Computing the Finite Time Lyapunov Exponent for Flows with Uncertainties

Guoqiao You * Shingyu Leung†

Abstract

We propose an Eulerian approach to compute the *expected* finite time Lyapunov exponent (FTLE) of uncertain flow fields. The definition extends the usual FTLE for deterministic dynamical systems. Instead of performing Monte Carlo simulations as in typical Lagrangian computations, our approach associates each initial flow particle with a probability density function (PDF) which satisfies an advection-diffusion equation known as the Fokker-Planck (FP) equation. Numerically, we incorporate Strang's splitting scheme so that we can obtain a second-order accurate solution to the equation. To further improve the computational efficiency, we develop an adaptive approach to concentrate the computation of the FTLE near the ridge, where the so-called Lagrangian coherent structure (LCS) might exist. We will apply our proposed algorithm to several test examples including a real-life dataset to demonstrate the performance of the method.

1 Introduction

Mathematical tools are needed to analyze, visualize, and then extract important information in complex fluid flows and other time-varying dynamical systems. One interesting approach is to partition the space-time domain into subregions based on certain quantity measured along with the passive tracer advected according to the associated dynamical system. Because of such a Lagrangian property in the definition of these quantities, the corresponding partition is named the Lagrangian coherent structure (LCS). The most commonly used way to extract the LCS is based on the so-called finite time Lyapunov exponent (FTLE) [17, 13, 14, 40, 24]. This quantity measures the rate of change in the distance between neighboring particles across a finite interval of time with an infinitesimal perturbation in the initial position. To obtain the FTLE field, one needs to first compute the flow map which links the initial location of a particle with the arrival position based on the characteristic line. Mathematically these particles in the extended phase space satisfy the ordinary differential equation (ODE) given by

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

with the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$ and a Lipschitz velocity field $\mathbf{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$. The flow map $\Phi_{t_0}^{t_0+T} : \mathbb{R}^d \to \mathbb{R}^d$ is defined as the mapping which takes the point $\mathbf{x}_0$ to the particle location at the final time $t = t_0 + T$, i.e $\Phi_{t_0}^{t_0+T}(\mathbf{x}_0) = \mathbf{x}(t_0 + T)$ with $\mathbf{x}(t)$ satisfying (1). The FTLE is then defined using the largest eigenvalue of the deformation matrix based on the Jacobian of this resulting flow map. Following the definition of Haller [13, 15, 16], one can see that the LCS is closely related to the *ridges* of the FTLE fields. In a series of recent studies [25, 26, 44, 47, 46, 45, 27, 32], we have developed various Eulerian approaches to numerically compute the FTLE on a fixed Cartesian mesh. The idea is to combine the approach with

*School of Statistics and Mathematics, Nanjing Audit University, Nanjing 211815, China. Email: **270217@nau.edu.cn**

†Department of Mathematics, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Email: **masyleung@ust.hk**

the level set method [34, 33] which allows the flow map to satisfy a Liouville equation. Such hyperbolic partial differential equations (PDEs) can then be solved by any well-established robust and high order accurate numerical methods.

Yet all the tools and approaches mentioned above require a noise-free velocity field that, unfortunately, can hardly be satisfied in practice. Instead, noise or uncertainty is common in measurements from any real applications such as computational fluid dynamics, weather research, and aerodynamics. As a result, it is necessary to develop numerical approaches that can visualize and analyze uncertain unsteady flow fields. There are currently several main approaches for understanding these flows. The first class are direct methods. A vector glyph approach has been developed to visualize uncertainty in winds and ocean currents and also environmental vector field data in [42]. They consider uncertainties in various aspects such as the direction, the magnitude, the mean direction and the length of the flow field. [19] has also introduced a new type of glyph to visualize unsteady flows with static images, in order to discriminate regions of different flow behaviors and regions of different uncertain variations in space and time. Texture-based techniques [2] are also presented to visualize 2-dimensional uncertain unsteady flow fields. In particular, they use semi-Lagrangian texture advection to show flow directions by streaklines and convey uncertainty by blurring these streaklines. All these direct methods try to encode uncertainties as additional visual channels such as glyphs or textures, which makes them not feasible for large complex datasets. Another class of uncertain flow visualization approaches are feature-based ones. They try to extract core features from uncertain flow fields. For example, vortex detectors such as λ_2 and Q-Criterion are applied to uncertain flow fields [36] and the detected vortices are represented as a probabilistic field. [37] has extended methods for extracting local features in crisp vector fields to uncertain fields. They also define probabilistic counterparts for critical points and vortex cores.

In this paper, we propose an Eulerian approach to compute the *expected* FTLE of uncertain flow fields, which gives a statistical overview of how transport behaviors differ in neighboring particle locations. Some recent literature has also studied the FTLE in uncertain velocity fields [39, 11]. They have extended the original definition of the flow map Φ to a *stochastic* flow map based on the Monte Carlo (MC) simulations by first solving the corresponding stochastic differential equation multiple times. One approach is to define the FTLE directly based on the maximal variance direction of the arrival location from all adjacent grid points, which can be done using principal component analysis (PCA) [39]. A more refined way is to compute the expectation of the gradient of the stochastic flow map from those multiple particle trajectories [11, 12]. Particles are advected stochastically with a (probably very large) number of MC simulation runs. The number of runs is iteratively determined. In each iteration, a number of runs are conducted and the iteration stops if the output does not statistically significantly change the solution anymore. This can be numerically extremely expensive. All these methods do not focus on the particular FTLE computation method itself, but on deriving the uncertainty from particle tracing and the FTLE results.

In our proposed algorithm, we first assign each particle starting from a mesh point a probability density function (PDF) to denote the distribution of its arrival locations and then show that these PDFs satisfy the usual Fokker-Planck (FP) equation with different initial conditions (delta functions). The FP equation can be efficiently solved using operator splitting methods [9]. These methods are simple yet powerful numerical algorithms to obtain the solution to complex PDEs by decomposing them into several simple subproblems so that the solution to each subproblem can be easily found. This technique has been widely used in various fields of applied sciences and engineering including Navier-Stokes equations [4, 29, 30], digital image processing [10], high frequency wave propagation [6] and related inverse problem [5], the homogenization problem of the Hamilton-Jacobi operator [7], numerical solutions to the Monge–Ampère equation [8, 28] and many more. In this work, we will apply Strang's splitting method so that we can design an implicit scheme to improve the computational efficiency and also can efficiently

reuse part of the intermediate information in the computations. After obtaining the PDF corresponding to each mesh point, we propose an *expected* FTLE which actually measures the differences in the advected PDFs. An advantage of our approach is that we only need discrete velocity data on mesh points since we are solving PDEs. Another advantage is that we do not need to care much about or put extra effort into the uncertainty in the flow by, for example, performing a large number of MC simulations.

The FP equation itself is not new to the computational dynamical system community. For example, coherent sets are approximated by singular vectors of the transfer operator computed by the solution of the FP equation [3]. By clustering the first n singular vectors, the method partitions the domain into n coherent subsets. Also, [20] has introduced a geometric heat-flow theory which views the LCS as a metastable/almost-invariant set under the Lagrangian FP equation. In both works, however, the underlying velocity field is assumed to be noise-free. The FP equation has been introduced only to understand the behavior of the given Lipschitz drift velocity field.

The rest of the paper is organized as follows. In Section 2, we first review the Eulerian approaches for constructing the backward flow map and then present the Itô stochastic differential equation (SDE) governing the uncertain unsteady flows for further development. Our approach is outlined in Section 3. In particular, we first derive the FP equations whose solutions are exactly the PDFs of the arrival locations of the initially uniformly sampled passive tracers. After listing the computational challenges in solving the FP equations, we introduce an effective algorithm based on Strang's operator splitting scheme. Having obtained these PDFs, we give the formula to compute the *expected* flow map starting from each mesh point and hence the corresponding *expected* FTLE field. To further improve the computational efficiency of the FTLE visualization, we propose an adaptive approach in Section 4. Finally, in Section 5 we test our algorithm on several examples and show its effectiveness compare to the MC schemes.

2 Background

In this section, we list some fundamental concepts and results for further developments. First, we will review the Eulerian approach for constructing the backward flow map, which will be used in the proposed algorithm for solving the FP equation. Subsequently, we set up the Itô SDE governing the particle behaviors in the uncertain unsteady flow fields.

2.1 Eulerian approaches for the backward flow map

As in [25, 26, 43], we define a vector valued function $\Psi = (\Psi^1, \Psi^2, \cdots, \Psi^d) : \Omega \times \mathbb{R} \to \mathbb{R}^d$ where $\Omega \subset \mathbb{R}^d$ is the computational domain. At $t = 0$, we initialize these functions by

$$\Psi(\mathbf{x}, 0) = \mathbf{x} = (x^1, x^2, \cdots, x^d)\,. \tag{2}$$

These functions provide a labelling for any particle in the phase space at $t = 0$. In particular, any particle initially located at $(\mathbf{x}_0, 0) = (x_0^1, x_0^2, \cdots, x_0^d, 0)$ in the extended phase space can be **implicitly** represented by the intersection of d codimension-1 surfaces represented by $\cap_{i=1}^d \{\Psi^i(\mathbf{x}, 0) = x_0^i\}$ 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.

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

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

$$\frac{\partial \Psi(\mathbf{x}, t)}{\partial t} + (\mathbf{f} \cdot \nabla)\Psi(\mathbf{x}, t) = \mathbf{0} \tag{3}$$

with the initial condition (2). This **implicit** representation therefore embeds all path lines in the extended phase space. For instance, the trajectory of a particle initially located at $(\mathbf{x}_0, 0)$ 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.

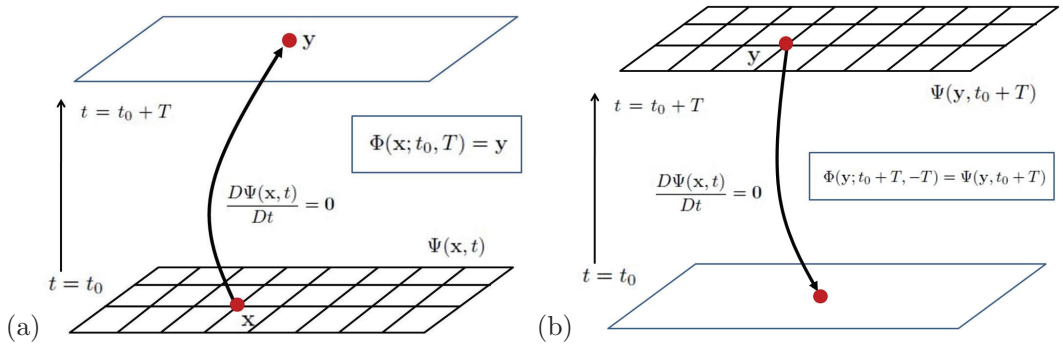

Figure 1: Lagrangian and Eulerian interpretations of the function Ψ [25]. (a) Lagrangian ray tracing from a given grid location $\mathbf{x}$ at $t = t_0$. Note that $\mathbf{y}$ might be a non-grid point. (b) Eulerian values of Ψ at a given grid location $\mathbf{y}$ at $t = t_0 + T$ gives the corresponding take-off location at $t = t_0$. Note the take-off location might not be a mesh point.

The solution to (3) contains much more information than what we have just interpreted above. Consider a given mesh location $\mathbf{y}$ in the phase space at time $t = T$, as shown in figure 1 (b), i.e. $(\mathbf{y}, T)$ in the extended phase space. Since the intersection $\cap_{i=1}^{d}\{\Psi^i(\mathbf{x},t) = \Psi^i(\mathbf{y},T)\}$ represents the path line in the extended phase space passing through $(\mathbf{y}, T)$, the level set functions $\Psi(\mathbf{y}, T)$ represent the coordinates of the takeoff location at $t = 0$ of a Lagrangian particle reaching $\mathbf{y}$ at $t = T$. Therefore, these level set functions defined on a uniform Cartesian mesh in fact give the *backward* flow map from $t = T$ to $t = 0$, i.e. $\Phi_T^0(\mathbf{y}) = \Psi(\mathbf{y}, T)$. Moreover, solution to the level set equations (3) for $t \in (0, T)$ provides also backward flow maps for all intermediate times, i.e. $\Phi_t^0(\mathbf{y}) = \Psi(\mathbf{y}, t)$.

That is, we obtain the *backward* flow map by numerically solving the Liouville equation *forward* in time. In particular, at each timestep, we solve the level set equation (3) *forward* in time from $t = t_n$ to $t = t_{n+1}$ with the initial condition $\Psi(\mathbf{x}, t_n) = \mathbf{x}$ imposed on the time level $t = t_n$. Then the one step backward flow map is given by $\Phi_{t_{n+1}}^{t_n}(\mathbf{x}) = \Psi(\mathbf{x}, t_{n+1})$, which can actually be constructed as an explicit formulation. For example, if the forward Euler method is used, we have

$$\Phi_{t_{n+1}}^{t_n}(\mathbf{x}_{i,j}) = \mathbf{x}_{i,j} - \Delta t\, \mathbf{f}(\mathbf{x}_{i,j}, t_n)$$

where $\Delta t = t_{n+1} - t_n$ and $\mathbf{x}_{i,j} \triangleq (x_i, y_j)$ with x_i the i-th grid point in the x-direction and y_j the j-th grid point in the y-direction. Higher order extensions are possible using typical weighted essentially non-oscillatory-total variation diminishing Runge-Kutta (WENO-TVDRK) approaches. For example, using the second order TVD-RK (TVD-RK2) scheme [35] we have

$$\Phi_{t_{n+1}}^{t_n}(\mathbf{x}_{i,j}) = \mathbf{x}_{i,j} - \frac{\Delta t}{2}\left(\mathbf{f}(\mathbf{x}_{i,j}, t_n) + \mathbf{f}(\mathbf{x}_{i,j}, t_{n+1})\right) + \frac{\Delta t^2}{2}\mathbf{f}(\mathbf{x}_{i,j}, t_{n+1}) \cdot \nabla \mathbf{f}(\mathbf{x}_{i,j}, t_n)\,. \tag{4}$$

2.2 The stochastic differential equation governing the uncertain unsteady flows

Dynamical systems are usually modeled by the ODE system (1). However, noise and uncertainty are common in measurements from any real applications and thus the evolution of particle trajectories does not necessarily satisfy (1) anymore. Assuming that the velocity field in the

dynamical system contains white noise, we model the evolution of a particle trajectory by the following Itô SDE

$$d\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)dt + \sigma(\mathbf{x}(t), t)d\mathbf{w}\,, \tag{5}$$

with the initial particle location at $\mathbf{x}(t_0) = \mathbf{x}_0 \in \mathbb{R}^d$. The function $\mathbf{f}$ is the underlying Lipschitz drift velocity function and σ is a $d \times d$ function matrix of space and time relating to the correlations between the random variables in different dimensions, which is a scalar matrix if the noise in different dimensions is assumed to be independent and identically distributed (i.i.d.). The function $\mathbf{w}$ denotes the Wiener process representing the standard Brownian motion which is independent of the random initial condition $\mathbf{x}_0$.

Since each particle trajectory now evolves with uncertainty according to (5), the flow map $\Phi_{t_0}^{t_0+T}$ from the initial time $t = t_0$ to the final time $t = t_0 + T$ also becomes stochastic. This makes the FTLE computation a challenging task. Most recent studies concerning the FTLE in uncertain velocity fields [39, 11] first conduct the MC simulation to solve the above SDE (5) for multiple runs. In each run, they might obtain a different arrival location at $t = t_0 + T$ of the particle starting from the same point $\mathbf{x}_0$ at $t = t_0$. After these runs, they collect all these different arrival locations for each particular particle and then take an *average* in some sense to obtain an *expected* flow map or directly an *expected* FTLE. In [11] for example, they define a so-called FTLE-D as

$$e_{t_0}^{t_0+T}(\mathbf{x}) = \frac{1}{T}\log\sqrt{\lambda_{\max}(E\left[\nabla\Phi\right]^T E\left[\nabla\Phi\right])}\,,$$

where $\nabla\Phi$ denotes the gradient of the flow map $\Phi_{t_0}^{t_0+T}$, $[.]^T$ represents the transpose of a matrix, $E\left[\cdot\right]$ is the expectation operator and $\lambda_{\max}(\cdot)$ is used to denote the maximum eigenvalue of the underlying matrix. In the implementation, $E\left[\nabla\Phi\right]$ is estimated by averaging the gradients of flow maps collected in all runs:

$$E\left[\nabla\Phi\right] \approx \frac{1}{m}\sum_{i=1}^{m}\nabla\Phi^{(i)}\,,$$

where m is the number of runs conducted and $\nabla\Phi^{(i)}$ can be estimated by the finite difference scheme, e.g. the central difference method, for every individual run.

Solving SDEs numerically is well-studied [22, 23, 18]. For example, [18] gives a MATLAB-based introduction to the SDE simulation. Yet the most widely used way is still the MC simulation. A detailed and complete toolbox for the MC simulation is presented in [38] which is called SDELAB also developed based on the MATLAB software. SDELAB is able to simulate sample paths of an SDE solution, compute statistics and estimate the parameters from data. In our numerical experiments given in Section 5, we will compare the results of our proposed algorithm with those from the simulations using the SDELAB.

Independent of the programming language or the toolbox used for the MC simulation, one has to set the number of MC simulation runs. Based on our experiences, if the noise level of the velocity data is relatively low, the MC simulation converges quickly. On the other hand, if the magnitude of the noise is relatively large, one might need to conduct the MC simulation many times. Even worse, we are not able to know beforehand how many runs is enough in order to obtain an acceptable and converged result. As a result, one has to monitor the result of each run and set a criterion to determine a time to stop the iterations.

3 The proposed approach

In this section, we propose an Eulerian approach to compute the FTLE of uncertain unsteady dynamical systems based on the *expected* flow map. Instead of the MC simulations, we construct a PDE (the FP equation) governing the evolution of the PDF of the stochastic arrival location of a certain particle. In view of the computational challenges in solving the FP equations with different initial conditions, we propose to use the Strang's operator splitting scheme to discretize

the FP equations after carefully treating the solution for the first timestep. Once the PDFs corresponding to the particles starting from all mesh points are obtained, we can define the *expected* flow map and thus the expected FTLE. At the end, we analyze the computational complexity of the proposed algorithm.

3.1 The Eulerian formulation

Let $p = p(\mathbf{x}, t; \mathbf{x}_0)$ be the PDF of the arrival location at time t of the particle taking off from $\mathbf{x}(t_0) = \mathbf{x}_0$ at the initial time $t = t_0$. Associated with the SDE (5), it can be shown that $p = p(\mathbf{x}, t; \mathbf{x}_0)$ satisfies the FP equation given by

$$\begin{aligned} &\frac{\partial p}{\partial t} + \nabla \cdot (\mathbf{f}p) = \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j} \left[D_{i,j}(\mathbf{x}, t)\, p \right] \\ &p(\mathbf{x}, t_0; \mathbf{x}_0) = \delta(\mathbf{x} - \mathbf{x}_0)\,, \end{aligned} \tag{6}$$

with the diffusion tensor $D(\mathbf{x}, t) = \sigma(\mathbf{x}, t)\sigma^T(\mathbf{x}, t)$. The initial condition is a Dirac delta function representing that the particle is initially located at the location $\mathbf{x} = \mathbf{x}_0$. We note that this equation is simply a convection diffusion equation. For simplicity, we only consider the case when the noise in each dimension is i.i.d. and then we have $D(\mathbf{x}, t)$ is a scalar matrix. Then the FP equation (6) can be written as

$$\frac{\partial p}{\partial t} + \nabla \cdot (\mathbf{f}p) = D_0 \Delta p\,,$$

where D_0 is the constant diffusion coefficient. Unlike others [39, 11], we propose in this work to directly solve this PDE using well-developed numerical schemes instead of conducting any MC simulations. If we further assume that the flow is incompressible, i.e. $\nabla \cdot \mathbf{f} = 0$, we have the following advection-diffusion equation

$$\begin{aligned} &\frac{\partial p}{\partial t} + \mathbf{f} \cdot \nabla p = D_0 \Delta p \\ &p(\mathbf{x}, t_0; \mathbf{x}_0) = \delta(\mathbf{x} - \mathbf{x}_0) \end{aligned} \tag{7}$$

with the velocity $\mathbf{f}$ given by the drift velocity function and the diffusion coefficient D_0 related to the variance of the noise or uncertainty in the measurements.

Once we obtain these PDFs corresponding to all the grid points, we propose the *expected* flow map $\Phi_{t_0}^{t_0+T} : \mathbb{R}^d \to \mathbb{R}^d$ as the mapping which takes the point $\mathbf{x}_0$ to the *expected* particle location at the final time $t = t_0 + T$, i.e

$$\hat{\Phi}_{t_0}^{t_0+T}(\mathbf{x}_0) = E[\Phi_{t_0}^{t_0+T}(\mathbf{x}_0)] = \int_{\mathbb{R}^d} \mathbf{x}\, p(\mathbf{x}, t_0 + T; \mathbf{x}_0)\, d\mathbf{x}\,, \tag{8}$$

with $p(\mathbf{x}, t; \mathbf{x}_0)$ satisfying the FP equation. Finally we define the *expected* FTLE field $e_{t_0}^{t_0+T}$ based on the *expected* flow map by

$$e_{t_0}^{t_0+T}(\mathbf{x}) = \frac{1}{T} \log \sqrt{\lambda_{\max}((\nabla \hat{\Phi})^T (\nabla \hat{\Phi}))}\,.$$

This definition extends the original definition of the FTLE. In particular, when there is no uncertainty in the velocity measurements, the diffusion coefficient D_0 drops to zero in the FP equation. The resulting PDF p is, therefore, a delta function and our definition reduces back to the original definition of the FTLE.

The term $\nabla \hat{\Phi}$ is the gradient of the *expected* flow map. It measures the change in the expected arrival location of the flow particle with respect to an infinitesimal change in the initial takeoff location. This term is equivalent to the expectation of the PDF gradient. Mathematically, this

expectation measures the *expected difference* between two PDFs corresponding to two different initial takeoff locations $\mathbf{x}_0$ and $\mathbf{y}_0$ by

$$\int_{\mathbb{R}^d \times \mathbb{R}^d} (\mathbf{x} - \mathbf{y})\, p(\mathbf{x}, t_0 + T; \mathbf{x}_0)\, p(\mathbf{y}, t_0 + T; \mathbf{y}_0)\, d\mathbf{x} d\mathbf{y}\,.$$

It equals to

$$\int_{\mathbb{R}^d} \mathbf{x}\, p(\mathbf{x}, t_0 + T; \mathbf{x}_0)\, d\mathbf{x} - \int_{\mathbb{R}^d} \mathbf{y}\, p(\mathbf{y}, t_0 + T; \mathbf{y}_0)\, d\mathbf{y} = \hat{\Phi}_{t_0}^{t_0+T}(\mathbf{x}_0) - \hat{\Phi}_{t_0}^{t_0+T}(\mathbf{y}_0)\,.$$

As we move these two initial takeoff locations towards each other giving the expectation of the PDF gradient, we have the gradient of our *expected* flow map. This is actually not too surprising, however. Since the expectation and the gradient operator are both linear, one can switch the operations and simply replace the term by the expectation of the PDF gradient.

The interpretation of the PDF gradient shares some similarities with the so-called FTLE-D as defined in [11] which computes the expectation by sending multiple rays from the adjacent locations $\mathbf{y}_0$ chosen near $\mathbf{x}_0$. This is a Lagrangian interpretation where one tries to implement MC-type simulations to approximate the PDF $p(\mathbf{x}, t_0+T; \mathbf{x}_0)$ and $p(\mathbf{y}, t_0+T; \mathbf{y}_0)$. Our Eulerian approach, on the other hand, computes the PDFs by a PDE formulation based on the FP equation.

3.2 Computational challenges

Seemingly straightforward, a simple numerical implementation might pose some computational challenges. The main concern is the overall computational complexity. Let Δx be the mesh size in each spatial direction, $N = O(\Delta x^{-1})$ be the number of mesh points in each spatial direction, and Δt be the timestep in solving the FP equation. Because the FP equation is of parabolic type, the typical computational complexity involved in obtaining the PDF $p = p(\mathbf{x}, t; \mathbf{x}_0)$ at the final time $t = t_0 + T$ is

$$O\left(N^d/\Delta t\right) = O\left(N^d/\Delta x^2\right) = O\left(N^{d+2}\right)$$

if an explicit finite difference method is used for the diffusion term. Since the initial condition is different for each particular grid point (given by the delta function centered at the corresponding grid point), we have to solve the FP equation for $O\left(N^d\right)$ different initial conditions. As a result, the overall computational complexity for finding all PDFs and therefore the *expected* flow map is $O(N^{2d+2})$, which is computationally very expensive. One main reason for this complexity is clearly that it assumes an explicit scheme is used for the diffusion part and, therefore, the method requires a relatively strict stability condition. One might attempt to replace it by an implicit scheme in order to relax the stability constraint. This is unfortunately difficult because the advection term in the equation might require some root-finding step. A better approach is to develop some explicit-implicit methods by treating the advection part with an explicit scheme while the diffusion part implicitly. By carefully choosing the timestep, one might be able to relax the stability condition for the whole numerical scheme. Another possible choice is mentioned in [3] where a numerical algorithm has been developed using an exponential time differencing scheme in time and a spectral collocation method in space. However, in our application, we have to solve the FP equation for $O(N^d)$ different initial conditions. We need to design some algorithms which would allow us to reuse parts of any intermediate computations.

Another issue is the handling of the initial condition $p(\mathbf{x}, t_0; \mathbf{x}_0) = \delta(\mathbf{x} - \mathbf{x}_0)$. In the level set community [34, 33], the delta function in the 2-D case is approximated using

$$\delta_\epsilon(x - x_0, y - y_0) = \begin{cases} 0 & \text{if } d(x, y) > \epsilon, \\ \dfrac{1}{2\epsilon} + \dfrac{1}{2\epsilon} \cos\left(\dfrac{\pi d(x, y)}{\epsilon}\right) & \text{otherwise,} \end{cases}$$

where $d(x,y) = \sqrt{(x-x_0)^2 + (y-y_0)^2}$. The usual practice is to pick $\epsilon = O(\Delta x)$ so that the delta function is resolved by several mesh points. However, it might be difficult to decide a proper choice of the regularization parameter ϵ in the current application. If the parameter is too small, we might not have enough grid points to resolve the variation in the function value. If it is too large, on the other hand, it smoothes out the function over a neighborhood which has the same effect as having some diffusion in the initial condition. This will introduce some inconsistency in the approximation of the numerical solution to the FP equation. Therefore, we need to carefully design a numerical procedure to handle this initial condition. A simple computational strategy will be given in Section 3.4.

3.3 Strang's splitting scheme

In view of the computational challenges we mentioned above, in this subsection we first propose to use Strang's splitting scheme [41] to discretize the FP equation from $t = t_n$ to $t = t_{n+1}$. In particular, suppose the PDF $p(x_i, y_j, t_n; x_0, y_0)$ is known at time $t = t_n$ at each grid point (x_i, y_j) where (x_0, y_0) is the initial location of the particle at $t = t_0$. To simplify the notations in the following discussion, we denote $p(x_i, y_j, t_n; x_0, y_0)$ by $p^n_{i,j}$, temporally hiding the dependence on the initial location. Then we want to obtain $p^{n+1}_{i,j}$ by discretizing the following Cauchy problem:

$$\begin{aligned} &\frac{\partial p}{\partial t} + \mathbf{f} \cdot \nabla p = D_0 \Delta p \\ &p(x, y, t_n; x_0, y_0)|_{(x_i, y_j)} = p^n_{i,j}\,. \end{aligned}$$

The decomposition of the FP equation consists of three steps using Strang's splitting scheme.

Step One. In the first step, we solve the following Liouville equation

$$\begin{aligned} &\frac{\partial p}{\partial t} + \mathbf{f} \cdot \nabla p = 0 \\ &p(x, y, t_n; x_0, y_0)|_{(x_i, y_j)} = p^n_{i,j} \end{aligned}$$

for half a timestep from $t = t_n$ to $t = t_n + \frac{1}{2}\Delta t$ to obtain an intermediate solution, denoted by $p^{n+\frac{1}{2}}_{i,j}$. In fact, this equation is equivalent to $Dp/Dt = 0$ implying that the function $p(x, y, t; x_0, y_0)$ is constant along each particle trajectory guided by the drift velocity field $\mathbf{f}(x, y, t)$. As a result, we have

$$p^{n+\frac{1}{2}}_{i,j} = p\left(\Phi^{t_n}_{t_n+\frac{1}{2}\Delta t}(x_i, y_j), t_n; x_0, y_0\right)$$

where $\Phi^{t_n}_{t_n+\frac{1}{2}\Delta t}(\cdot)$ denotes the backward flow map from $t = t_n + \frac{1}{2}\Delta t$ to $t = t_n$. This can be easily constructed using (4) proposed in Section 2.1. After that, constructing $p^{n+\frac{1}{2}}_{i,j}$ involves only an interpolation based on $p^n_{i,j}$, which could be done using any well-developed numerical scheme such as the function pchip/interp1/interp2/interp3/interpn developed in MATLAB.

Step Two. In the second step we handle the diffusion operator. In particular, we solve the following Cauchy problem

$$\begin{aligned} &\frac{\partial p}{\partial t} = D_0 \Delta p \\ &p(x, y, t_n; x_0, y_0)|_{(x_i, y_j)} = p^{n+\frac{1}{2}}_{i,j} \end{aligned}$$

for a whole timestep from $t = t_n$ to $t = t_{n+1}$ to obtain another intermediate solution, denoted by $(p_{i,j}^{n+1})'$, where $p_{i,j}^{n+\frac{1}{2}}$ is the intermediate solution obtained in the first step. This equation is the standard diffusion equation with a constant diffusion coefficient which could be easily solved using any typical method. One simple approach is to use a finite difference method. To keep the second order accuracy and relax the restriction on the timestep Δt of the algorithm, one can simply use the well-known Crank-Nicolson (CN) scheme. To obtain the spectral accuracy in the solution, one might also use the FFT (for the periodic flows) or the DST (for Dirichlet boundary condition on the flows). However, since the operator splitting has already introduced certain errors in the numerical solution, there is no particular reason to require extremely high accuracy in this intermediate stage.

Step Three. In the third step, we solve

$$\frac{\partial p}{\partial t} + \mathbf{f} \cdot \nabla p = 0$$
$$p(x, y, t_n + \frac{1}{2}\Delta t; x_0, y_0)|_{(x_i, y_j)} = \left(p_{i,j}^{n+1}\right)'$$

from $t = t_n + \frac{1}{2}\Delta t$ to $t = t_{n+1}$ to obtain $p_{i,j}^{n+1}$. This can also be easily achieved by using (4) and an interpolation together. In particular, we first use (4) to obtain the backward flow map $\Phi_{t_{n+1}}^{t_n+\frac{1}{2}\Delta t}(x_i, y_j)$, then $p_{i,j}^{n+1}$ can be obtained by interpolating at $\Phi_{t_{n+1}}^{t_n+\frac{1}{2}\Delta t}(x_i, y_j)$ based on $\left(p_{i,j}^{n+1}\right)'$.

In **Step One** and **Step Three**, we need to construct the numerical solution to the Liouville equation. This imposes a stability constraint $\Delta t = O(\Delta x)$ limited by the CFL condition. Because we are using an implicit scheme (the CN scheme) in **Step Two**, the overall numerical scheme is therefore constrained only by the CFL condition. Note that if the flow map between any time-levels is pre-computed beforehand, however, there will be no stability condition on both **Step One** and **Step Three** when constructing the PDF. This makes the overall scheme unconditionally stable and Δt can be chosen solely based on the consideration of the accuracy instead of the stability. In Section 5, because we construct the flow map together along with the PDF, we have used $\Delta t = O(\Delta x)$ in the numerical implementations.

Concerning the computational complexity, since the implicit scheme to the diffusion operator can be inverted using the LU decomposition or the ADI method, the overall computational complexity is now dropped to $O(N^{d+1})$ for each initial condition corresponding to the PDF associated to that location. Even though the overall computational complexity for solving all *expected* flow maps is still $O(N^{2d+1})$ which might seem to be overwhelmingly expensive, the flow maps $\Phi_{t_n+\frac{1}{2}\Delta t}^{t_n}$ and $\Phi_{t_{n+1}}^{t_n+\frac{1}{2}\Delta t}$ are actually reused for all $O(N^d)$ different initial conditions in **Step One** and **Step Three**, respectively. Now, because the MC simulation requires $O(LN^{d+1})$ operations for all mesh points where L is the runs of simulations conducted, our PDE-based approach would be computationally very attractive if $L \gg O(N^d)$. Qualitatively, as the noise level gets larger in the velocity field, one has to increase the number of trials in order to obtain a better approximation of the PDF on the arrival locations. Therefore, intuitively, our proposed approach is preferable for flows with a higher level of uncertainty. Having said that, however, we do not have any quantitative results on the convergence of the MC simulation in our current application. There is indeed analysis of the MC simulations [21], including some strong convergence and weak convergence results. For example, an approximating process Y is said to be strongly convergent with order $\gamma \in (0, \infty]$ if there exists a constant $K \in (0, \infty]$ such that $E|X_T - Y_n| \leq K\Delta t^\gamma$, where $\Delta t \in (0, 1]$ is the step size, X_T is the exact solution at time $t = T$ and Y_n is the approximation at $t = T$. Different integration algorithms may lead to different orders of accuracy in the MC simulations. For example, the Milstein algorithm has

a first order of strong convergence [31], while the Euler-Maruyama algorithm is of the order 0.5 [1]. However, there is no *a prior* explicit relationship between the value of L and the error in the MC simulation. As a result, one has to monitor the result of each run until a certain accuracy in the numerical solution is achieved.

In applications when the noise in each dimension is not independent and identically distributed (i.i.d.), the diffusion tensor $D(\mathbf{x}, t)$ in (6) will no longer be a constant multiple of an identity matrix. In this case, **Step One** and **Step Three** of the overall algorithm can remain unchanged. The only modification required is an efficient solver for the anisotropic diffusion equation in **Step Two** given by

$$\frac{\partial p}{\partial t} = \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}\left[D_{i,j}(\mathbf{x}, t)\, p\right]$$
$$p(x, y, t_n; x_0, y_0)|_{(x_i, y_j)} = p_{i,j}^{n+\frac{1}{2}}.$$

3.4 The treatment of the first timestep

As mentioned, the discretization to the initial condition $p(x, y, t_0; x_0, y_0) = \delta(x - x_0, y - y_0)$ might not be that straightforward. Therefore, we need to carefully construct the solution at the first timestep $t = t_1 = t_0 + \Delta t$, i.e. $p(x, y, t_1; x_0, y_0)$, based on the initial condition. It turns out that, without the need to discretize the initial condition, $p(x, y, t_1; x_0, y_0)$ can be explicitly represented in the operator splitting scheme. For simplicity, we consider only the two dimensional case. Higher dimensional generalization is relatively straightforward and, therefore, we will skip the discussion. We also represent the backward flow map from $t = t_1$ to $t = t_0$ obtained by our Eulerian approach as in Section 2.1 using $\Phi_{t_1}^{t_0}(x, y) = \left(\phi_{t_1}^{t_0}(x, y), \psi_{t_1}^{t_0}(x, y)\right)$.

In **Step One**, we solve the Liouville equation with the initial condition

$$p(x, y, t_0; x_0, y_0) = \delta(x - x_0, y - y_0)$$

from $t = t_0$ to $t = t_0 + \frac{1}{2}\Delta t$. The solution can be analytically constructed as

$$\begin{aligned} p^{\frac{1}{2}}(x, y) &= p\left(\Phi_{t_0+\frac{1}{2}\Delta t}^{t_0}(x, y), t_0; x_0, y_0\right) \\ &= \delta\left(\phi_{t_0+\frac{1}{2}\Delta t}^{t_0}(x, y) - x_0, \psi_{t_0+\frac{1}{2}\Delta t}^{t_0}(x, y) - y_0\right) \\ &= \delta\left(x - \phi_{t_0}^{t_0+\frac{1}{2}\Delta t}(x_0, y_0), y - \psi_{t_0}^{t_0+\frac{1}{2}\Delta t}(x_0, y_0)\right). \end{aligned}$$

In the second step, we obtain the solution to the diffusion equation from $t = t_0$ to $t = t_0 + \Delta t$ with the initial condition $p(x, y, t_0; x_0, y_0) = p^{\frac{1}{2}}(x, y)$ as

$$\left(p^1\right)'(x, y) = G_{\Delta t} * p^{\frac{1}{2}}(x, y) = \frac{1}{4\pi D_0 \Delta t} \exp\left[-\frac{1}{4D_0\Delta t}\left\|\mathbf{x} - \Phi_{t_0}^{t_0+\frac{1}{2}\Delta t}(x_0, y_0)\right\|_2^2\right]$$

where $G_{\Delta t}$ is the free space heat kernel for diffusion for a timestep of Δt and $*$ is the usual convolution operator. Finally, in **Step Three**, the advection equation is solved from $t = t_0 + \frac{1}{2}\Delta t$ to $t = t_0 + \Delta t$ with the initial condition $p\left(x, y, t_0 + \frac{1}{2}\Delta t; x_0, y_0\right) = \left(p^1\right)'(x, y)$. The solution can also be constructed using the flow map given by

$$\begin{aligned} p^1(x, y) &= \left(p^1\right)'\left(\Phi_{t_0+\Delta t}^{t_0+\frac{1}{2}\Delta t}(x, y)\right) \\ &= \frac{1}{4\pi D_0 \Delta t} \exp\left[-\frac{1}{4D_0\Delta t}\left\|\Phi_{t_0+\Delta t}^{t_0+\frac{1}{2}\Delta t}(x, y) - \Phi_{t_0}^{t_0+\frac{1}{2}\Delta t}(x_0, y_0)\right\|_2^2\right]. \end{aligned}$$

Therefore, instead of plugging in the delta function to the initial condition in the first step, we sample the approximated solution $p^1(x, y)$ constructed above at the grid points $(x, y) = (x_i, y_j)$ and use that as the initial condition for the next timestep.

As we can see from this construction, the heat kernel has a support of $O(\Delta t^{1/2})$. When we have $\Delta t = O(\Delta x)$, the Gaussian is distributed and sampled locally on a domain of size $O(\Delta x^{1/2}) \gg O(\Delta x)$. This approach thus automatically provides a better numerical initial condition for later computations than the smeared delta function approach as in the usual level set community.

4 An adaptive refinement approach for visualization

To further improve the computational efficiency of the *expected* FTLE, we propose the following adaptive refinement strategy. We first determine the coarsest and the finest scales for the computations with the mesh size denoted by Δx_0 and Δx_M. For simplicity, we can relate them by $\Delta x_M = 2^{-M}\Delta x_0$ for some integer $M > 0$. In practice, one can already significantly reduce the computational time using a relatively small integer M such as 2 or 3. On the finest scale Δx_M, we solve the Liouville equation for all flow maps Φ between the initial time and the final time. These solutions will be stored on the disk and will be used in all interpolation steps for solving the FP equations for the rest of the algorithm.

Now, at each grid point on the coarsest level Δx_0, we compute the *expected* flow map in a small neighborhood determined by the finest scale Δx_M, i.e. a grid point on the coarsest level together with four other initial locations at $(x, y) = (x_i \pm \Delta x_M, y_j \pm \Delta x_M)$. This means that, for each grid point on the coarsest level Δx_0, we compute five *expected* flow maps by solving the FP equation five times (with the delta function locating at different initial conditions). Then we can construct the corresponding deformation tensor at the coarsest level and obtain a very rough visualization of the *expected* FTLE.

To improve the resolution, we collect a subset of those grid points which have the *expected* FTLE larger than a thresholding value μ_0 and denote this set by Ω_0. Then we move to the next finer level Δx_1. For each grid point from Ω_0, we collect eight neighboring grid points in the neighborhood

$$(x_i \pm \Delta x_1, y_j), (x_i, y_j \pm \Delta x_1), (x_i + \Delta x_1, y_j \pm \Delta x_1), (x_i - \Delta x_1, y_j \pm \Delta x_1)$$

and compute the *expected* FTLE if its value has not been found. We collect a subset of all grid points which have the *expected* FTLE larger than μ_1 and denote the set by Ω_1. This procedure continues iteratively until in the last stage we have constructed the set Ω_{M-1} and have determined the *expected* FTLE on the finest scale

$$(x_i \pm \Delta x_M, y_j), (x_i, y_j \pm \Delta x_M), (x_i + \Delta x_M, y_j \pm \Delta x_M), (x_i - \Delta x_M, y_j \pm \Delta x_M)\,.$$

Finally, since the *expected* FTLE is not available at all grid points on the finest level Δx_M, we propose to fill in the missing values using the elliptic continuation. That is, we solve $-\Delta e = 0$ on those grid points where the *expected* FTLE is missing while keeping the existing value if we have already computed it from the above iterative procedure. Mathematically, we denote Ω' to be the set where we have visited in the above procedure with e' to be the corresponding value of the *expected* FTLE, and let $\Omega \backslash \Omega'$ be the set where the *expected* FTLE value is missing. Then the continuation can be easily computed by solving the following equation within the whole computational domain,

$$\left(\mathbb{1}_{\Omega'} - \mathbb{1}_{\Omega\backslash\Omega'}\Delta\right) e = \mathbb{1}_{\Omega'} e'$$

with the boundary condition $\partial_n e = 0$ on $\partial\Omega$ where the function $\mathbb{1}_{\Omega'}(\mathbf{x})$ is the characteristic function equal to 1 if $\mathbf{x} \in \Omega'$ and 0 otherwise.

In the iteration procedure, the choice of the parameter μ_m is obviously not unique. The smaller the value of this thresholding parameter, the more times the FP equation needs to be solved. If this value is chosen to be too large, we will not be able to resolve the FTLE solution on a finer scale. A possible choice is to determine μ_m adaptively on each level of refinement by setting it to be the average or the median of all available *expected* FTLE values. In this work, we determine the threshold value by the largest k-percentile of all available *expected* FTLE values. We have considered a wide range of k and found that even when we use as few as top 5-percentile FTLE values, the adaptive refinement approach would still lead to a numerical solution that matches extremely well with that of the full implementation.

Finally, note that in this adaptive algorithm, all *expected* flow maps are determined by solving the FP equation on the same finest scale Δx_M and therefore the accuracy can be consistently maintained on the same level. The adaptivity is imposed only on the visualization stage. To improve the computational time of the overall algorithm, however, one might explore parallel implementations for solving the FP equation such as various domain decomposition methods for the advection-diffusion equation. But this is not the main concentration of the current work and, therefore, we will not further investigate this direction.

5 Numerical results

In this section, we will demonstrate our proposed algorithm on two examples. The first example is the double gyre flow whose velocity field is synthetic and analytically determined by a stream-function. The second velocity field is given by a real dataset. Indeed, our proposed algorithm can also be easily extended to three-dimensions. It might require further improvement in the computational efficiency, and we will investigate any high dimensional example in the future.

5.1 The double gyre flow

This example is taken from [40] to describe a periodically varying double-gyre. The flow is modeled by the following stream-function $\psi(x, y, t) = A \sin[\pi g(x, t)] \sin(\pi y)$ where

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

In this example, we follow [40] and use $A = 0.1$, $\omega = 2\pi/10$ and $\epsilon = 0.1$.

At first, we use our proposed approach to simulate the evolution of the PDF $p(x, y, t; x_0, y_0)$ which corresponds to a particular particle taking off from (x_0, y_0) at the initial time. We set $D_0 = 0.001$ and use 513×257 mesh points to discretize the computational domain $\Omega = [0, 2]\times[0, 1]$, i.e. $\Delta x = \Delta y = 1/256$, where all required interpolation steps are implemented using the interp2 function in MATLAB. Figures 2 and 3 show the solutions of the PDFs $p(x, y, t; 1, 0.5)$ and $p(x, y, t; 0.5, 0.75)$, respectively, at several time slices. At the beginning, the PDF is densely distributed near the center where a high peak value is achieved. As time evolves, the PDF spreads all around and becomes flatter and flatter. Then we numerically verify the accuracy of the proposed approach. The blue line in Figure 4 shows the L_2 error of the PDF $p(x, y, t; 0.5, 0.75)$ at time $t = 10$ computed using our proposed approach with Δx varying from $1/32$ to $1/512$ while keeping $\Delta t/\Delta x$ fixed. We can see that our proposed approach is approximately second order accurate, which matches our expectation. We have also implemented the corresponding Lie's splitting scheme, where one takes one whole Δt step in **Step One** while removing the **Step Three** in the algorithm. The corresponding L_2 error of the PDF $p(x, y, 10; 0.5, 0.75)$ is plotted as the red line in Figure 4, which does show approximately first order accuracy. Since we do not have the exact solution, we first directly solve equation (7)

using the explicit scheme with small Δx and Δt and then use the resulting solution as the reference solution.

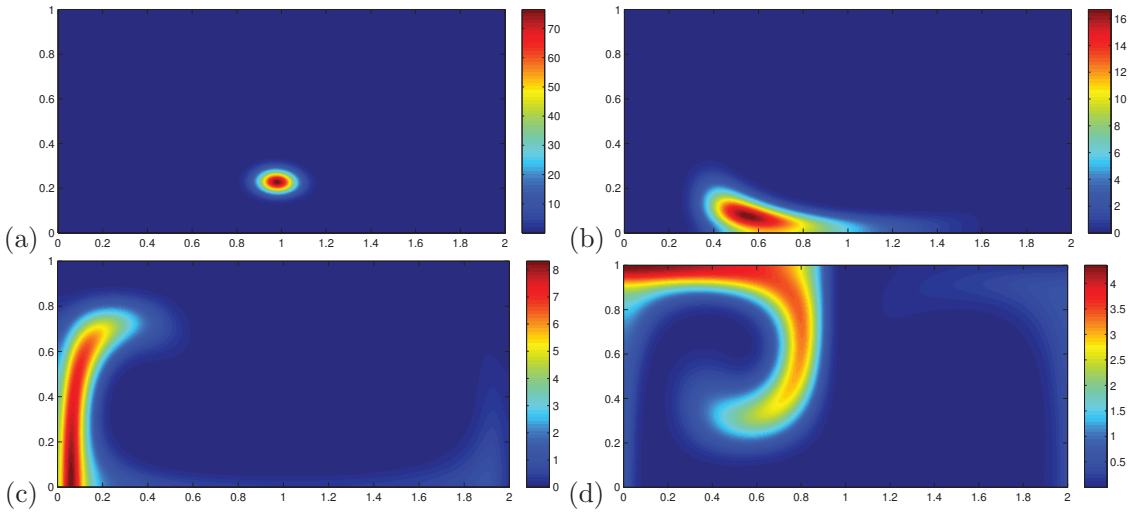

Figure 2: (Section 5.1) The PDF $p(x, y, t; 1, 0.5)$ computed using $\Delta x = \Delta y = 1/256$ at (a) $t = 1$, (b) $t = 3$, (c) $t = 6$ and (d) $t = 10$.

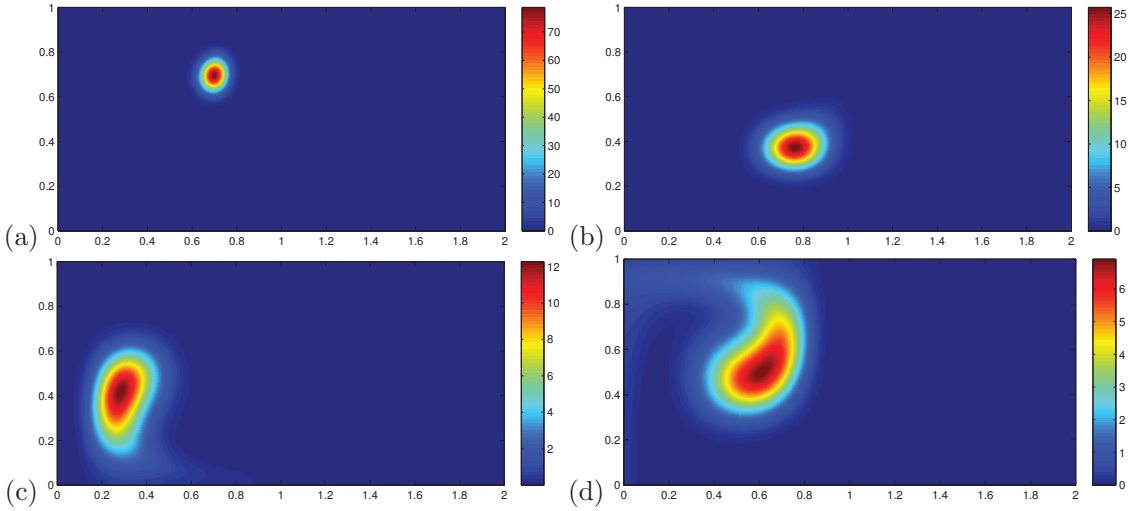

Figure 3: (Section 5.1) The PDF $p(x, y, t; 0.5, 0.75)$ computed using $\Delta x = \Delta y = 1/256$ at (a) $t = 1$, (b) $t = 3$, (c) $t = 6$ and (d) $t = 10$.

Then we compute the FTLE e_0^{10} based on the *expected* flow map $\hat{\Phi}_0^{10}$. Figure 5 shows the solutions of e_0^{10} for $D_0 = 0.0001$, 0.001, 0.005 and 0.01, respectively, computed using the mesh size $\Delta x = \Delta y = 1/128$. We can see that, as the diffusion coefficient D_0 increases, the corresponding FTLE ridge becomes wider and flatter. As a comparison, we have also used the SDELAB toolbox [38] to directly conduct the MC simulation, as discussed in Section 2.2. In particular, we first conduct the MC simulation with a large number, L, of trials and thus can obtain L stochastic arrival locations at $t = 10$ for each particular particle starting from a mesh point $\mathbf{x}_{i,j}$ at $t = 0$. Subsequently, we take an average of these L arrival locations to approximate the *expected* flow map $\hat{\Phi}_0^{10}(\mathbf{x}_{i,j})$ for this mesh point $\mathbf{x}_{i,j}$. After collecting the approximations

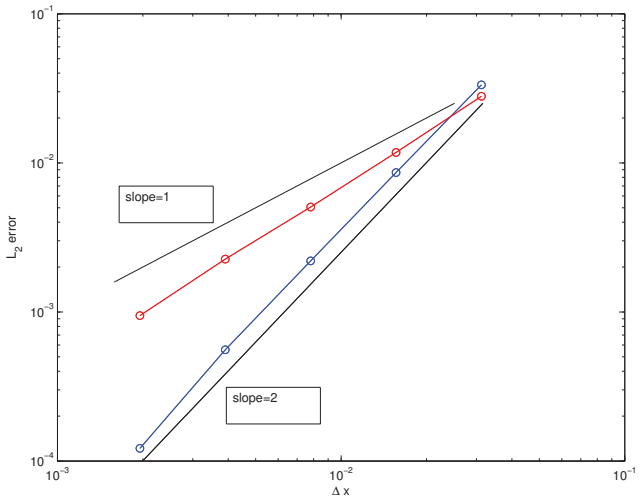

Figure 4: (Section 5.1) The L_2 error of the PDF $p(x, y, 10; 0.5, 0.75)$ computed with Δx varying from 1/32 to 1/512. The blue line corresponds to the solutions computed using our proposed Strang's splitting approach, while the red line represents the corresponding solutions computed using the Lie's splitting scheme. Reference lines are plotted in black.

to $\hat{\Phi}_0^{10}(\mathbf{x}_{i,j})$ for all mesh points, we are able to compute the corresponding FTLE e_0^{10}. Figure 6 gives the corresponding solutions when $L = 900$. We cannot observe obvious FTLE ridges except for the relatively small diffusion coefficient $D_0 = 0.0001$. However, as we increase L from 900 to 10000, the solutions (as shown in Figure 7) using the MC simulation for all D_0's become closer to the solutions of our proposed approach. In particular, the solution for $D_0 = 0.001$ is already good using $L = 10000$. Yet we still cannot see any obvious FTLE ridge for $D_0 = 0.005$ and $D_0 = 0.01$.

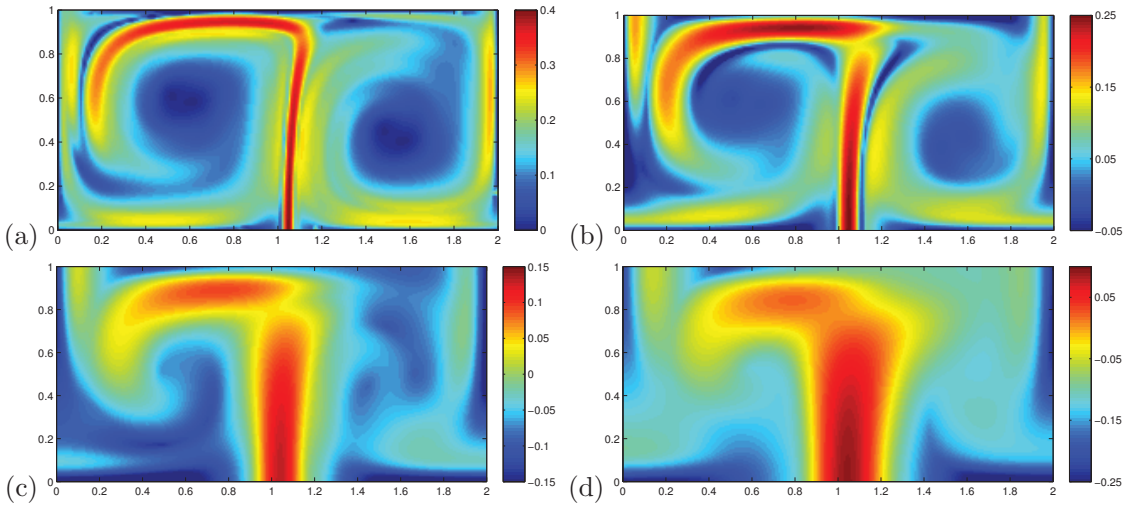

Figure 5: (Section 5.1) The FTLE e_0^{10} with $\Delta x = \Delta y = 1/128$ for (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

We also increase the number of trials in the MC simulation L from 10000 to 90000 for

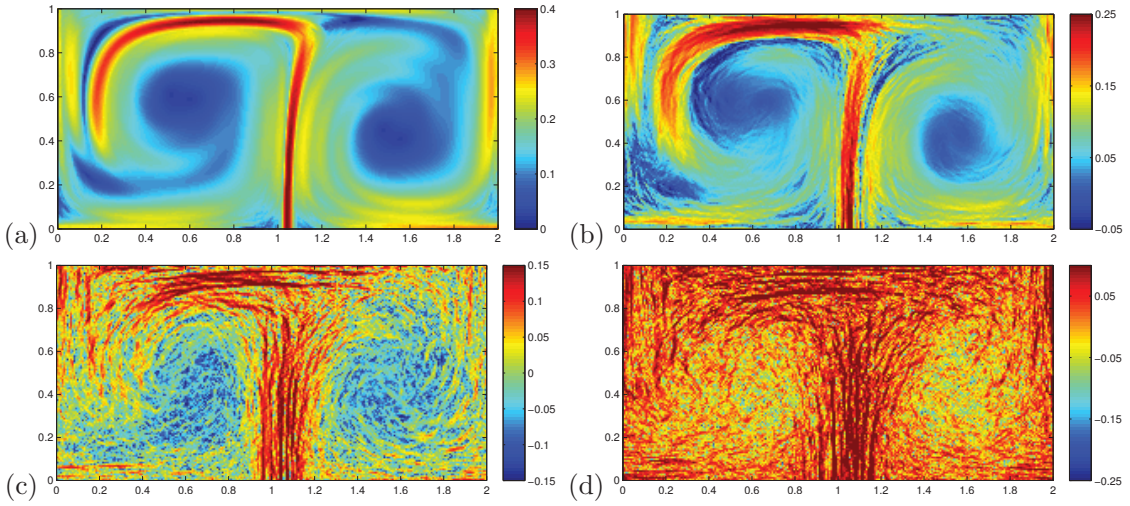

Figure 6: (Section 5.1) The FTLE e_0^{10} computed by the MC simulation with 900 trials using $\Delta x = \Delta y = 1/128$ with (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

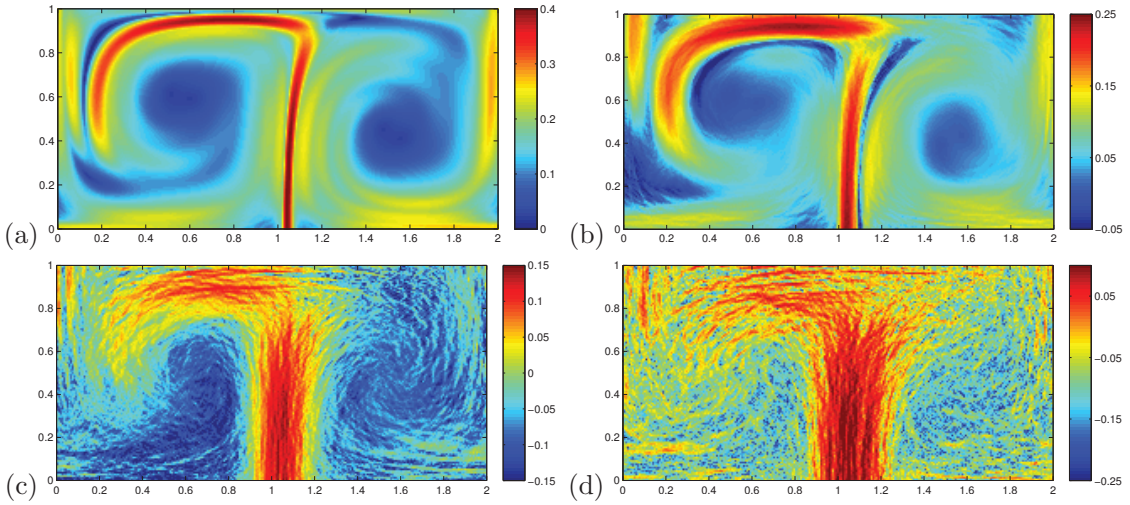

Figure 7: (Section 5.1) The FTLE e_0^{10} computed by the MC simulation with 10000 trials using $\Delta x = \Delta y = 1/128$ with (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

$D_0 = 0.005$ and $D_0 = 0.01$. The solutions are shown in Figure 8. As expected, these solutions get even closer to the solutions of our proposed approach given in Figure 5(c) and (d). But the two solutions still do not have sufficiently smooth FTLE ridges. We believe that the solutions of the MC simulation will eventually converge to the solutions computed using our approach as we further increase the number of runs.

L	900	10000	90000	Our approach
Duration (in hours)	1.01	11.11	160.22	36.79

Table 1: Computational time of our proposed algorithm and the MC scheme with different L's

Another issue concerning the MC scheme is the treatment of the boundary condition. The double gyre flow is a measure-preserving flow with no flux going across the boundary of the

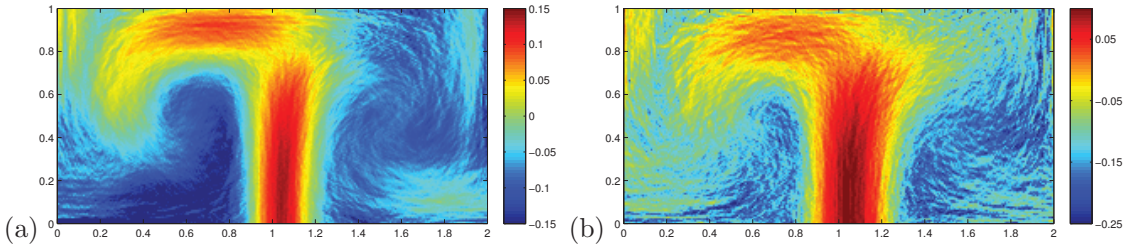

Figure 8: (Section 5.1) The FTLE e_0^{10} computed by the MC simulation with 90000 trials using $\Delta x = \Delta y = 1/128$ with (a) $D_0 = 0.005$, (b) $D_0 = 0.01$.

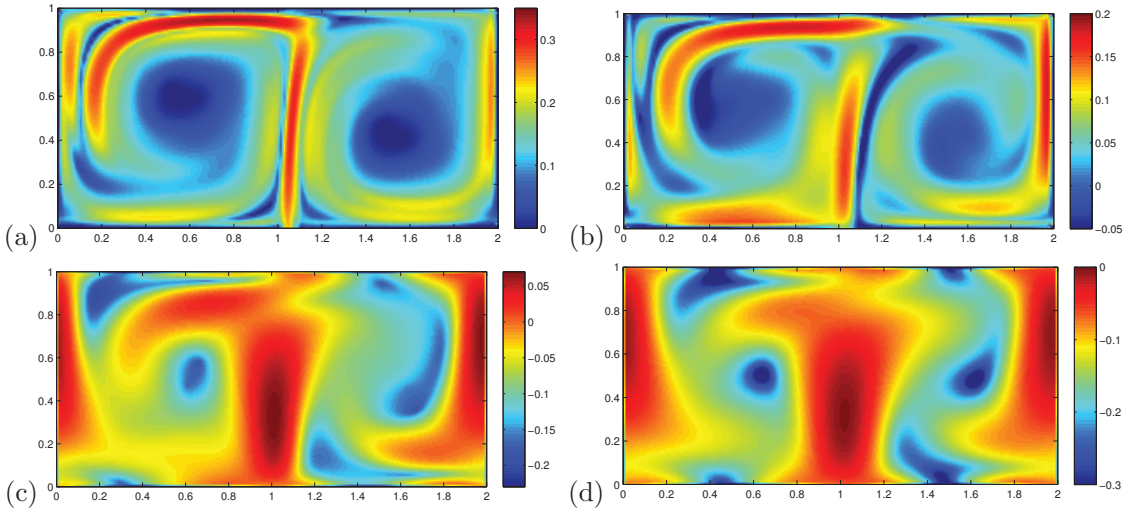

Figure 9: (Section 5.1) The FTLE e_0^{10} with the periodic boundary condition and mesh size $\Delta x = \Delta y = 1/128$ corresponding to (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

computational domain Ω. However, some particles might actually leave the computational domain during the MC simulation due to the Brownian motion. In the previous implementations of the MC simulation, we have set a mechanism to monitor the evolution of each particle trajectory and abandoned the trajectory once it leaves the domain. This might introduce unnecessary error in the numerical simulations. To better compare the solutions using the MC scheme and our proposed approach, we propose to impose the periodic boundary condition for both approaches. The solutions of our proposed approach are given in Figure 9, from which we can still clearly identify the FTLE ridges for all D_0's. The solutions of the MC simulations are given in Figure 10 and 11, corresponding to $L = 10000$ and 90000, respectively. We can see that the solution for $L = 90000$ is closer to the solution of our approach. Furthermore, no obvious FTLE ridge can be identified for $D_0 = 0.005$ and $D_0 = 0.01$, even with $L = 90000$.

Here we discuss the adaptive refinement approach as proposed in Section 4. We first set the coarsest and the finest scales with the mesh size $\Delta x = \Delta y = 1/32$ and $\Delta x = \Delta y = 1/128$ respectively, i.e. we simply have $M = 2$. In the implementation, the thresholding values μ_0 and μ_1 are determined based on the k-percentile of all available *expected* FTLE values. This means that $k = 100$ implies the original algorithm and $k = 50$ leads to the median of all computed values. First, we set $k = 20$ and compute the FTLE e_0^{10} with the periodic boundary condition. That is, we use the adaptive refinement approach to recompute the solutions given

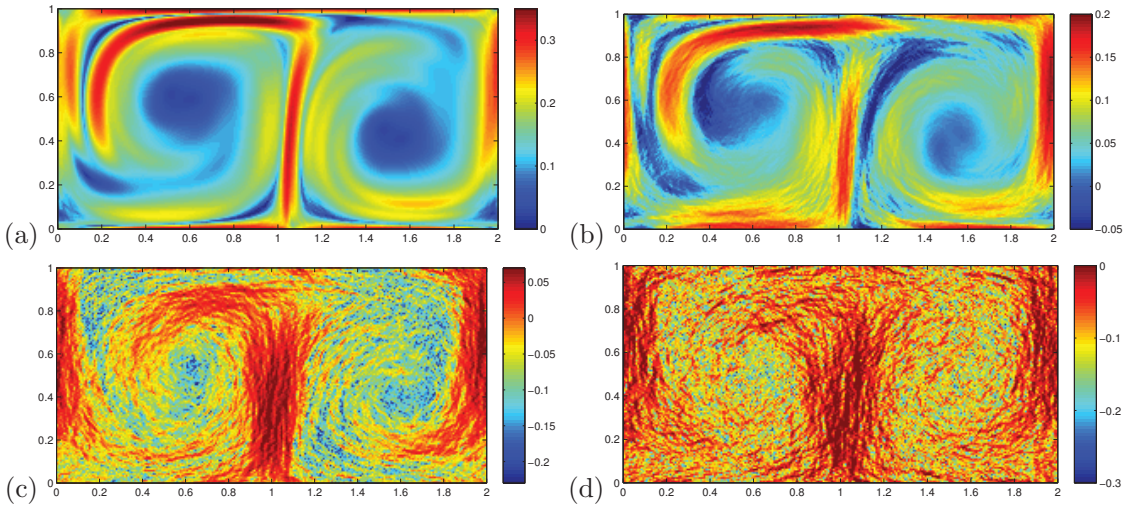

Figure 10: (Section 5.1) The FTLE e_0^{10} computed by the MC simulation with 10000 trials using the periodic boundary condition and $\Delta x = \Delta y = 1/128$ corresponding to (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

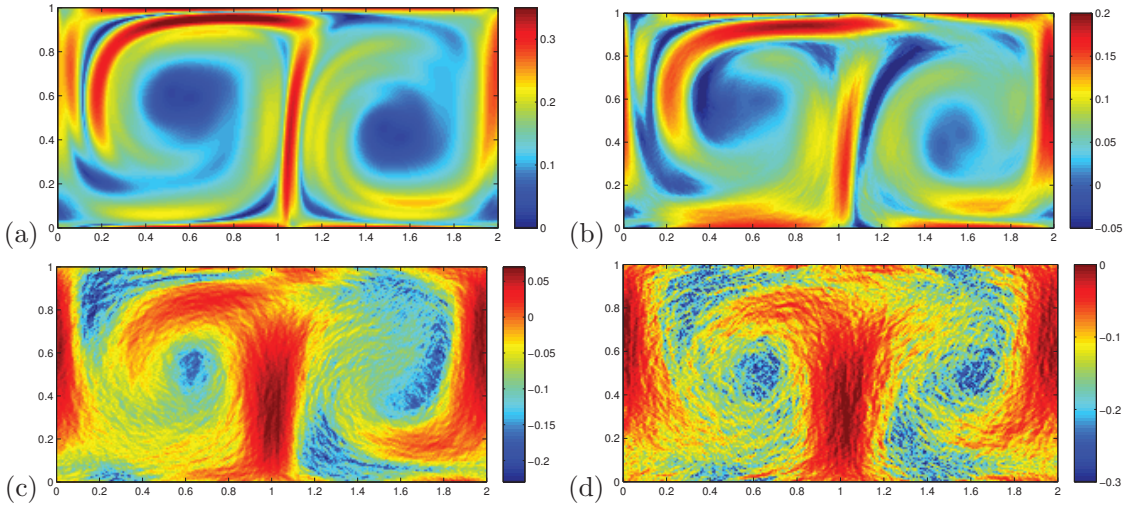

Figure 11: (Section 5.1) The FTLE e_0^{10} computed by the MC simulation with 90000 trials using the periodic boundary condition and $\Delta x = \Delta y = 1/128$ corresponding to (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

in Figure 9. The results corresponding to different D_0's are shown in Figure 12. We can see that the adaptive approach gives very good approximations to the solutions in Figure 9. To measure the difference between these two solutions in Figures 9 and 12 more quantitatively, we have plotted the absolute difference in Figure 13. We first concentrate on those locations with large *expected* FTLE values computed on the finest adaptive level. In Figure 14, these grid locations are highlighted in red. Since the solution at these places matches exactly with the direct computations, the absolute difference at those locations is in fact zero. The region with large deviations in Figure 13 mainly concentrates in those locations whose FTLE values are determined from the elliptic continuation, as highlighted in green in Figure 14.

In terms of the computational efficiency, we can clearly see the effect of the proposed adap-

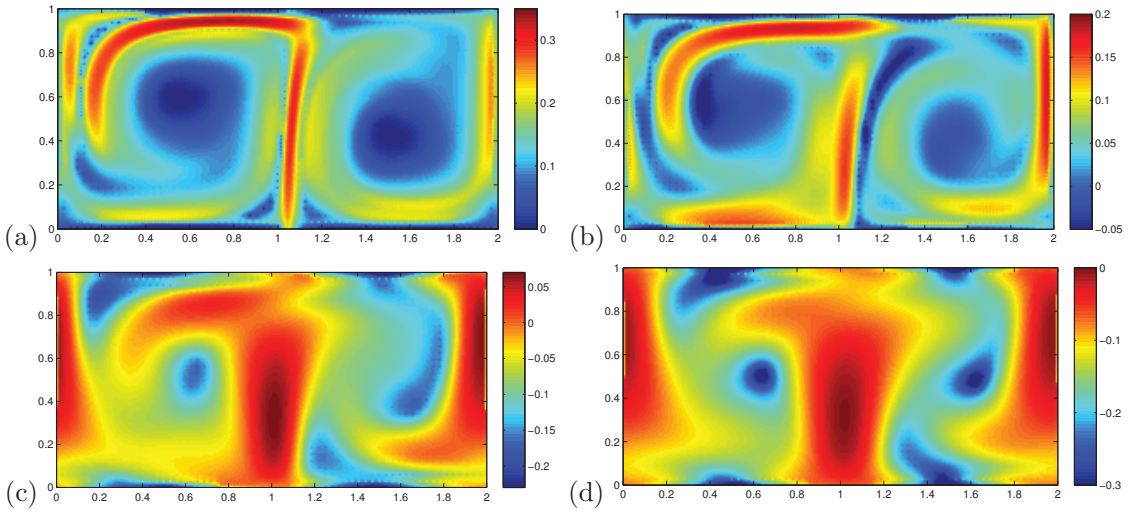

Figure 12: (Section 5.1) The FTLE e_0^{10} computed using the adaptive refinement approach with $k = 20$ corresponding to (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$. These solutions match well with those in Figure 9.

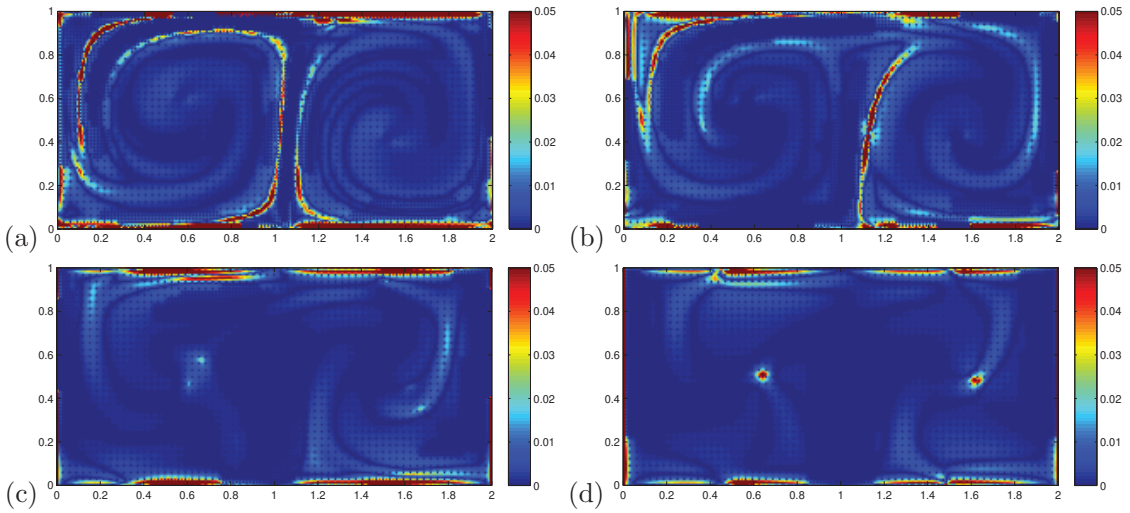

Figure 13: (Section 5.1) The absolute difference between the solutions in Figure 9 and Figure 12 corresponding to (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

tivity that most of the computational time is now spent on those locations near the ridge where the FTLE is relatively large. For those regions away from the ridge, we simply fill in the values approximately using an elliptic operator. This significantly reduces the computational time. In particular, the adaptive approach takes about 35.1%, 38.1% and 44.0% of the original computational time, for $k = 5$, 10 and 20, respectively. For example, the computation of e_0^{10} in Figure 9 and 12 takes about 36.85 hours and 16.22 hours, respectively, for each subfigure. In Figure 15, we consider the case with the diffusion coefficient given by $D_0 = 0.005$ and compare the solutions with different values of k's. For the most extreme case, we only refine the solution near those locations giving the top 5-percentile FTLE values. As we can see, the solution well approximates the one by the full implementation. We have also repeated the test for a slightly larger diffusion coefficient $D_0 = 0.01$. The solutions are plotted in Figure 16.

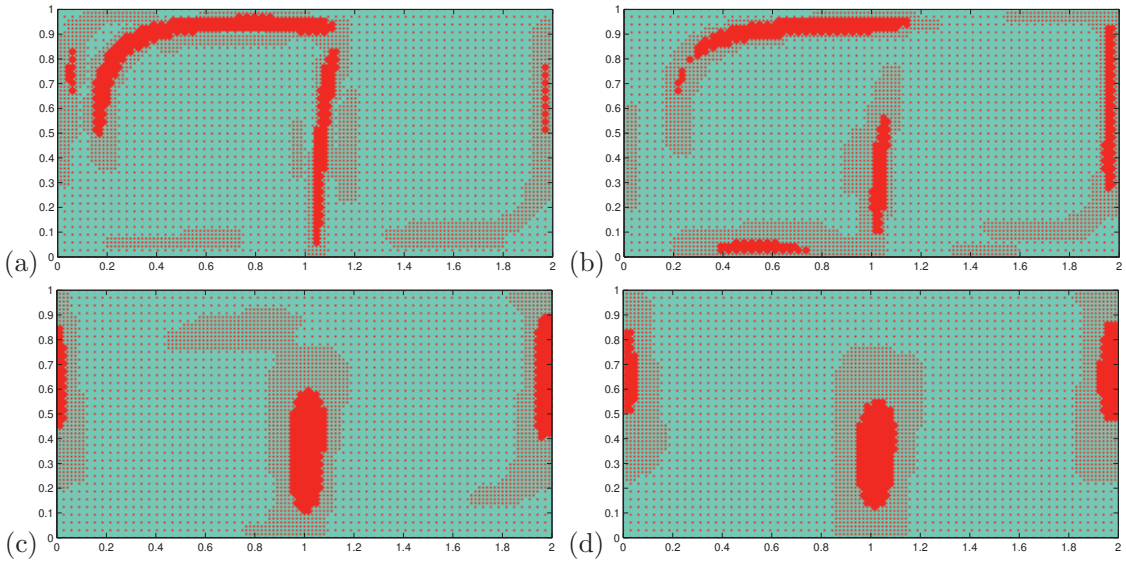

Figure 14: (Section 5.1) We plot in red those mesh points for which the FTLE is computed directly. The FTLE e_0^{10} computed using the adaptive refinement approach with $k = 20$ corresponding to (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

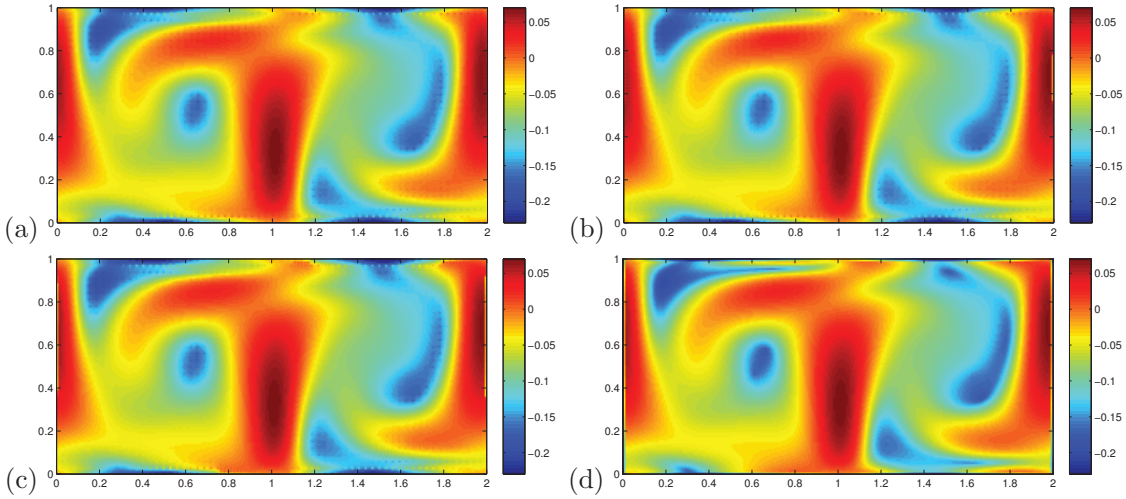

Figure 15: (Section 5.1) The FTLE e_0^{10} computed using the adaptive refinement approach with $D_0 = 0.005$ and (a) $k = 5$, (b) $k = 10$, (c) $k = 20$ (i.e. Figure 12(c)) and (d) $k = 100$ (i.e. full implementation as in Figure 9(c)), respectively.

To end this example, we briefly report the computational time of various approaches. All solutions in this work are computed using a laptop computer with a 2.6 GHz Intel core $i7$ processor. Table 1 shows the time required to compute e_0^{10} using our proposed algorithm (corresponding to Figure 5) and the MC scheme with different L's (corresponding to Figures 6, 7 and 8) for each individual D_0. We agree that it is not fair to provide any comparison on the computational efficiency of the Lagrangian and the Eulerian approaches and it is not the main purpose here. Instead, we simply want to demonstrate that the computational time of the MC approach depends heavily on both the magnitude of D_0 and also the number of trials L. As

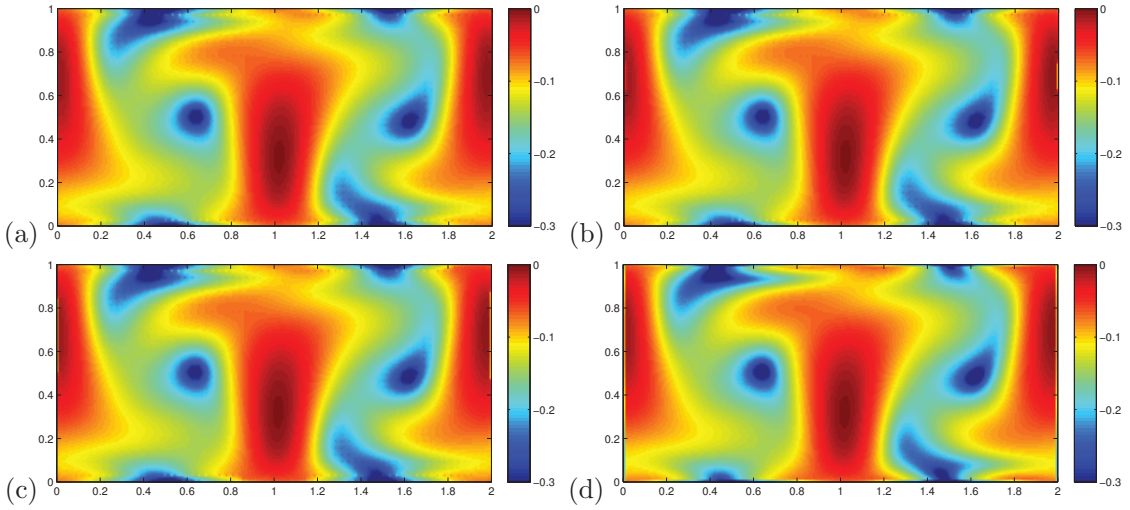

Figure 16: (Section 5.1) The FTLE e_0^{10} computed using the adaptive refinement approach with $D_0 = 0.01$ and (a) $k = 5$, (b) $k = 10$, (c) $k = 20$ (i.e. Figure 12(d)) and (d) $k = 100$ (i.e. full implementation as in Figure 9(d)), respectively.

we observed earlier, the MC scheme can already give a reasonably good solution for a relatively small diffusion coefficient D_0 (such as $D_0 = 0.0001$) using roughly 900 simulations. However, as we increase D_0, one has to significantly increase the number of trials in the MC simulation. For example, when we increase D_0 by 50 times (i.e. $D_0 = 0.005$), the solution still does not seem to converge even with ($L =$) 90000 trials in the MC simulation (Figures 6(c), 7(c) and 8(c)) which already requires over 160 times the computational time needed for the case $D_0 = 0.0001$. One has to carefully choose the number of trials to balance the computational cost and the accuracy in the solution. This observation is consistent with our claim at the end of Section 3.3.

Of course, there are various ways how one can improve the computational time and the efficiency of the MC simulation. For example, one can surely try to improve the efficiency of the SDELAB package. Since this is the most crucial component in the MC simulation, one can significantly reduce the computational time of the overall algorithm. Another way is to parallelize the computations, which can be naturally done in the Lagrangian framework by assigning either different initial conditions or different trials to different processors. However, like what we have emphasized several times in the article, we are not focusing on ways to improve the Lagrangian computations and, therefore, we will not further investigate on the SDE implementations. Finally, as mentioned in Section 3.3, if the noise along each dimension is not i.i.d., the diffusion tensor $D(\mathbf{x}, t)$ in (6) is not a scalar multiple of the identity matrix. Our proposed scheme can be easily adapted but we will leave this as future work to compare the performance with that by the MC simulations.

5.2 An application to real dataset

To demonstrate the effectiveness of our proposed algorithm, we consider the Ocean Surface Current Analyses Real-time (OSCAR) dataset in which the velocity data is only available at discrete locations. The OSCAR data was obtained from JPL Physical Oceanography DAAC and developed by ESR. It covers 0° to 360° longitude and −80° to 80° latitude. The resolution is 1/3° in each spatial direction and about 5 days in the temporal direction. We have chosen an ocean region near the Line Islands as the computational domain, which is enclosed by $S17°$ to $N8°$ latitude and $E180°$ to $E230°$ longitude. In the temporal direction, we have chosen the first 50 days in year 2015. For a better visualization, we first interpolate the velocity data to obtain

a finer resolution of $1/6^\circ$ in each spatial direction and 0.25 days in the temporal direction, which gives $\Delta x = \Delta y = 1/6$ and $\Delta t = 0.25$.

For this dataset, the velocity is available only at some uniformly sampled locations, interpolation is required at each time step for each initial particle if the MC simulation is used. This is extremely time-consuming. Instead, in this example, we only show the solutions of our Eulerian algorithm. We are not able to quantitatively compare our solution with any exact solution but can only qualitatively examine the effect of D_0 on our numerical solution. In Figure 17, we reproduce Figure 11 from [47] or Figure 7(a) from [46] showing the FTLE e_0^{50} for the noiseless velocity, i.e. $D_0 = 0$. The Eulerian approach can capture all fine features in the FTLE field. Now, Using the proposed algorithm, we compute the FTLE e_0^{50} based on the *expected* flow map $\hat{\Phi}_0^{50}$. Figure 18 shows the solutions of e_0^{50} for $D_0 = 0.0001$, 0.001, 0.005 and 0.01, respectively. As we increase the diffusion coefficient D_0, we see that the computed FTLE field is smoothened and the ridge in the FTLE diminished as expected.

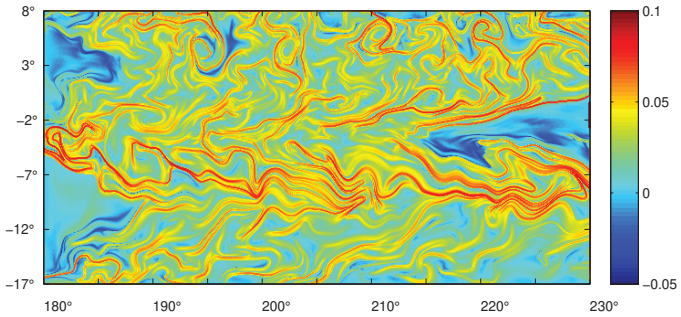

Figure 17: (Section 5.2) The FTLE e_0^{50} for the velocity without uncertainty, i.e. $D_0 = 0$, as shown in Figure 11 from [47] or Figure 7(a) from [46].

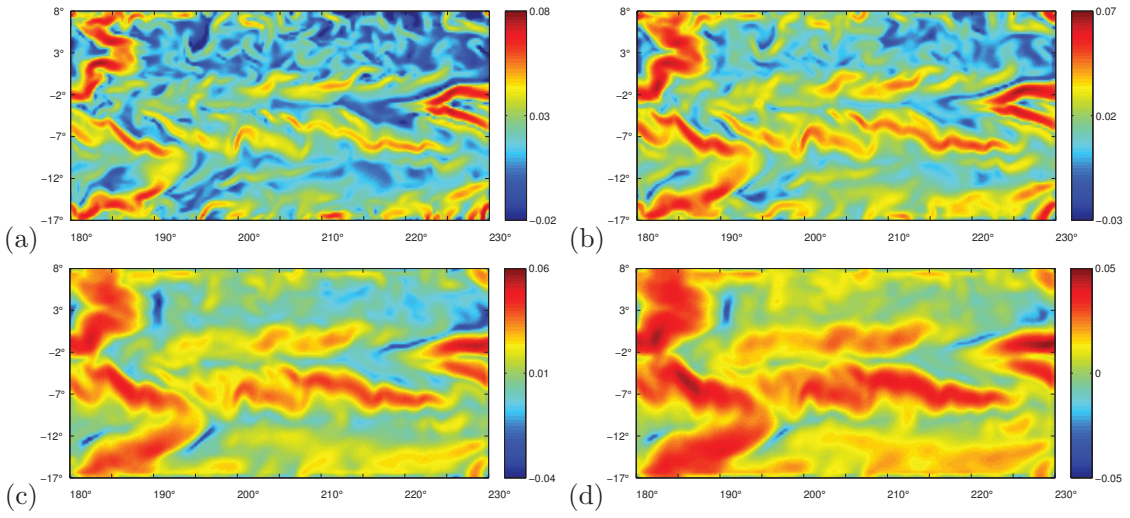

Figure 18: (Section 5.2) The FTLE e_0^{50} computed using our proposed approach corresponding to (a) $D_0 = 0.0001$, (b) $D_0 = 0.001$, (c) $D_0 = 0.005$ and (d) $D_0 = 0.01$.

Acknowledgment

The work of You was supported in part by the Natural Science Foundation of Jiangsu Province (BK20171071) and the National Natural Science Foundation of China (11701287). The second author's work was supported by the Hong Kong RGC under grant 16302819.

References

[1] M.A. Atalla. Finite-difference approximations for stochastic differential equations. *Collection of Scientific Works*, pages 11–16, 1986.

[2] R.P. Botchen, D. Weiskopf, and T. Ertl. Texture-based visualization of uncertainty in flow fields. *Proceedings of IEEE Visualization 2005*, pages 647–654, 2005.

[3] A. Denner, O. Junge, and D. Matthes. Computing coherent sets using the Fokker-Planck equation. *Journal of Computational Dynamics*, 3(2):163–177, 2016.

[4] R. Glowinski. *Splitting methods for the numerical solution of the incompressible Navier-Stokes equations*. Vistas in Applied Mathematics (A.V. Balakrishnan, A.A. Dorodnitsyn, and J.L. Lions, eds.), Optimization Softwares, New York, 1986, pp. 57-95.

[5] R. Glowinski, S. Leung, and J. Qian. A penalization-regularization-operator splitting method for eikonal based traveltime tomography. *SIAM J. Imaging Sciences*, 8(2):1263–1292, 2015.

[6] R. Glowinski, S. Leung, and J. Qian. Operator-splitting based fast sweeping methods for isotropic wave propagation in a moving fluid. *SIAM J. Sci. Comput.*, 38(2):A1195–A1223, 2016.

[7] R. Glowinski, S. Leung, and J. Qian. A simple explicit operator-splitting method for effective Hamiltonians. *SIAM J. Sc. Comput.*, 40(1):A484–A503, 2018.

[8] R Glowinski, H. Liu, S. Leung, and J. Qian. A Finite Element/Operator-Splitting Method for the Numerical Solution of the Two Dimensional Elliptic Monge-Ampere Equation. *J. Sci. Comput.*, 79(1):1–47, 2019.

[9] Roland Glowinski, Stanley J Osher, and Wotao Yin. *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2017.

[10] Tom Goldstein and Stanley Osher. The split Bregman method for L1-regularized problems. *SIAM journal on imaging sciences*, 2(2):323–343, 2009.

[11] H. Guo, W. He, T. Peterka, H.-W. Shen, S.M. Collis, and J.J. Helmus. Finite-time Lyapunov exponents and Lagrangian coherent structures in uncertain unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1672–2016, 2016.

[12] H. Guo, W. He, S. Sea, H.-W. Shen, E.M. Constantinescu, Liu C., and T. Peterka. Extreme-scale stochastic particle tracing for uncertain unsteady flow visualization and analysis. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2710–2724, 2019.

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

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

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

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

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

[18] D.J. Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM J. Numer. Anal.*, 43:525–546, 2001.

[19] M. Hlawatsch, P. Leube, W. Nowak, and D. Weiskopf. Flow radar glyphs - static visualization of unsteady flow with uncertainty. *IEEE Trans. Vis. Comput. Graph.*, 17:1949–1958, 2011.

[20] D. Karrasch and J. Keller. A geometric heat-flow theory of Lagrangian coherent structures. *arXiv preprint arXiv:1608.05598*, 2016.

[21] P.E. Kloeden. A survey of numerical methods for stochastic differential equations. *Stochastic Hydrol. Hydraul.*, 3:155–178, 1989.

[22] P.E. Kloeden and E. Platen. Numerical solution of stochastic differential equations. *Springer*, 1992.

[23] P.E. Kloeden, E. Platen, and H. Schurz. Numerical solution of sde through computer experiments. *Springer*, 1994.

[24] F. Lekien, S.C. Shadden, and J.E. Marsden. Lagrangian coherent structures in n-dimensional systems. *Journal of Mathematical Physics*, 48:065404, 2007.

[25] S. Leung. An Eulerian approach for computing the finite time Lyapunov exponent. *J. Comput. Phys.*, 230:3500–3524, 2011.

[26] S. Leung. A backward phase flow method for the finite time Lyapunov exponent. *Chaos*, 23(043132), 2013.

[27] S. Leung, G. You, T. Wong, and Y.K. Ng. Recent developments in Eulerian approaches for visualizing continuous dynamical systems. *Proceedings of the Seventh International Congress of Chinese Mathematicians*, (2):579–622, 2019.

[28] H. Liu, R Glowinski, S. Leung, and J. Qian. A Finite Element/Operator-Splitting Method for the Numerical Solution of the Three dimensional Monge-Ampere Equation. *J. Sci. Comput.*, 81(3):2271–2302, 2019.

[29] T. Lu, P. Neittaanmaki, and X.-C. Tai. A parallel splitting up method and its application to Navier-Stokes equations. *Applied Mathematics Letters*, 4:25–29, 1991.

[30] T. Lu, P. Neittaanmaki, and X.-C. Tai. A parallel splitting up method for partial differential equations and its application to Navier-Stokes equations. *RAIRO Math. Model. and Numer. Anal.*, 26:673–708, 1992.

[31] G.N. Milstein. Approximate integration of stochastic differential equations. *Theor. Prob. Appl.*, 19:557–562, 1974.

[32] Y.K. Ng and S. Leung. Estimating the Finite Time Lyapunov Exponent from Sparse Lagrangian Trajectories. *Commun. Comput. Phys.*, 26(4):1143–1177, 2019.

[33] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[34] 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.

[35] S. J. Osher and C. W. Shu. High-order Essentially NonOscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Num. Anal.*, 28:907–922, 1991.

[36] M. Otto and H. Theisel. Vortex analysis in uncertain vector fields. *Comput. Graph. Forum*, 31:1035–1044, 2012.

[37] C. Petz, K. Pothkow, and H.C. Hege. Probabilistic local features in uncertain vector fields with spatial correlation. *Comput. Graph. Forum*, 31:1045–1054, 2012.

[38] U Picchini. Sde toolbox: Simulation and estimation of stochastic differential equations with matlab. *http://sdetoolbox.sourceforge.net*, 2007.

[39] D. Schneider, J. Fuhrmann, W. Reich, and G. Scheuermann. A variance based FTLE-like method for unsteady uncertain vector fields. *Topological Methods in Data Analysis and Visualization II*, 2012.

[40] 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.

[41] G. Strang. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5:506–517, 1968.

[42] C.M. Wittenbrink, A. Pang, and S.K. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Trans. Vis. Comput. Graph.*, 2:266–279, 1996.

[43] G. You and S. Leung. An Eulerian method for computing the coherent ergodic partition of continuous dynamical systems. *J. Comp. Phys.*, 264:112–132, 2014.

[44] G. You and S. Leung. A fast semi-implicit level set method for curvature dependent flows with an application to limit cycles extraction in dynamical systems. *Commun. Comput. Phys.*, 18(1):203–229, 2015.

[45] G. You and S. Leung. An Improved Eulerian Approach for the Finite Time Lyapunov Exponent. *J. Sci. Comput.*, 76(3):1407–1435, 2018.

[46] G. You and S. Leung. Eulerian based interpolation schemes for flow map construction and line integral computation with applications to coherent structures extraction. *J. Sci. Comput.*, 74(1):70–96, 2018.

[47] G. You, T. Wong, and S. Leung. Eulerian methods for visualizing continuous dynamical systems using Lyapunov exponents. *SIAM J. on Scientific Computing*, 39(2):A415–A437, 2017.