backward* FTLE

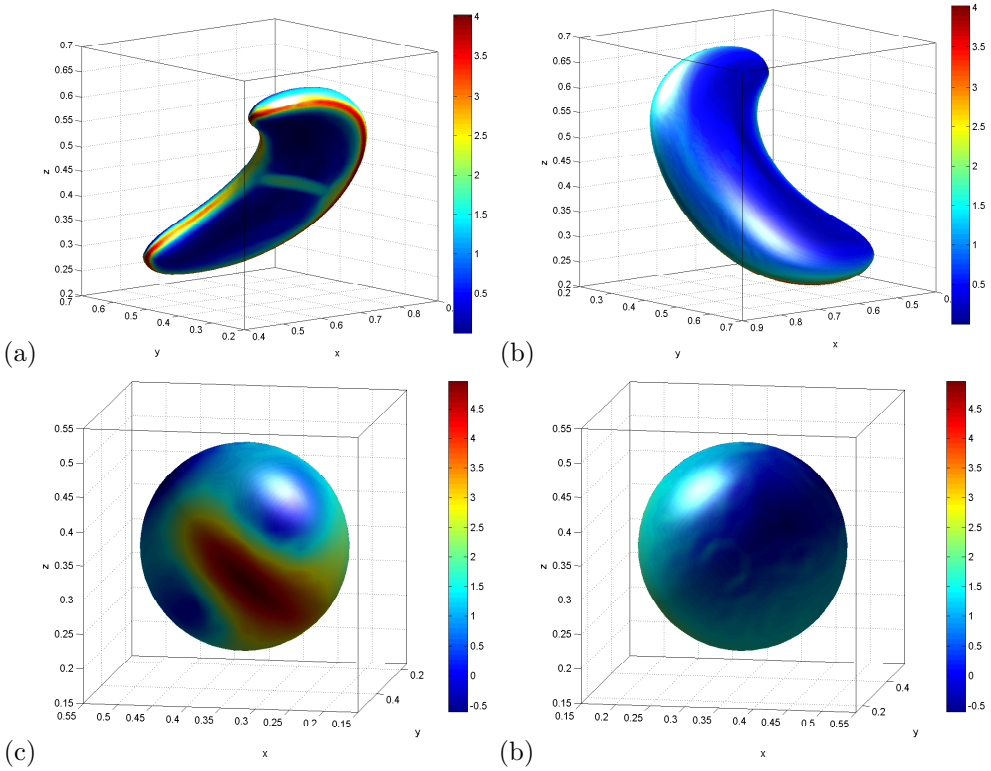

Figure 10: (Example 6.5) (a) and (b) The *backward* FTLE on an evolving manifold under the vortex motion at $t = 0.5$ with $T = 0.5$. (c) and (d) The corresponding *forward* FTLE at $t = 0.0$ with $T = 0.5$.

on the evolved manifold at $t = 0.5$. The *forward* FTLE at $t = 0.0$ is shown in (c) and (d). The special structures of the FTLE defined on the evolving interface are nicely captured. Note that no parametrization of the evolving surface is needed in this computation. The interface is implicitly represented by the level set function defined on the fixed Cartesian mesh.

6.5.2. Application to paraxial geometrical optics

Paraxial assumption is widely used in the seismic community [9, 38, 44, 36, 26, 22, 23]. In this example, we will apply the technique we have developed above to study the validity of such assumption. Some preliminary results will be shown at the end of the section. We consider the linear acoustic wave equation. According to the Debye procedure, we first insert the high frequency asymptotic ansatz into the wave equation. Among all the terms in the asymptotic ansatz, the most important term is the zeroth order term, the so-called geometrical optics term [17]. This geometrical optics term consists of two functions, a phase function satisfying the eikonal equation which is a first-order nonlinear PDE, and an amplitude function which solves a linear transport equation with the phase gradient as coefficients, i.e.

$$|\nabla\tau| = \frac{1}{c}\,, \quad \nabla\cdot(A^2\nabla\tau) = 0\,. \tag{70}$$

Thus to construct the geometrical optics term, we first solve the eikonal equation for the phase function and then integrate the transport equation afterwards for amplitude. We consider a point source condition for the eikonal equation defined in an open, bounded domain

$$\Omega = \{(x,z) : x \in [x_{\min}, x_{\max}]\,,\ z \in [0, z_{\max}]\}\,. \tag{71}$$

In many applications, for example, wave propagation in seismics application, the traveltime T is carried by the so-called subhorizontal rays [9, 43], where subhorizontal means oriented in the positive z-direction. By the method of characteristics, we use z as a time-like variable so that we derive the following reduced system [36, 26, 37].

$$\begin{aligned} \frac{dx}{dz} &= \tan\theta \\ \frac{d\theta}{dz} &= \frac{1}{c}(c_z\tan\theta - c_x)\,, \end{aligned} \tag{72}$$

with the initial condition $x(z=0) = x_s$ and $\theta(z=0) = \theta_s$ where $\theta_s \in [\theta_{\min}, \theta_{\max}] \subset (-\pi/2, \pi/2)$.

Obviously it might not be always valid to make such paraxial approximation in the computational domain. A simple condition is that all emitted rays from the point source travel in one single direction which is assumed to be the positive z-direction. Mathematically, this condition requires

$$|\theta(z)| < \pi/2\,, \tag{73}$$

where $\theta(z)$ is the angle made by the ray and the positive z-direction on a given z-level.

In the typical ray tracing method, one first approximates this *open* interval $(-\pi/2, \pi/2)$ by $[\theta_{\min}, \theta_{\max}]$ for some $\theta_{\min} > -\pi/2$ and $\theta_{\max} < \pi/2$. Rays are then emitted from the point source location with initial take-off angles chosen uniformly from this closed interval. One disadvantage of this Lagrangian formulation concerns the resolution of the solution. Since these rays will travel in the phase space according to the ray system, such Lagrangian formulation does not guarantee uniform resolution of the traveltime. In [36, 26, 37], we propose an Eulerian approach based on the level set method. Since we care only about those rays emitting from a given single point source located at $x = x_s$, we can implicitly represent all these rays by the zero level set of a level set function $\phi(x, \theta; z)$ in the phase space. Following [36, 26, 37], we solve

$$\phi_z + \frac{dx}{dz}\phi_x + \frac{d\theta}{dz}\phi_\theta = 0 \tag{74}$$

and extract the corresponding zero level set $\phi^{-1}(0)$ to obtain *all* arrival rays on the given level z with an corresponding takeoff angle $\theta(z=0) \in [\theta_{\min}, \theta_{\max}]$. For more details, we refer any interested readers to [36, 26, 37] for a complete derivation.

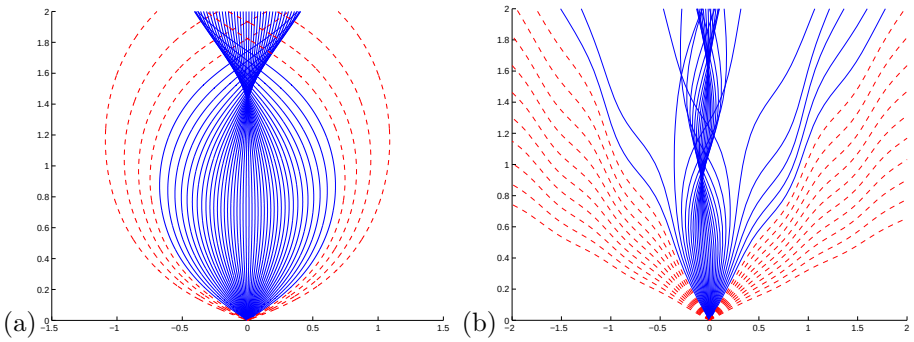

Figure 11: Ray trajectories of the velocity models (a) $c_w(x,z)$. and (b) $c_s(x,z)$. In (a), we cannot approximate the θ interval by $[\theta_{\min}, \theta_{\max}]$ since there are always rays missing (red dashed line).

Even with uniform resolution in the arrivaltime, paraxial approximation might not always give good approximations to the overall wavefield after we have obtained the solution of the amplitude function along these characteristics. The main reason is that the approximation in the domain for the initial takeoff angle might lead to incomplete arrival rays. We consider the following two simple velocity models.

$$\begin{aligned} c_w(x,z) &= 3 - 2.5\exp\left(-\frac{x^2}{2}\right) \\ c_s(x,z) &= 1 + 0.2\sin\left(\frac{\pi z}{2}\right)\sin[3\pi(x+0.55)]\,. \end{aligned} \tag{75}$$

Their corresponding ray trajectories are shown in figure 11. We plot in solid blue lines for those rays shooting from the point source at $x_s = 0$ with take-off angle $\theta_{\min} < \theta_0 < \theta_{\max}$ for some fixed $\theta_{\min}$ and $\theta_{\max}$. We also trace several rays from the same point source with take-off angles outside this interval. These rays are shown using red dashed lines in figure 11.

We first consider the waveguide example $c(x,z) = c_w(x,z)$ in figure 11 (a). Rays with initial takeoff angle outside the domain $[\theta_{\min}, \theta_{\max}]$ will turn back and reach the region near $x = 0$. Ignoring these rays by truncating $(-\pi/2, \pi/2)$ using $[\theta_{\min}, \theta_{\max}]$, we will not have enough information to construct the overall wavefield at a larger z since these missing rays will interact with other rays to get an interference pattern. In the second case as shown in figure 11 (b), on the other hand, these red-dashed rays have a different dynamic. In this particular example, these rays will leave the computational domain and will not affect the wavefield in the region $|x| < 1$ on the level $z = 2$.

Based on this Eulerian formulation, in this section we suggest a way to detect if the truncation in the θ-space provides a good approximation to the resulting wavefield using the FTLE. We follow the technique in algorithm 3 to compute the *forward* FTLE. Since the existence of LCS implies that trajectories on the evolving manifold will travel as patches and the separation will grow in time (the direction z in this application).

We show some preliminary results of the *forward* FTLE on an evolving manifold in figures 12 and 13. We plot the contours of the FTLE with $T = z_f - z$ in a small neighborhood of the zero level set of ϕ. In particular, figure 12 (c) and figure 13 (c) show the *forward* FTLE on the depth $z = 0$ with $z_f = 2$ and $T = 2$. As seen in figure 12 (c), there is no obvious LCS observed. This implies that all emitted rays from the single point source are traveling roughly as a single patch. It is therefore not clear how one can determine the $\theta_{\max}$ and the $\theta_{\min}$ in the paraxial approximation. Figure 13 (c) shows the computed FTLE in the sinusoidal model. The structure is much clear and this confirms us the computational domain in the θ-direction is big enough to capture necessary arrival rays for later wavefield construction.

A future work is to extend this approach to carefully study the paraxial assumption in [26] for computing paraxial geometrical optics approximation to the 3D wave equation. In that case, emitted rays from the single point source will be represented by a co-dimensional two surface in the $x - y - \theta - \phi$ phase space.

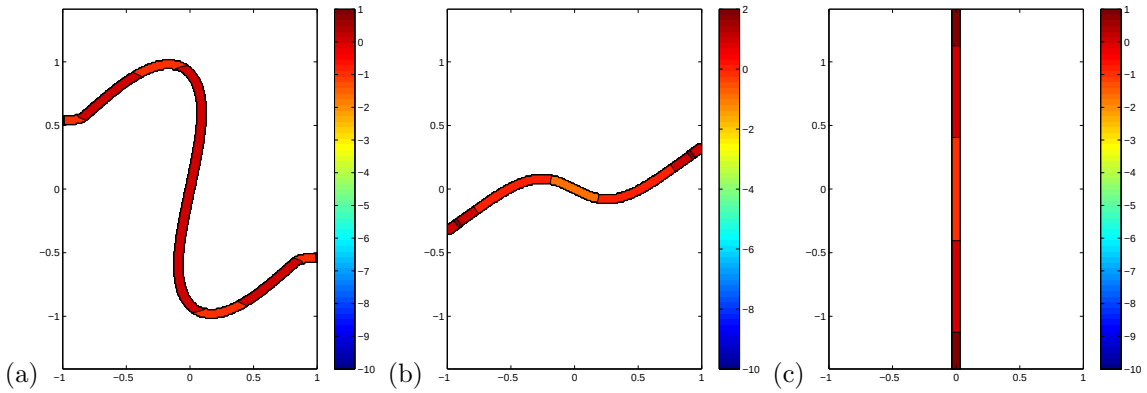

Figure 12: (Example 6.5) The *forward* FTLE of the waveguide model c_w using $\Delta x = \Delta\theta = 1/256$ with $z_f = 2$ at (a) $z = 1.8$, (b) $z = 0.8$, and (c) $z = 0.0$. No LCS is observed on the evolving manifold. This implies rays from the point source will travel as a single patch and it is not straight-forward how to obtain the truncated computational domain $[\theta_{\min}, \theta_{\max}]$ in the paraxial computation.

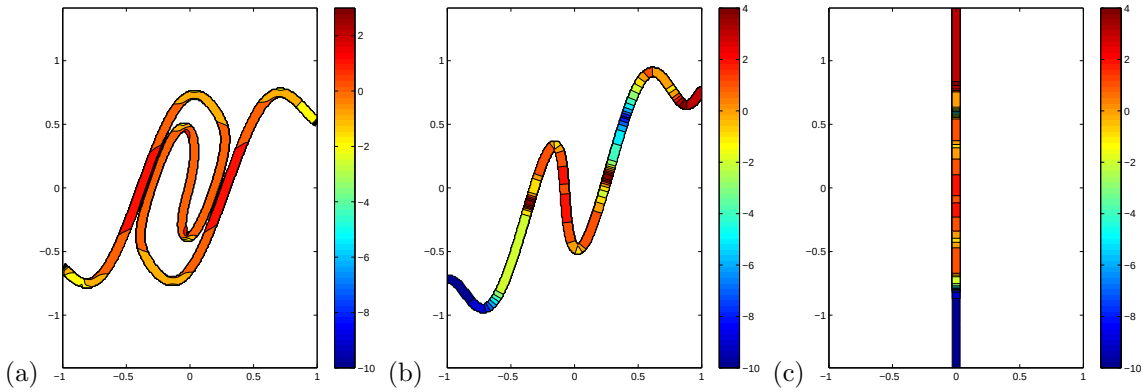

Figure 13: (Example 6.5) The *forward* FTLE of the Sinusoidal model c_s using $\Delta x = \Delta\theta = 1/256$ with $z_f = 2$ at (a) $z = 1.8$, (b) $z = 0.8$ and (c) $z = 0.0$. LCS is clearly observed which implies that rays from the points may travel in very different directions with a small perturbation in the initial take-off angle.

7. Conclusion and future work

In this paper, we propose an Eulerian framework for approximating the FTLE based on the level set formulation. The related flow map is computed by solving Liouville equations on a fixed mesh. Four algorithms are proposed to compute the FTLE under various situations. Algorithm 1 proposes an Eulerian approach for computing the *backward* FTLE by determining the corresponding flow map using the Liouville equation. This algorithm determines the FTLE on a single given time $t = T$. To efficiently obtain a wider range of FTLE, we propose in algorithm 2 an approximation of the FTLE by simply propagating the scaled FTLE obtained at $t = T$ using algorithm 1. Note however that this algorithm only approximates the true FTLE with an error of $O(S/T)$ or $O(S/(S + T))$ with S is the time difference from $t = T$. One should be careful when interpreting this output.

The second part of the paper computes the FTLE on evolving manifolds. A pure Eulerian approach is proposed in algorithm 3 which embeds the moving surface using a level set function. All computations are done on a fixed uniform Cartesian mesh. The method requires no local coordinate system or any local parametrization of the interface. A corresponding Lagrangian version is suggested in algorithm 4.

In the future, we propose to combine the current approaches with a CFD solver or to incorporate our algorithms with a flow field from real life measurements. We will also generalize the example in Section 6.5.2 to carefully study the paraxial assumption in computing the high frequency asymptotic solution to wave propagation in high dimensions.

Acknowledgements

The author would like to thank the referees for many useful suggestions on improving the paper and would also like to acknowledge the hospitality of the University of California, Irvine where preliminary work was performed. This work is partially supported by the Hong Kong RGC under Grant GRF602210.

References

[1] M. Bertalmio, L.-T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174:759–780, 2001.

[2] J. Brandman. A level-set method for computing the eigenvalues of elliptic operators defined on compact hypersurfaces. *J. Sci. Comput.*, 37:282–315, 2008.

[3] S.L. Brunton and C.W. Rowley. Fast computation of finite-time Lyapunov exponent fields for unsteady flows. *Chaos*, 20:017503, 2010.

[4] E.J. Candès and L. Ying. Fast geodesics computation with the phase flow method. *J. Comput. Phys.*, 220:6–18, 2006.

[5] B.M. Cardwell and K. Mohseni. Vortex shedding over two-dimensional airfoil: Where do the particles come from? *AIAA J.*, 46:545–547, 2008.

[6] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183:83–116, 2002.

[7] C. Garth, F. Gerhardt, X. Tricoche, and H. Hagen. Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics*, 13:1464–1471, 2007.

[8] C. Garth, G.S. Li, X. Tricoche, C.D. Hansen, and H. Hagen. *Visualization of Coherent Structures in Transient 2D flows.* Springer, 2009.

[9] S. Gray and W. May. Kirchhoff migration using eikonal equation traveltimes. *Geophysics*, 59:810–817, 1994.

[10] M.A. Green, C.W. Rowley, and A.J. Smiths. Using hyperbolic Lagrangian coherent structures to investugate vortices in biospired fluid flows. *Chaos*, 20:017510, 2010.

[11] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D*, 149:248–277, 2001.

[12] G. Haller. Lagrangian structures and the rate of strain in a partition of two-dimensional turbulence. *Phys. Fluids A*, 13:3368–3385, 2001.

[13] G. Haller. Lagrangian coherent structures from approximate velocity data. *Physics of Fluid*, 14:1851–1861, 2002.

[14] G. Haller. A variational theory of hyperbolic Lagrangian Coherent Structure. *Physica D*, 240:574–598, 2011.

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

[16] G. S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21:2126–2143, 2000.

[17] J. B. Keller and R. M. Lewis. Asymptotic methods for partial differential equations: the reduced wave equation and Maxwell's equations. *Surveys in Applied Mathematics*, 1:1–82, 1995.

[18] F. Lekien and N. Leonard. Dynamically consistent Lagrangian coherent structures. *Experimental Chaos: 8-th Experimental Chaos Conference*, pages 132–139, 2004.
[19] F. Lekien and J.E. Marsden. Tricubic interpolation in three dimensions. *Internat. J. Numer. Methods Engrg.*, 63:455–471, 2005.
[20] F. Lekien and S.D. Ross. The computation of finite-time Lyapunov exponents on unstructured meshes and for non-euclidean manifolds. *Chaos*, 20:017505, 2010.
[21] F. Lekien, S.C. Shadden, and J.E. Marsden. Lagrangian coherent structures in n-dimensional systems. *Journal of Mathematical Physics*, 48:065404, 2007.
[22] S. Leung and J. Qian. A transmission tomography problem based on multiple arrivals from paraxial liouville equations. *Expanded Abstract for the SEG 75th Annual Meeting*, 2005.
[23] S. Leung and J. Qian. Transmission traveltime tomography based on paraxial liouville equations and level set formulations. *Inverse Problems*, 23:799–821, 2007.
[24] S. Leung and J. Qian. Eulerian Gaussian beams for Schrödinger equations in the semi-classical regime. *J. Comput. Phys.*, 228:2951–2977, 2009.
[25] S. Leung, J. Qian, and R. Burridge. Eulerian Gaussian beams for high frequency wave propagation. *Geophysics*, 72:SM61–SM76, 2007.
[26] S. Leung, J. Qian, and S. Osher. A level set method for three-dimensional paraxial geometrical optics with multiple sources. *Commun. Math. Sci.*, 2(4):643–672, 2004.
[27] D. Lipinski and K. Mohseni. Flow structures and fluid transport for the hydromedusae Sarsia tubulosa and Aequorea victoria. *J. Exp. Biology*, 212:2436–2447, 2009.
[28] S. Lukens, X. Yang, and L. Fauci. Using Lagrangian coherent structures to analyze fluid mixing by cillia. *Chaos*, 20:017511, 2010.
[29] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *J. Comput. Phys.*, 225:300–321, 2007.
[30] K. Newton. *The N-Vortex problem: Analytical Techniques.* Springer, New York, 2001.
[31] P.K. Newton and S.D. Ross. Chaotic advection in the restricted four-vortex problem on a sphere. *Physica D*, 223:36–53, 2006.
[32] S. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer-Verlag, New York, 2003.
[33] S. J. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.
[34] S. J. Osher and C. W. Shu. High-order Essentially NonOscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Num. Anal.*, 28:907–922, 1991.
[35] D. Peng, B. Merriman, S. Osher, H. K. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155:410–438, 1999.
[36] J. Qian and S. Leung. A level set based Eulerian method for paraxial multivalued traveltimes. *J. Comput. Phys.*, 197:711–736, 2004.
[37] J. Qian and S. Leung. A local level set method for paraxial multivalued geometric optics. *SIAM J. Sci. Comput.*, 28:206–223, 2006.
[38] J. Qian and W. W. Symes. Paraxial eikonal solvers for anisotropic quasi-P traveltimes. *J. Comput. Phys.*, 173:1–23, 2001.
[39] F. Sadlo and R. Peikert. Efficient visualization of Lagrangian coherent structures by filtered amr ridge extraction. *IEEE Transactions on Visualization and Computer Graphics*, 13:1456–1463, 2007.
[40] T. Sapsis and G. Haller. Inertial particle dynamics in a hurricane. *Journal of the Atmospheric Sciences*, 66:2481–2492, 2009.
[41] J. A. Sethian. *Level set methods.* Cambridge Univ. Press, second edition, 1999.
[42] S.C. Shadden, F. Lekien, and J.E. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D*, 212:271–304, 2005.
[43] W. W. Symes. Mathematics of reflection seismology. In *Annual Report, The Rice Inversion Project, (http://www.trip.caam.rice.edu/)*. Rice University, 1995.
[44] W. W. Symes and J. Qian. A slowness matching Eulerian method for multivalued solutions of eikonal equations. *J. Sci. Comp.*, 19:501–526, 2003.
[45] W. Tang, P.W. Chan, and G. Haller. Accurate extraction of Lagrangian coherent structures over finite domains with application to flight data analysis over Hong Kong international airport. *Chaos*, 20:017502, 2010.
[46] W. Tang and T. Peacock. Lagrangian coherent structures and internal wave attractors. *Chaos*, 20:017508, 2010.