Journal of Applied Analysis and Computation
Volume 12, Number 2, April 2022, 568–590
Website:http://www.jaac-online.com
DOI:10.11948/20210229

FAST IDENTIFICATION OF THE HYPERBOLIC LAGRANGIAN COHERENT STRUCTURES IN TWO-DIMENSIONAL FLOWS BASED ON THE EULERIAN-TYPE ALGORITHMS*

Guoqiao You[1] and Changfeng Xue[2,†]

Abstract Based on the so-called mixing-based partition, we propose an efficient Eulerian algorithm to identify the hyperbolic Lagrangian coherent structure (LCS) of any given two-dimensional (2D) flow fields, which is framework independent. To extract the required LCS, the proposed algorithm only needs to solve one single partial differential equation. Moreover, data is only required at mesh points in the implementation of the proposed algorithm. In contrast, traditional Lagrangian ray tracing approach needs to solve a system consisting of two ordinary differential equations together with a line integral along the particular particle trajectory. Furthermore, if the velocity data is only available at mesh points, the Lagrangian approach needs to implement interpolation to obtain the velocity and also the velocity gradient at non-mesh points along the particle trajectory taking off from each mesh point, which could be quite time-consuming. Based on the doubling technique, we also propose an efficient iterative Eulerian-type algorithm to identify the longtime LCS for 2D periodic flows. Numerical examples are provided to confirm the accuracy, efficiency and effectiveness of the proposed Eulerian algorithms.

Keywords Eulerian approach, partial differential equations, flow map, Lagrangian coherent structure, flow visualization.

MSC(2010) 37A25, 37M25, 76M27.

1. Introduction

It has been a long and important task to develop tools to visualize, understand and then extract useful information in various kinds of complex dynamical systems, including ocean flows [14, 29], hurricane structures [28], flight path [2, 32], gravity waves [33], blood mixing in cardiovascular flows [1] and some other chaotic systems [3, 6, 13, 20, 23, 24]. A lot of effort has been made to understand and visualize different types of behaviors in such dynamical systems, such as elliptical zones,

†The corresponding author. Email: cfxue@ycit.edu.cn(C. Xue)
[1]Department of Applied Mathematics, Nanjing Audit University, West Yushan Road, Nanjing, China
[2]School of Mathematics and Physics, Yancheng Institute of Technology, Hope Avenue Middle Road, Yancheng, China
*The authors were supported by National Natural Science Foundation of China (No. 12071409) and Natural Science Foundation of Jiangsu Province (No. BK20211293).

hyperbolic trajectories, chaotic attractors, and mixing regions. Among these, Lagrangian coherent structure (LCS) [4,11,12,29] has emerged and become a hot topic that has attracted the attention from a growing number of scientists in different research areas. Physically, LCS partitions the space-time domain into subregions based on certain quantity measured along with the passive tracer advected according to the associated dynamical system. People have been trying to develop efficient tools or algorithms to identify, extract and visualize the underlying LCS of given dynamical systems. As far, the most commonly used approach to identify LCS is based on the finite time Lyapunov exponent (FTLE) [7, 8, 12, 16, 27, 29], which 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. Following the definition of Haller [7, 9, 10], one can see that the hyperbolic LCS is closely related to the ridges of the FTLE fields. In particular, one first computes the FTLE field and then locates its ridge, the latter is then regarded as the required LCS. As a result, the computation of the FTLE has become the key component of extracting the LCS.

FTLE is obtained by firstly computing the flow map which links the initial location of a particle with the arrival position based on the characteristic line, or equivalently, the particle trajectory. Mathematically a dynamical system is modeled by the ordinary differential equation (ODE)

$$\dot{\mathbf{x}}(t) = \mathbf{u}(\mathbf{x}(t), t) \tag{1.1}$$

with the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$ and a Lipschitz velocity field $\mathbf{u} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$. 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 (1.1). The FTLE is then defined using the largest eigenvalue of the deformation matrix based on the Jacobian of this resulting flow map.

There are generally two classes of approaches for computing the FTLE and hence locating the resulting LCS of given continuous dynamical systems. Since FTLE is long treated as a Lagrangian quantity, the Lagrangian ray tracing approach is a natural choice, which solves the ODE system (1.1) using any well-developed numerical integrator. Since the timestep in these methods is only restricted by the stiffness of the ODE system, it seems that a much larger ODE timestep $\Delta t > O(\Delta x)$ could be used in some applications. However, without considering the stability condition for the ODE integrator, the numerical solution to the Lagrangian flow map may suffer from oscillatory behaviors or the numerical integrator may even lead to unreliable solutions as introduced in [18]. On the other hand, these Lagrangian approaches require the velocity data at arbitrary points in the computational domain. This implies that one has to in general implement some interpolation routines in the numerical code. Unfortunately, it could be numerically challenging to develop an interpolation approach which is computationally cheap, high order accurate yet monotone.

In a series of studies [17–19, 34–36, 40], we have developed various Eulerian approaches to numerically compute the FTLE on a fixed Cartesian mesh. The idea is to incorporate the approach with the level set method [25,26] which allows the flow map to satisfy the Liouville equations. Such hyperbolic partial differential equations (PDEs) can then be solved by any well-established robust and high order accurate numerical methods. In particular, the Eulerian methods proposed in [17, 18, 34, 36] have all suggested to reverse the time and solve the PDEs *backward* in time to

obtain the *forward* flow map and thus the *forward* FTLE. For example, to obtain the forward flow map from the initial time $t = 0$ to the final time $t = T$, one needs to solve the Liouville equations backward in time from $t = T$ to $t = 0$ with the terminal condition given at $t = T$. This implementation is numerically inconvenient in some occasions, especially when incorporating with some computational fluid dynamic (CFD) solvers, since the velocity field is loaded from the current time $t = T$ backward in time to the initial time. This implies that the whole velocity data at all time steps has to be stored in the disk which might not be practical at all. In view of this point, we have proposed and analyzed the Eulerian interpolation methods [35,40] to solve the forward flow map *on the fly* so that the PDEs are solved forward in time. Yet in some applications, a couple of flow maps might be required. It would be inefficient if we solve the PDEs all over again for every new flow map. Recently we have developed a novel Eulerian algorithm [37] to efficiently construct a lot of flow maps by pre-processing some appropriately chosen intermediate flow maps beforehand. On the other hand, noise is commonly observed in measurements from any real applications such as CFD, weather research, and aerodynamics. By extending the usual FTLE for deterministic dynamical systems [38], we have also proposed an Eulerian approach to compute the expected FTLE of uncertain flow fields.

Although FTLE on a manifold can be skillfully defined such that it is coordinate-frame independent [15], FTLE defined and computed based on a uniform Cartesian mesh is generally not objective, i.e. it is coordinate-frame dependent, which might lead to the nonobjectivity of the resulting LCS. A mixing-based partition of 2-dimensional (2D) turbulence has been proposed in [8], which is coordinate-frame invariant. The LCS extracted based on this kind of partition does not rely on the computation of FTLE and is objective. Given an incompressive Lipschitz velocity field $\mathbf{u} : \Omega \times \mathbb{R} \to \mathbb{R}^2$ with the computational domain $\Omega \subset \mathbb{R}^2$, the partition is accomplished according to the sign of

$$\varphi^{\pm} = \frac{\langle \xi^{\pm}, M\xi^{\pm} \rangle}{2|\xi^{\pm}||S\xi^{\pm}|}, \tag{1.2}$$

where $S = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$ is the rate-of-strain tensor, $M = dS/dt + 2S\nabla \mathbf{u}$ is the strain acceleration tensor and $\xi^{\pm} = (s_{22}, -s_{12} \pm |S|\sqrt{2})^T$ with s_{ij} denoting the entries of S and $|S| = \sqrt{\Sigma_{i,j} s_{ij}^2}$.

In this paper, following the methods we developed in [39], we propose an efficient Eulerian approach to identify the hyperbolic LCS based on this kind of partition. With the proposed approach, only one single PDE needs to be solved in order to obtain the required hyperbolic LCS. Furthermore, our approach only concerns data defined at mesh points and hence no interpolation is required. These advantages will reduce the overall computational cost and improve the accuracy of the corresponding algorithms, which are verified by our numerical experiments. Since sometimes the LCS over a longtime interval is required, we also propose an Eulerian algorithm to compute the longtime LCS for periodic flows based on the doubling technique developed in [18].

The outline of this paper is as follows. In Section 2 we review some previous work and build the basic concepts and formulas that will be used in later sections. In Section 3, we give the proposed Eulerian algorithms, including the Eulerian algorithm for computing the LCS and also an Eulerian algorithm for computing the

longtime LCS for periodic flows. For completeness, the computational complexity is given in this section. After that, in Section 4 we discuss the accuracy of the Eulerian algorithm for the longtime LCS construction. Finally, numerical examples are given in Section 5 to support our claims.

2. Set up

In this section, we briefly review some previous Eulerian algorithms and also build a few concepts and formulas for further development.

2.1. Flow map

As introduced in Section 1, to simplify the notation, we collect the solutions to the ODEs (1.1) for all initial conditions in Ω at all time $t \in \mathbb{R}$ and introduce the flow map

$$\Phi_a^b : \Omega \to \mathbb{R}^2$$

such that $\Phi_a^b(\mathbf{x}_0) = \mathbf{x}(b)$ represents the arrival location $\mathbf{x}(b)$ at $t = b$ of the particle trajectory satisfying the ODEs (1.1) with the initial condition $\mathbf{x}(a) = \mathbf{x}_0$ at the initial time $t = a$. This implies that the mapping will take a point from $\mathbf{x}(a)$ at $t = a$ to another point $\mathbf{x}(b)$ at $t = b$. Φ_a^b is called a *forward* flow map if $a < b$ and a *backward* flow map if $a > b$.

2.2. The Eulerian approach for computing the flow map

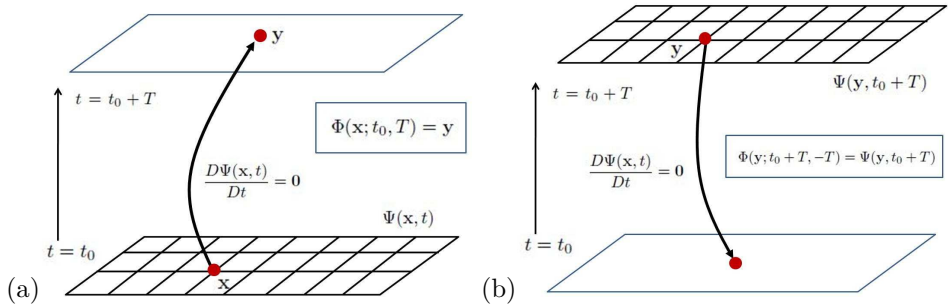


Figure 1. Lagrangian and Eulerian interpretations of the function Ψ [17]. (a) Lagrangian ray tracing from a given grid location $\mathbf{x}$ at $t = 0$. Note that $\mathbf{y}$ might be a non-grid point. (b) Eulerian values of Ψ at a given grid location $\mathbf{y}$ at $t = T$ gives the corresponding take-off location at $t = 0$. Note the take-off location might not be a mesh point.

In this subsection, we review the Eulerian formulation of the flow map computations which will be used in the next section when we propose our Eulerian algorithms for identifying the LCS. We follow the idea in [17, 18, 34] and define a vector-valued function $\Psi = (\Psi^1, \Psi^2, \cdots, \Psi^d) : \Omega \times \mathbb{R} \to \mathbb{R}^2$. At $t = 0$, we initialize these functions by

$$\Psi(\mathbf{x}, 0) = \mathbf{x} = (x^1, x^2, \cdots, x^d). \tag{2.1}$$

These functions provide a labeling for any particle in the phase space at t = 0. In particular, any particle initially located at $(\mathbf{x}, t) = (\mathbf{x}_0, 0) = (x_0^1, x_0^2, \cdots, x_0^d, 0)$ 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 $\mathbb{R}^d$. 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,

$$\frac{\partial\Psi(\mathbf{x},t)}{\partial t} + (\mathbf{u}\cdot\nabla)\Psi(\mathbf{x},t) = \mathbf{0} \tag{2.2}$$

with the initial condition (2.1).

The above **implicit** representation 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. Furthermore, the forward flow map at a grid location $\mathbf{x} = \mathbf{x}_0$ from $t = 0$ to $t = T$ is given by $\Phi_0^T(\mathbf{x}_0) = \mathbf{y}$ where $\mathbf{y}$ satisfies $\Psi(\mathbf{y}, 0+T) = \Psi(\mathbf{x}_0, 0) \equiv \mathbf{x}_0$. Note that, in general, $\mathbf{y}$ is a non-mesh location. The typical two dimensional scenario is illustrated in Figure 1(a).

The solution to (2.2) contains much more information than what was referred to above. Consider a given mesh location $\mathbf{y}$ in the phase space at the time $t = T$, as shown in Figure 1(b), i.e. $(\mathbf{y}, T)$ in the extended phase space. As discussed in our previous work, these level set functions $\Psi(\mathbf{y}, T)$ 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, the solution to the level set equations (2.2) for $t \in (0, T)$ provides also backward flow maps for all intermediate times, i.e. $\Phi_t^0(\mathbf{y}) = \Psi(\mathbf{y}, t)$.

To compute the forward flow map, on the other hand, [17] has proposed to simply reverse the above process by initializing the level set functions at $t = T$ by $\Psi(\mathbf{x}, T) = \mathbf{x}$ and solving the corresponding level set equations (2.2) backward in time.

2.3. The doubling technique to compute the longtime flow map

In [18], an efficient method has been proposed to compute the longtime FTLE for periodic flows. The idea is to develop a map doubling phase flow method for longtime flow map computations. To compute the longtime backward flow map, for example, we first construct the solution $\Psi(\mathbf{x}, T_m)$ by solving the Liouville equations (2.2) forward in time from $t = 0$ to $t = T_m$ where T_m is the period of the flow. To determine $\Psi(\mathbf{x}, 2T_m)$, we use the phase flow property and obtain $\Psi(\mathbf{x}, 2T_m) = \Psi(\Psi(\mathbf{x}, T_m), T_m)$.

In general, once we have obtained the solution $\Psi(\mathbf{x}, 2^{k-1}T_m)$, we can obtain

$$\Psi(\mathbf{x}, 2^k T_m) = \Psi(\Psi(\mathbf{x}, 2^{k-1}T_m), 2^{k-1}T_m).$$

Finally, if we take $T = 2^n T_m$, the *backward* flow map from $t = T$ to $t = 0$ is given by $\Phi_T^0(\mathbf{x}) = \Psi(\mathbf{x}, 2^n T_m)$.

The idea to compute the *forward* flow map is simple. We can solve the Liouville equation *backward* in time from $t = T$ to $t = T - T_m$. Then we iterate the map

n-times to get the overall flow map *forward* in time from $t = 0$ to $t = T = T_m \cdot 2^n$. Once the longtime flow map is computed, the corresponding Jacobian can be easily obtained by any finite difference method.

2.4. The Lagrangian approach for identifying the LCS upon the mixing-based partition

The mixing-based partition is done according to the sign of

$$\varphi^{\pm} = \frac{\langle \xi^{\pm}, M\xi^{\pm} \rangle}{2|\xi^{\pm}||S\xi^{\pm}|}.$$

Concretely speaking, at any time t the computational domain Ω can be uniquely partitioned into three regions: the elliptic region $\mathcal{E}(t)$ consisting of points $\mathbf{x}$ where $min\{\varphi^+(\mathbf{x},t), \varphi^-(\mathbf{x},t)\} < 0$ or $S(\mathbf{x},t) = O_{2\times 2}$, the parabolic region $\mathcal{P}(t)$ consisting of points $\mathbf{x}$ where $min\{\varphi^+(\mathbf{x},t), \varphi^-(\mathbf{x},t)\} = 0$ and the hyperbolic region $\mathcal{H}(t)$ consisting of points $\mathbf{x}$ where $min\{\varphi^+(\mathbf{x},t), \varphi^-(\mathbf{x},t)\} > 0$. Once the partition of the domain Ω is known at any $t \in \mathcal{I}$, we are able to identify the underlying hyperbolic LCS, where $\mathcal{I}$ is the time span of the velocity field and we take $\mathcal{I} = [0, T]$ for simplicity hereafter. As introduced in [8], the hyperbolic repelling LCS can be defined as material lines along which the time integral of the local flux is maximal. Mathematically, hyperbolic LCS at the particular time level $t = 0$ can be located by looking for local maximum curves of the scalar field

$$\sigma_0^T(\mathbf{x}) = \int_{\{t\in[0,T]|\mathbf{x}(t)\in\mathcal{H}(t)\}} |S(\mathbf{x}(t),t)|dt\,, \tag{2.3}$$

which indicates the total flux along the particle trajectory $\mathbf{x}(t)$ taking off from $\mathbf{x}$ at the initial time $t = 0$.

Since the quantity σ_0^T is defined as a Lagrangian-type quantity, one natural approach to compute σ_0^T is the Lagrangian ray tracing method [8]. In particular, one first solves the ODE system (1.1), given the initial condition $\mathbf{x}(0) = \mathbf{x}_0$, to obtain the particle trajectory $\mathbf{x}(t)$ starting from the location $\mathbf{x}_0$ at the initial time $t = 0$. After that, a standard numerical integrator is used to approximate the integral (2.3) to obtain the value of $\sigma_0^T(\mathbf{x})$ at the location $\mathbf{x}_0$, i.e. $\sigma_0^T(\mathbf{x}_0)$. In the implementation, one has to continuously monitor the particle position at any particular time level $t \in [0, T]$, to see if $\mathbf{x}(t)$ belongs to $\mathcal{H}(t)$ or not. Another issue is, when the velocity data is only available at mesh points, the implementation of the Lagrangian approach will involve several factors which could be very time-consuming and even affect the accuracy of the results. Except for the numerical integration, these factors also include interpolation at mesh points to obtain velocity and velocity gradient data at non-mesh points. All these factors have to be treated with high-order numerical methods, otherwise small LCS structures might not be extracted from the computed σ_0^T field. Finally, with the concept of flow map, formula(2.3) can be rewritten as

$$\sigma_0^T(\mathbf{x}) = \int_{\{t\in[0,T]|\Phi_0^t(\mathbf{x})\in\mathcal{H}(t)\}} |S(\Phi_0^t(\mathbf{x}),t)|dt\,. \tag{2.4}$$

3. The Eulerian approach for identifying the hyperbolic LCS

As discussed above, the Lagrangian approach has several disadvantages in computing the σ_0^T field and hence identifying the hyperbolic LCS. Especially when the velocity data is only available at mesh points, one needs to first use the finite difference method to obtain the velocity gradient data at mesh points and then implement interpolation based on the obtained discrete velocity gradient data and also on the discrete velocity data defined at mesh points, which could be quite time-consuming and not that accurate. In this section, we will first propose an efficient Eulerian algorithm to compute the σ_0^T field which requires no interpolation. Furthermore, only one single PDE needs to be solved in the proposed algorithm. Based on the doubling technique [18], we also propose an Eulerian algorithm to compute the longtime σ_0^T field for periodic flows, in order to locate the longtime LCS. At the end of this section, we will briefly analyze the computational complexity of the proposed Eulerian algorithm.

3.1. An Eulerian algorithm for computing the σ_0^T field

We are proposing, in this subsection, an Eulerian algorithm to compute the σ_0^T field which does not require interpolation on the discrete velocity gradient data nor the discrete velocity data. Suppose that $[0,T]$ is the interval we focus on and therefore, the $\sigma_0^T(\mathbf{x})$ field is what we need to compute in order to identify the underlying hyperbolic LCS.

At first, we rewrite (2.4) in a more compact form

$$\sigma_0^T(\mathbf{x}) = \int_0^T Q(\Phi_0^t(\mathbf{x}), t)dt \tag{3.1}$$

where $Q(\mathbf{x},t) \triangleq |S(\mathbf{x},t)| \cdot \chi_{\mathcal{H}(t)}(\mathbf{x})$ with the indicator function

$$\chi_{\mathcal{H}(t)}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \mathcal{H}(t), \\ 0, & \text{otherwise.} \end{cases}$$

Then we define a real-valued function $F(\mathbf{x},t) : \Omega \times [0,T] \to \mathbb{R}$ such that $F(\mathbf{x},t) = \sigma_t^T(\mathbf{x})$, i.e.

$$F(\mathbf{x},t) \triangleq \int_t^T Q(\Phi_t^\tau(\mathbf{x}), \tau)d\tau\,, \tag{3.2}$$

which measures the time integral of the local flux along the trajectory of the particle starting from the point $\mathbf{x}$ at the time level t and arriving at the time level T. It is obvious that $F(\mathbf{x},T) = 0$ and $F(\mathbf{x},0) = \sigma_0^T(\mathbf{x})$ for any $\mathbf{x} \in \Omega$. Furthermore, $F(\mathbf{x},t)$ decreases along any particle trajectory. Indeed, we have the following result.

Lemma 3.1. *Let $F(\mathbf{x},t)$ be defined as in (3.2), then the material derivative of $F(\mathbf{x},t)$ along any particle trajectory is $-Q(\mathbf{x},t)$, i.e. $DF(\mathbf{x},t)/Dt = -Q(\mathbf{x},t)$, where $D(\cdot)/Dt$ denotes the material derivative.*

Proof. We still use $\mathbf{x}(t)$ to denote the particle trajectory and suppose that the particle is located at $\mathbf{x}_0$ at the initial time $t = 0$. As a result, we have $\mathbf{x}(t) = \Phi_0^t(\mathbf{x}_0)$.

Then we only need to prove that $dF(\mathbf{x}(t),t)/dt = -Q(\mathbf{x}(t),t)$. In fact,

$$F(\mathbf{x}(t),t) = \int_t^T Q(\Phi_t^\tau(\mathbf{x}(t)),\tau)d\tau = \int_t^T Q(\Phi_t^\tau(\Phi_0^t(\mathbf{x}_0)),\tau)d\tau = \int_t^T Q(\Phi_0^\tau(\mathbf{x}_0),\tau)d\tau.$$

Then by the derivative rule of integral with parameters, we have $dF(\mathbf{x}(t),t)/dt = -Q(\Phi_0^t(\mathbf{x}_0),t) = -Q(\mathbf{x}(t),t)$. The proof completes. □

According to Lemma 1 and recalling that $D/Dt = \partial/\partial t + \mathbf{u}\cdot\nabla$, we immediately have

$$\frac{\partial F(\mathbf{x},t)}{\partial t} + (\mathbf{u}\cdot\nabla)F(\mathbf{x},t) = -Q(\mathbf{x},t)\,. \tag{3.3}$$

To obtain the σ_0^T field, we first solve the PDE (3.3) *backward* in time from $t = T$ to $t = 0$ with the terminal condition $F(\mathbf{x},T) = 0$ and then assign $\sigma_0^T(\mathbf{x}) = F(\mathbf{x},0)$. In the whole process, the computation of the right hand side of PDE (3.3), i.e. $-Q(\mathbf{x},t)$, is only implemented at mesh points. Compared to the traditional Lagrangian ray tracing approach, the proposed Eulerian approach has several advantages. Firstly, only one single PDE (3.3) needs to be solved to obtain the σ_0^T field while the Lagrangian approach needs to compute the integral (2.3) together with the ODE system (1.1) consisting of two ODEs for a 2D dynamical system. Secondly, if the velocity data is only available at mesh points, e.g. from certain CFD solver, no interpolation is required in our Eulerian approach since all computations are done only on mesh. The finite difference step involved in the computation can be easily accomplished using any well-developed approach such as ENO or WENO [21, 30]. After the $\sigma_0^T(\mathbf{x})$ field is obtained, the LCS is identified at $t = 0$ by locating the local maximum curves.

Here we also want to emphasize that it is not necessary to compute and store $\mathcal{H}(t)$ for any $t \in [0,T]$ beforehand. In the implementation, we propose to compute $\mathcal{H}(t)$ and $Q(\mathbf{x},t)$ simultaneously while solving the PDE (3.3). For example, at a particular time step $t = t_k$, based on the velocity data $\mathbf{u}(\mathbf{x}_{ij},t_k)$ at each mesh point $\mathbf{x}_{ij}$, we use the finite difference scheme to obtain the velocity gradient $\nabla\mathbf{u}(\mathbf{x}_{ij},t_k)$ and subsequently the rate-of-strain tensor $S(\mathbf{x}_{ij},t_k)$ and the vectors $\xi^{\pm}(\mathbf{x}_{i,j},t_k)$ on the mesh. After that, based on the velocity data $\mathbf{u}(\mathbf{x}_{ij},t_{k+1})$ and $\mathbf{u}(\mathbf{x}_{ij},t_{k-1})$ at the two time levels $t = t_{k+1}$ and $t = t_{k-1}$, we approximate the temporal variation of the velocity gradient $\frac{\partial\nabla\mathbf{u}}{\partial t}|_{(\mathbf{x}_{ij},t_k)}$ and $\frac{\partial S}{\partial t}|_{(\mathbf{x}_{ij},t_k)}$ is immediately obtained. Then $dS/dt = \partial S/\partial t + (\mathbf{u}\cdot\nabla)S$ can be obtained by implementing the finite difference one more time on $S(\mathbf{x}_{ij},t_k)$, which subsequently gives the value of the strain acceleration tensor $M = dS/dt + 2S\nabla\mathbf{u}$ at each mesh point $\mathbf{x}_{ij}$. Then $\varphi^{\pm}(\mathbf{x}_{ij},t_k)$ can be computed using formula (1.2), from which we can obtain the partition of the domain Ω at $t = t_k$, including the required $\mathcal{H}(t_k)$. Finally, $Q(\mathbf{x}_{ij},t_k)$ is computed by $Q(\mathbf{x}_{ij},t_k) = |S(\mathbf{x}_{ij},t_k)|\cdot\chi_{\mathcal{H}(t_k)}(\mathbf{x}_{ij})$.

We summarize the proposed Eulerian approach in Algorithm 1:

Algorithm 1 (An Eulerian approach for computing the $\sigma_0^T(\mathbf{x}_{ij})$ field):

1. Discretize the computational domain

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

$$t_k = t_0 + k\Delta t, \quad \Delta t = \frac{T}{K}, \quad k = 0, 1, ..., K\,.$$

2. Initialize the function $F(\mathbf{x}_{ij}, T) = 0$ with $\mathbf{x}_{ij} = (x_i, y_j)$.
3. Solve PDE (3.3), using any well-developed numerical method like WENO5-TVDRK2 [5,22,30], *backward* from time $t = T$ to the initial time $t = 0$ to obtain $F(\mathbf{x}_{ij}, 0)$.
4. Assign $\sigma_0^T(\mathbf{x}_{ij}) = F(\mathbf{x}_{ij}, 0)$.

3.2. An efficient Eulerian algorithm to compute the longtime σ_0^T field

In [18], an efficient method has been proposed to compute the longtime flow map for periodic flows based on the map doubling phase flow method, as introduced in Section 2.3. Here we take a similar idea and propose an efficient Eulerian algorithm to compute the longtime σ_0^T field for periodic flows based on its additivity. In particular, we have the following result.

Theorem 3.1. *Suppose* $\sigma_{t_0}^{t_1}(\mathbf{x}) = \int_{t_0}^{t_1} Q(\Phi_{t_0}^{\tau}(\mathbf{x}), \tau)d\tau$ *as defined in (3.1), then*

$$\sigma_0^t(\mathbf{x}) = \sigma_s^t(\Phi_0^s(\mathbf{x})) + \sigma_0^s(\mathbf{x}), \ \ \forall s \geq 0, \ t \geq s\,. \tag{3.4}$$

Proof. According to the definition of the flow map, we have $\Phi_s^{\tau}(\Phi_0^s(\mathbf{x})) = \Phi_0^{\tau}(\mathbf{x})$ and therefore,

$$\sigma_s^t(\Phi_0^s(\mathbf{x})) = \int_s^t Q(\Phi_s^{\tau}(\Phi_0^s(\mathbf{x})), \tau)d\tau = \int_s^t Q(\Phi_0^{\tau}(\mathbf{x}), \tau)d\tau\,.$$

As a result,

$$\begin{aligned}\sigma_s^t(\Phi_0^s(\mathbf{x})) + \sigma_0^s(\mathbf{x}) &= \int_s^t Q(\Phi_0^{\tau}(\mathbf{x}), \tau)d\tau + \int_0^s Q(\Phi_0^{\tau}(\mathbf{x}), \tau)d\tau \\ &= \int_0^t Q(\Phi_0^{\tau}(\mathbf{x}), \tau)d\tau = \sigma_0^t(\mathbf{x})\,.\end{aligned}$$

□

In the implementation, we first solve PDEs (3.3) and (2.2) *backward* from $t = T_m$ to $t = 0$ with the terminal conditions $F(\mathbf{x}_{ij}, T_m) = 0$ and $\Psi(\mathbf{x}_{ij}, T_m) = \mathbf{x}_{ij}$, respectively, where T_m is the period of the flow. Then we have the $\sigma_0^{T_m}$ field $\sigma_0^{T_m}(\mathbf{x}_{ij}) = F(\mathbf{x}_{ij}, 0)$ and the *forward* flow map $\Phi_0^{T_m}(\mathbf{x}_{ij}) = \Psi(\mathbf{x}_{ij}, 0)$. First, due to the periodicity, we have $\Phi_0^{2T_m}(\mathbf{x}_{ij}) = \Phi_{T_m}^{2T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij})) = \Phi_0^{T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))$, which can be obtained by interpolation on $\Phi_0^{T_m}(\mathbf{x}_{ij})$. By formula (3.4), we have $\sigma_0^{2T_m}(\mathbf{x}_{ij}) = \sigma_0^{T_m}(\mathbf{x}_{ij}) + \sigma_{T_m}^{2T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))$. Also by the periodicity, we have $\sigma_{T_m}^{2T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij})) = \sigma_0^{T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))$, which can be obtained by interpolation on $\sigma_0^{T_m}(\mathbf{x}_{ij})$, and then $\sigma_0^{2T_m}(\mathbf{x}_{ij})$ can be obtained.

In general, once $\sigma_0^{2^{k-1}T_m}(\mathbf{x}_{ij})$ and $\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij})$ are known at mesh points $\mathbf{x}_{ij}$ where $k \in \mathbb{N}_+$, we can compute

$$\Phi_0^{2^kT_m}(\mathbf{x}_{ij}) = \Phi_{2^{k-1}T_m}^{2^kT_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij})) = \Phi_0^{2^{k-1}T_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij}))$$

and also $\sigma_0^{2^k T_m}(\mathbf{x}_{ij}) = \sigma_0^{2^{k-1}T_m}(\mathbf{x}_{ij}) + \sigma_{2^{k-1}T_m}^{2^k T_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij})) = \sigma_0^{2^{k-1}T_m}(\mathbf{x}_{ij}) + \sigma_0^{2^{k-1}T_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij}))$, where $\Phi_0^{2^{k-1}T_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij}))$ and $\sigma_0^{2^{k-1}T_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij}))$ can be obtained by interpolation on $\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij})$ and $\sigma_0^{2^{k-1}T_m}(\mathbf{x}_{ij})$, respectively. In summary, if we take $T = 2^n T_m$, the σ_0^T field can be computed as $\sigma_0^T(\mathbf{x}_{ij}) = \sigma_0^{2^n T_m}(\mathbf{x}_{ij})$ where we only need to iteratively interpolate n times upon $\Phi_0^{T_m}(\mathbf{x}_{ij})$ and $\sigma_0^{T_m}(\mathbf{x}_{ij})$. The corresponding Eulerian algorithm for 2D periodic flows is summarized in Algorithm 2.

Algorithm 2 (An efficient Eulerian algorithm for computing the σ_0^T field for 2D periodic flows where $T = 2^n T_m$):

1. Discretize the computational domain as in Algorithm 1 to get x_i, y_j, t_k.
2. Initialize the level set functions on the time level $t = T_m$
$$\Psi(\mathbf{x}_{ij}, T_m) = \mathbf{x}_{ij}\,,$$
$$F(\mathbf{x}_{ij}, T_m) = 0\,.$$
3. Solve PDEs (2.2) and (3.3) *backward* together from $t = T_m$ to $t = 0$ and then assign $\Phi_0^{T_m}(\mathbf{x}_{ij}) = \Psi(\mathbf{x}_{ij}, 0)$ and $\sigma_0^{T_m}(\mathbf{x}_{ij}) = F(\mathbf{x}_{ij}, 0)$, respectively.
4. For $k = 1, 2, \cdots, n$, interpolate to obtain
$$\Phi_0^{2^k T_m}(\mathbf{x}_{ij}) = \Phi_0^{2^{k-1}T_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij}))$$
and
$$\sigma_0^{2^k T_m}(\mathbf{x}_{ij}) = \sigma_0^{2^{k-1}T_m}(\mathbf{x}_{ij}) + \sigma_0^{2^{k-1}T_m}(\Phi_0^{2^{k-1}T_m}(\mathbf{x}_{ij}))\,.$$
5. Assign $\sigma_0^T(\mathbf{x}_{ij}) = \sigma_0^{2^n T_m}(\mathbf{x}_{ij})$.

3.3. Computational complexity

To end this section, we consider the computational complexity of the proposed Algorithm 1. Let I and K be the number of mesh points in one spatial direction and the temporal direction, respectively. In Algorithm 1, only PDE (3.3) needs to be solved. The discretization of the PDE takes $O(I^2)$ operations at each time step. Summing up the procedures at all time steps, the overall computational complexity comes to $O(KI^2)$ which is optimal in the sense that each grid point is visited for only $O(1)$ time.

4. Accuracy analysis on the longtime LCS identification

For periodic dynamical systems, we have proposed an efficient Eulerian algorithm, i.e. Algorithm 2, to compute the longtime σ_0^T field and thus identify the longtime hyperbolic LCS in Section 3.2. In particular, the interpolation scheme is iteratively implemented to obtain the longtime flow map and the longtime σ_0^T field. Now we are analyzing the accuracy of this algorithm in this section. We first list the following three lemmas, which will be used for further development. The first two lemmas are exactly Lemma 3 and Lemma 5 from [35] and the third one is a rewritten version of Theorem 1 from [36].

Lemma 4.1. *Suppose that the velocity field* $\mathbf{u}(\mathbf{x},t)$ *is smooth enough and has the Lipschitz constant* L *on the computational domain* Ω*. Then for any spatial variable* x_i*, we have*

$$\left|\frac{\partial \Phi_0^t(\mathbf{x})}{\partial x_i}\right| \le e^{Lt}$$

for any $t>0$*.*

Lemma 4.2. *Suppose that the velocity field* $\mathbf{u}$ *is smooth enough. For each* $s\ge 2$*, there exists a constant* C_s *such that for any multi-index* γ *with* $|\gamma|=s$ *and any* $\mathbf{x}\in\Omega$*, we have*

$$|\partial^\gamma \Phi_0^t(\mathbf{x})| \le C_s\, t\, e^{(2s-1)Lt},\ \ \forall t>0\,.$$

Lemma 4.3. *Assuming that* $\Phi_0^{T_m}(\mathbf{x}_{ij})$ *computed in step 3 of Algorithm 2 has second order accuracy and the interpolation scheme in step 4 is at least second order accurate,* $\Phi_0^{2T_m}(\mathbf{x}_{ij})$ *is also second order accurate.*

Formula (3.1) can be equivalently rewritten as $\sigma_0^t(\mathbf{x}) = \int_0^t Q(\Phi_0^\tau(\mathbf{x}),\tau)d\tau$. Taking the derivative with respect to t gives

$$\frac{d}{dt}\sigma_0^t(\mathbf{x}) = Q(\Phi_0^t(\mathbf{x}),t)\,. \tag{4.1}$$

In the rest of this section, we suppose that $\mathbf{u}(\mathbf{x},t)$ and $Q(\mathbf{x},t)$ are smooth enough and have the Lipschitz constants L and M, respectively, on the computational domain Ω. Then we give the following two preliminary results.

Lemma 4.4. *For any spatial variable* x_i*, we have*

$$\left|\frac{\partial \sigma_0^t(\mathbf{x})}{\partial x_i}\right| < \frac{M}{L}e^{Lt}$$

for any $t>0$*.*

Proof. Taking the partial derivative of both sides of (4.1) with respect to x_i, we have

$$\frac{d}{dt}\frac{\partial \sigma_0^t(\mathbf{x})}{\partial x_i} = \frac{\partial \Phi_0^t(\mathbf{x})}{\partial x_i}\cdot\nabla Q\,.$$

Then

$$\frac{d}{dt}\left|\frac{\partial \sigma_0^t(\mathbf{x})}{\partial x_i}\right| \le \left|\frac{d}{dt}\frac{\partial \sigma_0^t(\mathbf{x})}{\partial x_i}\right| = \left|\frac{\partial \Phi_0^t(\mathbf{x})}{\partial x_i}\cdot\nabla Q\right| \le Me^{Lt} = \frac{d}{dt}\left(\frac{M}{L}e^{Lt}\right)$$

where the first inequality is a direct corollary of the triangle inequality and the second inequality is due to Lemma 4.1. It is equivalent to $\frac{d}{dt}\left(\left|\frac{\partial \sigma_0^t(\mathbf{x})}{\partial x_i}\right| - \frac{M}{L}e^{Lt}\right)\le 0$, which gives $\left|\frac{\partial \sigma_0^t(\mathbf{x})}{\partial x_i}\right| - \frac{M}{L}e^{Lt} \le \left|\frac{\partial \sigma_0^0(\mathbf{x})}{\partial x_i}\right| - \frac{M}{L}e^0 = -\frac{M}{L}$, for any $t>0$, i.e. $\left|\frac{\partial \sigma_0^t(\mathbf{x})}{\partial x_i}\right| \le \frac{M}{L}(e^{Lt}-1) < \frac{M}{L}e^{Lt}$. □

Lemma 4.5. *For each* $s\ge 2$*, there exists a constant* D_s *such that for any multi-index* γ *with* $|\gamma|=s$ *and any* $\mathbf{x}\in\Omega$*, we have*

$$|\partial^\gamma \sigma_0^t(\mathbf{x})| \le D_s\, e^{2s^2Lt},\ \ \forall t>0\,.$$

Proof. For any fixed γ satisfying $|\gamma| = s \geq 2$, differentiating both sides of formula (4.1) gives

$$\frac{d\partial^\gamma \sigma_0^t}{dt} = \sum_{p=1}^{s} \sum_{(\gamma_1,\cdots,\gamma_p):\gamma=\sum_{l=1}^{p}\gamma_l} \left(\sum_{k_1,\cdots,k_p=1}^{d} Q_{k_1\cdots k_p} \prod_{j=1}^{p} \partial^{\gamma_j}\phi^{k_j} \right),$$

with $\partial^\gamma \sigma_0^t|_{t=0} = 0$, where ϕ^{k_j} denotes the k_j-th component of $\Phi_0^t(\mathbf{x})$ and $Q_{k_1\cdots k_p} \triangleq \frac{\partial^p Q}{\partial x_{k_1}\cdots\partial x_{k_p}}$. The second summation is over all different choices of $(\gamma_1,\cdots,\gamma_p)$ such that $\gamma = \gamma_1 + \cdots + \gamma_p$ and $|\gamma_l| > 0$.

If $|\gamma_j| = 1$, we have $|\partial^{\gamma_j}\phi^{k_j}| \leq e^{Lt} \leq e^{2|\gamma_j|Lt}$ by Lemma 4.1. If $|\gamma_j| \geq 2$, according to Lemma 4.2, we have $|\partial^{\gamma_j}\phi^{k_j}| \leq C_{|\gamma_j|} t e^{(2|\gamma_j|-1)Lt}$. When $t > 0$, the maximum of the function te^{-Lt} is achieved at $t = 1/L$ and therefore, $te^{-Lt} \leq 1/(Le)$. As a result, $|\partial^{\gamma_j}\phi^{k_j}| \leq \frac{C_{|\gamma_j|}}{Le} e^{2|\gamma_j|Lt}$ for $|\gamma_j| \geq 2$. In summary, $|\partial^{\gamma_j}\phi^{k_j}| \leq Ce^{2|\gamma_j|Lt}$ for any γ_j where $C = \max\{1, C_2/(Le), \cdots, C_s/(Le)\}$. Subsequently we have $|\prod_{j=1}^{p} \partial^{\gamma_j}\phi^{k_j}| \leq C^p\, e^{2s^2Lt}$ and $\frac{d|\partial^\gamma\sigma_0^t|}{dt} \leq \left|\frac{d\partial^\gamma\sigma_0^t}{dt}\right| \leq C'\, e^{2s^2Lt} = \frac{d}{dt} D_s e^{2s^2Lt}$ where $D_s \triangleq \frac{C'}{2s^2L}$, C' depends on s and the norms of partial derivatives of Q. Therefore, $\frac{d}{dt}\left(|\partial^\gamma\sigma_0^t| - D_s e^{2s^2Lt}\right) \leq 0$, which gives $|\partial^\gamma\sigma_0^t| - D_s e^{2s^2Lt} \leq -D_s$ for any $t \geq 0$. That is, $|\partial^\gamma\sigma_0^t| \leq D_s(e^{2s^2Lt} - 1) \leq D_s e^{2s^2Lt}$. □

Now we are ready to state our main results about the accuracy of Algorithm 2. In the rest of this section, we assume that the interpolation operator has a bounded norm which is independent of the mesh size.

Theorem 4.1. *Assuming that $\Phi_0^{T_m}(\mathbf{x}_{ij})$ and $\sigma_0^{T_m}(\mathbf{x}_{ij})$ computed in step 3 of Algorithm 2 both have second order accuracy and the interpolation scheme in step 4 is at least second order accurate, $\sigma_0^{2T_m}(\mathbf{x}_{ij})$ is also second order accurate.*

Proof. According to formula (3.4), we have

$$\sigma_0^{2T_m}(\mathbf{x}_{ij}) = \sigma_0^{T_m}(\mathbf{x}_{ij}) + \sigma_{T_m}^{2T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))\,.$$

Since $\sigma_0^{T_m}(\mathbf{x}_{ij})$ is second order accurate, we only need to prove that $\sigma_{T_m}^{2T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij})) = \sigma_0^{T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))$ is also second order accurate.

Using $\tilde{\cdot}$ to denote the numerical solution and $\mathcal{I}$ to denote the interpolation operator, $\sigma_0^{T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))$ is then approximated by $\mathcal{I}\tilde{\sigma}_0^{T_m}(\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij}))$ with the error

$$\begin{aligned}
|\mathcal{I}\tilde{\sigma}_0^{T_m}(\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij})) - \sigma_0^{T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))| \leq & |\mathcal{I}\tilde{\sigma}_0^{T_m}(\tilde{\Phi}_0^{T_m}(\mathbf{x}_{i,j})) - \mathcal{I}\sigma_0^{T_m}(\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij}))| \\
& + |\mathcal{I}\sigma_0^{T_m}(\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij})) - \sigma_0^{T_m}(\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij}))| \\
& + |\sigma_0^{T_m}(\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij})) - \sigma_0^{T_m}(\Phi_0^{T_m}(\mathbf{x}_{ij}))| \\
\triangleq & I_1 + I_2 + I_3\,.
\end{aligned}$$

Due to the second order accuracy of $\sigma_0^{T_m}(\mathbf{x}_{ij})$, we can find a constant C_0 such that

$$|\tilde{\sigma}_0^{T_m}(\mathbf{x}_{ij}) - \sigma_0^{T_m}(\mathbf{x}_{ij})| \leq C_0\Delta x^2.$$

Suppose N_I is the norm of the interpolation operator which is Δx-independent, then I_1 is bounded by

$$I_1 \leq N_I \cdot \max_{\mathbf{x}_{ij}} |\tilde{\sigma}_0^{T_m}(\mathbf{x}_{ij}) - \sigma_0^{T_m}(\mathbf{x}_{ij})| \leq C_1\Delta x^2.$$

Since the interpolation scheme is at least second order accurate, I_2 is bounded by

$$I_2 \leq C_2' \Delta x^2 \max_{|\gamma|=2} \sup_{\mathbf{x}_{ij}} |\partial^\gamma \sigma_0^{T_m}(\mathbf{x}_{ij})|.$$

According to Lemma 4.5, there exists a constant D_2 such that

$$\max_{|\gamma|=2} \sup_{\mathbf{x}_{ij}} |\partial^\gamma \sigma_0^{T_m}(\mathbf{x}_{ij})| \leq D_2\, e^{8LT_m}$$

which implies $I_2 \leq C_2 \Delta x^2$ where C_2 is a constant. Finally, I_3 is bounded by

$$I_3 \leq \sup_{\mathbf{x}\in\Omega} |\nabla \sigma_0^{T_m}(\mathbf{x})| |\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij}) - \Phi_0^{T_m}(\mathbf{x}_{ij})|\,.$$

Since $\Phi_0^{T_m}(\mathbf{x}_{ij})$ is second order accurate, $|\tilde{\Phi}_0^{T_m}(\mathbf{x}_{ij}) - \Phi_0^{T_m}(\mathbf{x}_{ij})| \leq C_3' \Delta x^2$ where C_3' is a constant. Besides, by Lemma 4.4, we have $\sup_{\mathbf{x}\in\Omega} |\nabla \sigma_0^{T_m}(\mathbf{x})| \leq C_3''\, e^{LT_m}$ where C_3'' is a constant and therefore $I_3 \leq C_3 \Delta x^2$ where $C_3 = C_3' C_3''\, e^{LT_m}$. As a result, $\sigma_0^{2T_m}(\mathbf{x}_{ij})$ is second order accurate. □

Iteratively applying Lemma 4.3 and Theorem 4.1 will give the following result:

Theorem 4.2. *Assume that $\Phi_0^{T_m}(\mathbf{x}_{ij})$ and $\sigma_0^{T_m}(\mathbf{x}_{ij})$ computed in step 3 of Algorithm 2 are both second order accurate and the interpolation scheme in step 4 is at least second order accurate. $\Phi_0^{2^k T_m}(\mathbf{x}_{ij})$ and $\sigma_0^{2^k T_m}(\mathbf{x}_{ij})$ are second order accurate for any positive integer k.*

5. Numerical examples

In this section, we will demonstrate the proposed algorithms on four examples. The first three examples are the double gyre flow, the Rayleigh-Bénard convection cells and the forced-damped Duffing van der Pol equation. The velocity fields are all synthetic and are analytically determined by a stream function. The fourth velocity field is given by a real dataset.

5.1. The double gyre flow

This example is taken from [29] to describe a periodically varying double-gyre. The flow is modeled by the following stream-function $\psi(x, y, t) = A \sin[\pi k(x, t)] \sin(\pi y)$, where

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

In this example, we follow [29] and set $A = 0.1$, $\omega = 2\pi/10$ and use only the discrete velocity data at mesh points.

We first set $\epsilon = 0.1$ and look into the mixing-based partitions of the computational domain $\Omega = [0, 2] \times [0, 1]$ at several time slices. In particular, we have shown in Figure 2 the partitions at the time $t = 1$, $t = 5$, $t = 7$ and $t = 10$, where the dark red region, the blue region and the thin green layer correspond to the hyperbolic region $\mathcal{H}(t)$, the elliptic region $\mathcal{E}(t)$ and the parabolic region $\mathcal{P}(t)$, respectively.

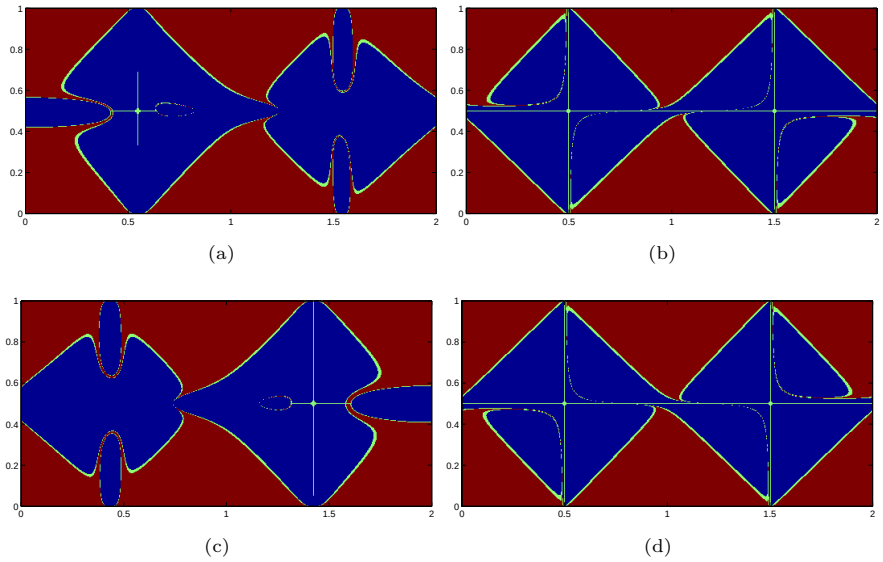


Figure 2. (Section 5.1) The mixing-based partition with $\epsilon = 0.1$ at the time level (a) $t = 1$, (b) $t = 5$, (c) $t = 7$ and (d) $t = 10$.

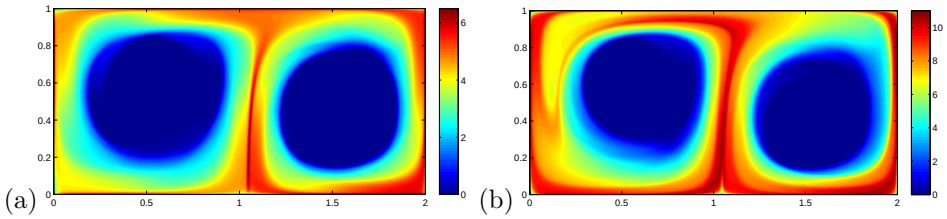


Figure 3. (Section 5.1) (a) The $\sigma_0^5(\mathbf{x})$ field and (b) the $\sigma_0^{10}(\mathbf{x})$ field with $\epsilon = 0.1$, computed using the proposed Algorithm 1.

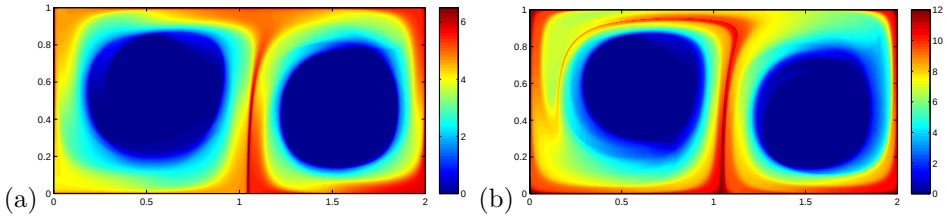


Figure 4. (Section 5.1) (a) The $\sigma_0^5(\mathbf{x})$ field and (b) the $\sigma_0^{10}(\mathbf{x})$ field with $\epsilon = 0.1$, computed using the Lagrangian approach.

Table 1. (Section 5.1) The computational time of the proposed Eulerian approach and the Lagrangian approach with different Δx's and fixed $\Delta t/\Delta x$.

Δx	1/32	1/64	1/128	1/256	1/512
Eulerian approach	1.1s	4.5s	28.6s	258.9s	2024.7s
Lagrangian approach	1.8s	10.1s	61.3s	607.1s	4842.3s

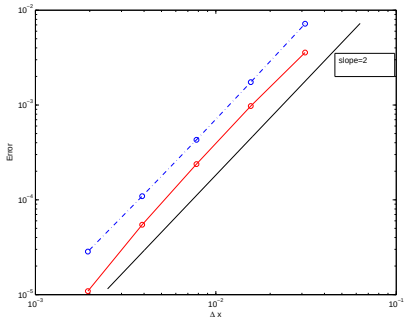


Figure 5. (Section 5.1) The L_1 errors of the solutions $\sigma_0^{10}(\mathbf{x})$ computed using the proposed Eulerian approach (red solid line) and the Lagrangian approach (blue dot dashed line). We also plot a solid black line with slope 2 as a reference.

Note that in the original definition, the parabolic region is given by $\mathcal{P}(t) = \{\mathbf{x} \in \Omega : \min\{\varphi^+(\mathbf{x}, t), \varphi^-(\mathbf{x}, t)\} = 0\}$. To have a better visualization, we have relaxed the parabolic region as $\mathcal{P}(t) = \{\mathbf{x} \in \Omega : \min\{\varphi^+(\mathbf{x}, t), \varphi^-(\mathbf{x}, t)\} \in (-0.1\Delta x, 0.1\Delta x)\}$. Then we use the proposed Algorithm 1 to compute the $\sigma_0^5(\mathbf{x})$ field and the $\sigma_0^{10}(\mathbf{x})$ field as shown in Figure 3(a) and (b), respectively, with the mesh size $\Delta x = \Delta y = 1/256$. The hyperbolic LCS at $t = 0$ is then located by looking for local maximum curves of the corresponding $\sigma_0^T(\mathbf{x})$ field. It can be seen that we can identify sharper hyperbolic LCS from the $\sigma_0^{10}(\mathbf{x})$ field. As a comparison, the solutions computed with the Lagrangian approach are also shown in Figure 4 where the TVD-RK2 scheme is used for the integration of corresponding ODEs and the third order cubic spline method is used as the interpolation operator for solving the velocity and velocity gradient at off-grid points. The solutions from the two approaches are almost the same. Then we compare the L_1 errors of the solutions $\sigma_0^{10}(\mathbf{x})$ using the two approaches as shown in Figure 5, with Δx varying from $1/32$ to $1/512$ while keeping $\Delta t/\Delta x$ fixed. Since we do not have the exact solution, the comparison benchmark is the Lagrangian solution with a pretty small Δx and Δt where the velocity field is analytically given and the RK4 method is used as the numerical integration scheme. As can be seen from Figure 5, our Eulerian approach is a little more accurate than the Lagrangian approach and the proposed Eulerian approach shows second order accuracy with respect to Δx. For the Lagrangian approach, high order numerical algorithms have been used to implement the integration and interpolation and therefore, we can also observe second order accuracy. However, once low order interpolation schemes are used for the Lagrangian approach, it will no longer show second order accuracy. As for the computational efficiency, our Eulerian approach behaves much better than the Lagrangian approach. First, our approach only needs to solve one single PDE (3.3) while the Lagrangian approach needs to solve the ODE system (1.1) and also an integral (2.3). Furthermore, the Lagrangian approach requires interpolation on the discrete velocity data and also the discrete velocity gradient data defined on the mesh, which could cost much computational time. The comparison is given in Table 1 which shows that the Lagrangian approach requires twice to three times the computational time of our Eulerian approach.

Then we implement our algorithms for $\epsilon = 0.5$. We have also shown the mixing-partition of the physical space at the time level $t = 1$, $t = 5$, $t = 7$ and $t = 10$, respectively, in Figure 6(a), (b), (c) and (d). Recalling the definition of the σ_0^T field

as given by (2.3), the integral is not computed along the whole particle trajectory $\mathbf{x}(t)$, because the trajectory might not always stay in the hyperbolic region $\mathcal{H}(t)$. For example, we have considered the particular trajectory of the particle taking off the point $(0.5, 0.25)$. The locations of this particle at $t = 1$, $t = 5$, $t = 7$ and $t = 10$ are marked as white triangles in Figure 6, from which we can see that the particle stays in the hyperbolic region at $t = 1$ and $t = 7$ while in the elliptic region at $t = 5$ and $t = 10$. Corresponding to our Eulerian formulation (3.3), it means that $Q(\Phi_0^5(0.5, 0.25), 5) = Q(\Phi_0^{10}(0.5, 0.25), 10) = 0$. The $\sigma_0^5(\mathbf{x})$ field and the $\sigma_0^{10}(\mathbf{x})$ field are given in Figure 7, also from the latter we can observe sharper and finer hyperbolic LCS.

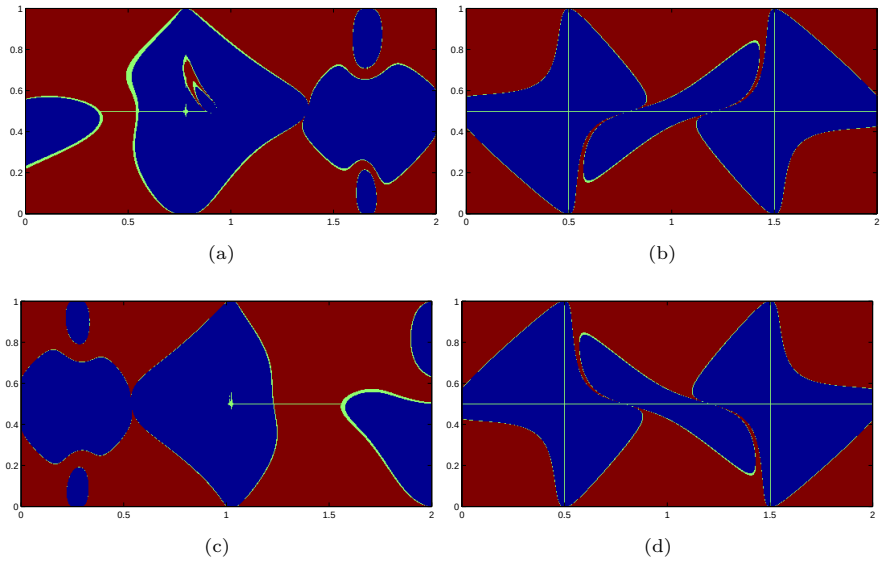


Figure 6. (Section 5.1) The mixing-based partition with $\epsilon = 0.5$ at the time level (a) $t = 1$, (b) $t = 5$, (c) $t = 7$ and (d) $t = 10$. The white triangles are the locations of the particle taking off from $(0.5, 0.25)$ at $t = 0$.

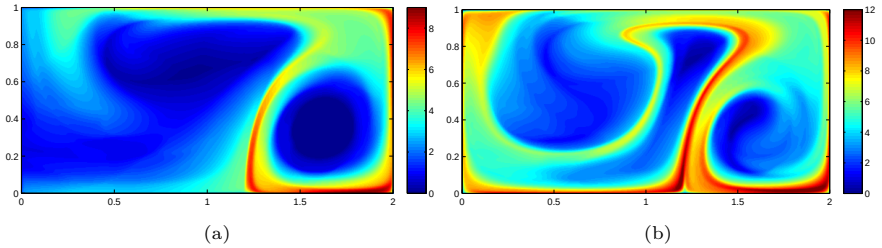


Figure 7. (Section 5.1) (a) The $\sigma_0^5(\mathbf{x})$ field and (b) the $\sigma_0^{10}(\mathbf{x})$ field with $\epsilon = 0.5$, computed using the proposed Algorithm 1.

When $\epsilon = 0.5$, the double gyre flow is periodically varying with the period $T_m = 10$. As a result, we can use the Algorithm 2 proposed in Section 3.2 to compute the longtime $\sigma_0^T(\mathbf{x})$ field, i.e. $T \gg 1$. In particular, we solve PDEs (2.2) and (3.3) backward from $t = T_m = 10$ and $t = t_0 = 0$ to obtain $\sigma_0^{10}(\mathbf{x}_{ij})$ and

$\Phi_0^{10}(\mathbf{x}_{ij})$ at each mesh point $\mathbf{x}_{ij}$. Then based on $\sigma_0^{10}(\mathbf{x}_{ij})$ and $\Phi_0^{10}(\mathbf{x}_{ij})$, we only need to iteratively interpolate 1, 2, 3 and 4 times to obtain the $\sigma_0^T(\mathbf{x}_{ij})$ field for $T = 20$, 40, 80 and 160, respectively. The corresponding solutions are given in Figure 8, where we have used $\Delta x = \Delta y = 1/256$. It can be seen that the bigger the value of T, the more complex the underlying hyperbolic LCS. Finally we numerically show the convergence behavior of the solution $\sigma_0^{160}(\mathbf{x}_{ij})$ by gradually decreasing the mesh size. The L_1 errors of the solution $\sigma_0^{160}(\mathbf{x})$ are plotted using the red solid line in Figure 9, which shows approximately second order accuracy matching with our claim in Theorem 3.

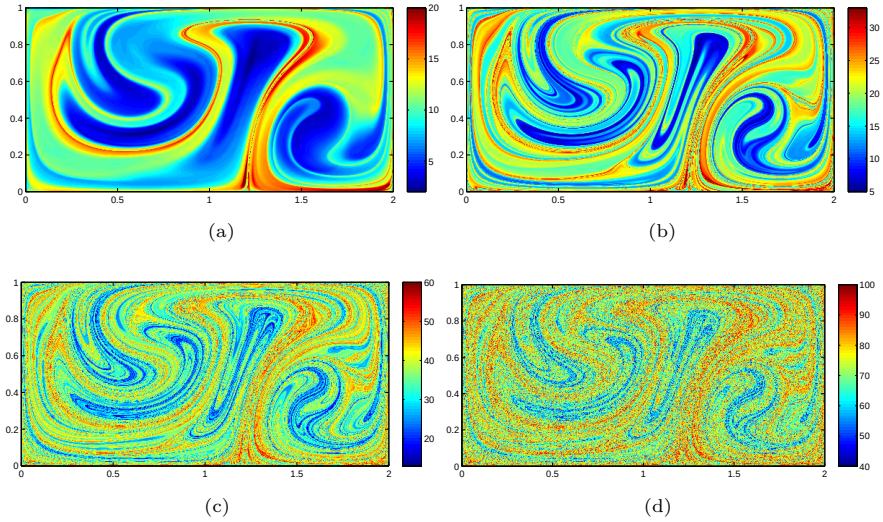


Figure 8. (Section 5.1) The $\sigma_0^T(\mathbf{x})$ field computed using the Algorithm 2, for (a) $T = 20$, (b) $T = 40$, (c) $T = 80$ and (d) $T = 160$, respectively.

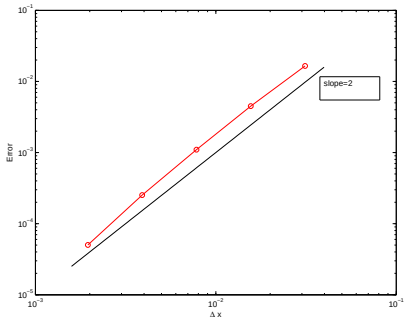


Figure 9. (Section 5.1) The L_1 errors of the solution $\sigma_0^{160}(\mathbf{x})$ computed using the proposed Eulerian approach (red solid line). We also plot a solid black line with slope 2 as a reference.

5.2. Rayleigh-Bénard convection cells

In this section, we take the example of Rayleigh-Bénard convection cells introduced in [31]. The stream-function of this model is given by

$$\psi(x, y, t) = \sin[\pi(x - g(t))] \sin(\pi y) .$$

Following [15], we here use a quasi-periodic roll motion $g(t) = 0.3 \sin(4\pi t) + 0.1 \sin(2t)$. The mixing-based partition is shown in Figure 10 at four different time levels $t = 0$, $t = 0.3$, $t = 0.7$ and $t = 1.9$. Then the proposed algorithm is used to compute the σ_0^1 and σ_0^2 fields, based on which we can identify the underlying LCS. The corresponding solutions are shown in Figure 11(a), (b), respectively, computed using the mesh size $\Delta x = \Delta y = 1/256$. As we can see, the LCS identified from the σ_0^2 field is much finer and more complex than the σ_0^1 field.

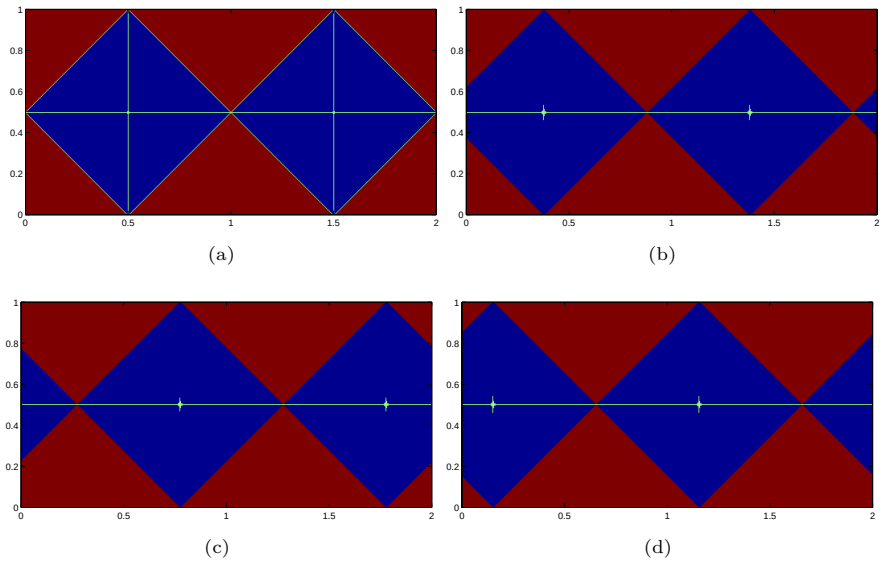


Figure 10. (Section 5.2) The mixing-based partition at the time level (a) $t = 0$, (b) $t = 0.3$, (c) $t = 0.7$ and (d) $t = 1.9$.

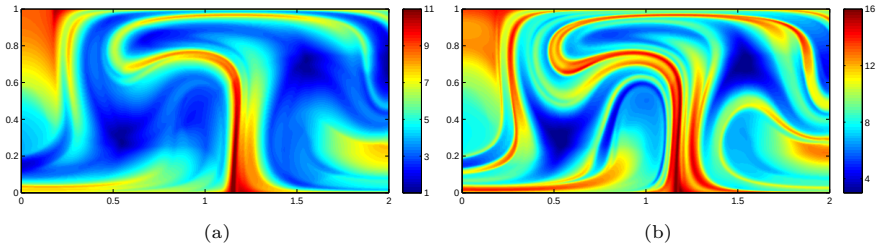


Figure 11. (Section 5.2) (a) The $\sigma_0^1(\mathbf{x})$ field and (b) the $\sigma_0^2(\mathbf{x})$ field computed using the proposed Eulerian approach.

5.3. The forced-damped Duffing van der Pol equation

We consider the dynamical system governed by a Duffing and a van der Pol oscillator, as in [11]. The system is given by

$$u = y\,,\; v = x - x^3 + 0.5y(1 - x^2) + 0.1\sin t\,.$$

The computational domain is $[-2, 2] \times [-1.5, 1.5]$, with mesh size $\Delta x = \Delta y = 0.01$. We use the proposed algorithm to compute the σ_0^{10} field. The discretization size is rather coarse for such a complicated dynamics, but the proposed algorithm can still capture the fine details of the LCS as shown in Figure 12.

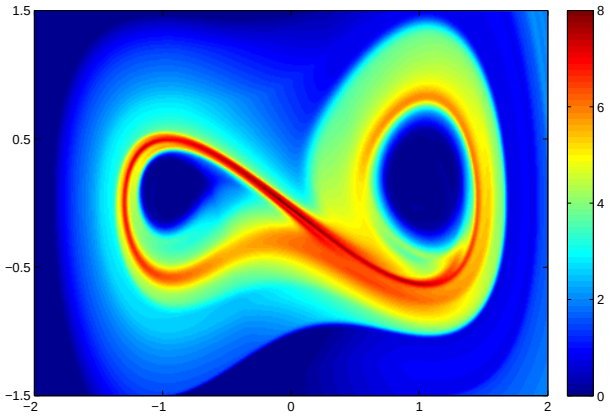
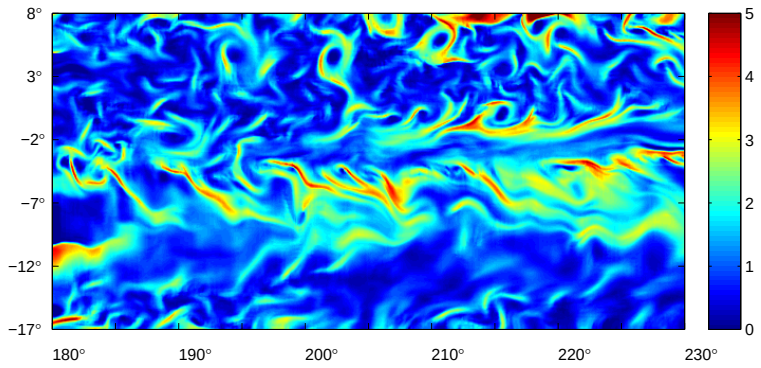


Figure 12. (Section 5.3) (a) The σ_0^{10} field computed using the proposed Eulerian approach.

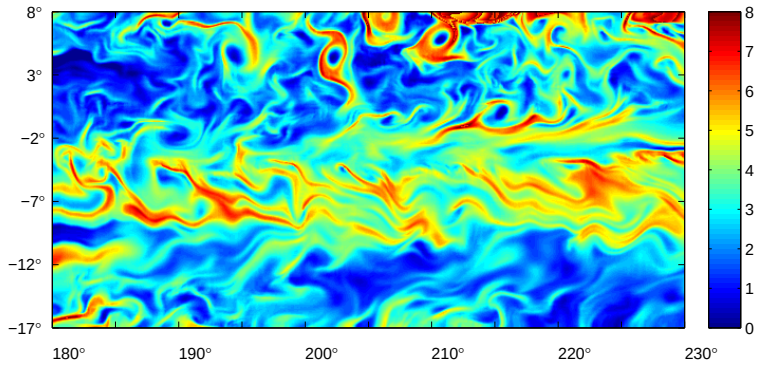
5.4. Ocean Surface Current Analyses Real-time (OSCAR)

To demonstrate the effectiveness of the proposed algorithm, we consider the Ocean Surface Current Analyses Real-time (OSCAR) dataset in which the velocity data is only available at discrete locations. In [40], we have used this dataset to test the behaviors of the Eulerian methods for computing the FTLE field and the ISLE field. 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 1/12° in each spatial direction and 0.125 days in the temporal direction which gives $\Delta x = \Delta y = 1/12$ and $\Delta t = 0.125$. Here we compute the σ_0^T field of this dataset in order to identify the hyperbolic LCS. The σ_0^{20} and σ_0^{50} fields, computed using our Algorithm 1, are shown in Figure 13(a) and (b), respectively. We can see that the proposed Eulerian approach works well in capturing fine features of the LCS even for real data. Using the proposed algorithm, the CPU times required to compute the σ_0^{20} and σ_0^{50} fields are 44 and 101 seconds, respectively. As a comparison, we have also computed the σ_0^{50} field using the Lagrangian approach as shown in Figure

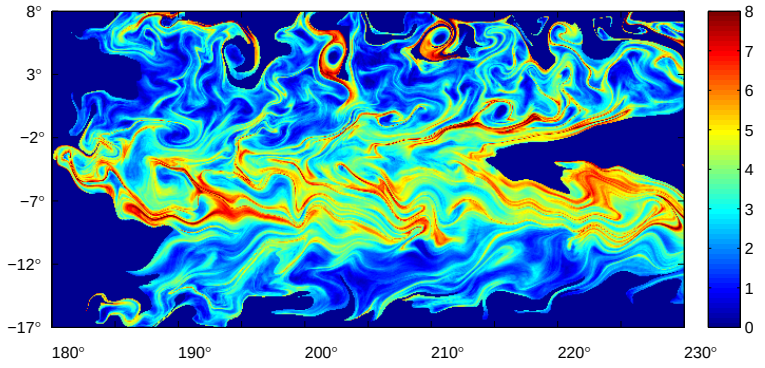
13(c). We can observe similar FTLE ridge structures in Figure 13(b) and (c). However, the Lagrangian approach requires much more time (243 seconds) to compute the σ_0^{50} field than the proposed algorithm.



(a)



(b)



(c)

Figure 13. (Section 5.4) (a) The σ_0^{20} field and (b) the σ_0^{50} field computed using the proposed Eulerian approach. (c) The σ_0^{50} field computed using the Lagrangian approach.

6. Conclusions

Based on the so-called mixing-based partition of the physical space, we have proposed an efficient Eulerian algorithm to identify the hyperbolic LCS of any given two-dimensional flow field. The only thing we need to do is solving one single PDE without interpolation procedure. In contrast, the traditional Lagrangian ray tracing approach needs to solve an ODE system together with a line integral for each particle taking from a mesh point. Furthermore, if velocity data is only available at mesh points, the Lagrangian approach needs to use the interpolation schme to obtain the velocity and the velocity gradient data along the each particle trajectory. As a result, the proposed Eulerian algorithm is more efficient. Based on the doubling technique, we have also proposed an efficient iterative Eulerian algorithm to identify the longtime LCS for periodic flows. Moreover, we have theoretically shown that the proposed algorithms have second order accuracy. Numerical examples have shown the accuracy, efficiency and effectiveness of the proposed Eulerian algorithms.

Acknowledgements

The authors are grateful to the anonymous referees for their useful comments and suggestions.

References

[1] M. Badas, F. Domenichini and G. Querzoli, *Quantification of the blood mixing in the left ventricle using finite time Lyapunov exponents*, Meccania, 2017, 52, 529–544.

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

[3] A. Carusone, C. Sicot, J. Bonnet and J. Borée, *Transient dynamical effects induced by single-pulse fluidic actuation over an airfoil*, Exp. Fluids, 2021, 62, 25.

[4] M. Dawoodian and A. Sau, *Kinetics and prey capture by a paddling jellyfish: three-dimensional simulation and Lagrangian coherent structure analysis*, J. Fluid Mech., 2021, DOI: 10.1017/jfm.2020.1069.

[5] S. Gottlieb and C. Shu, *Total variation diminishing Runge-Kutta schemes*, Mathematics of Computation, 1998, 67, 73–85.

[6] M. Green, C. Rowley and A. Smiths, *Using hyperbolic Lagrangian coherent structures to investigate vortices in biospired fluid flows*, Chaos, 2010, 20, 017510.

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

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

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

[10] G. Haller, *A variational theory of hyperbolic Lagrangian coherent structure*, Physica D, 2011, 240, 574–598.

[11] G. Haller and T. Sapsis, *Lagrangian coherent structures and the smallest finite-time Lyapunov exponent*, Chaos, 2011, 21(2), 017510.

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

[13] D. Lasagna, A. Sharma and J. Meyers, *Periodic shadowing sensitivity analysis of chaotic systems*, J. Comput. Phys., 2019, 391, 119–141.

[14] F. Lekien and N. Leonard, *Dynamically consistent Lagrangian coherent structures*, Experimental Chaos: 8-th Experimental Chaos Conference, 2004, 132–139.

[15] F. Lekien and S. Ross, *The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds*, Chaos, 2010, 20, 017505.

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

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

[18] S. Leung, *The backward phase flow method for the Eulerian finite time Lyapunov exponent computations*, Chaos, 2013, 23, 043132.

[19] 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, 2019, 2, 579–622.

[20] D. Lipinski and K. Mohseni, *Flow structures and fluid transport for the hydromedusae Sarsia tubulosa and Aequorea victoria*, J. Exp. Biology, 2009, 212, 2436–2447.

[21] X. Liu, S. Osher and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 1994, 115, 200–212.

[22] X. Liu, S. J. Osher and T. Chan, *Weighted Essentially NonOscillatory schemes*, J. Comput. Phys., 1994, 115, 200–212.

[23] S. Lukens, X. Yang and L. Fauci, *Using Lagrangian coherent structures to analyze fluid mixing by cillia*, Chaos, 2010, 20, 017511.

[24] B. Manda, B. Senyange and C. Skokos, *Chaotic wave-packet spreading in two-dimensional disordered nonlinear lattices*, Phys. Rev. E, 2020, 101, 032206.

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

[26] S. J. Osher and J. A. Sethian, *Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 1988, 79, 12–49.

[27] T. Sagristè, S. Jordan and S. F., *Visual analysis of the finite-time Lyapunov exponent*, Comput. Graph. Forum, 2020, 39(3), 331–342.

[28] T. Sapsis and G. Haller, *Inertial particle dynamics in a hurricane*, Journal of the Atmospheric Sciences, 2009, 66, 2481–2492.

[29] S. Shadden, F. Lekien and J. Marsden, *Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows*, Physica D, 2005, 212, 271–304.

[30] C. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, 1697 (Edited by B. Cockburn, C. Johnson, C. Shu and E. Tadmor), Springer, 1998, 325–432. Lecture Notes in Mathematics.

[31] T. Solomon and J. Gollub, *Chaotic particle-transport in timedependent rayleigh-b¡ enard convection*, Phys. Rev. A, 1988, 38, 6280–6286.

[32] W. Tang, P. 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, 2010, 20, 017502.

[33] W. Tang and T. Peacock, *Lagrangian coherent structures and internal wave attractors*, Chaos, 2010, 20, 017508.

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

[35] G. You and S. Leung, *Eulerian based interpolation schemes for flow map construction and line integral computation with applications to Lagrangian coherent structures extraction*, Journal of Scientific Computing, 2018, 74, 70–96.

[36] G. You and S. Leung, *An improved Eulerian approach for the finite time Lyapunov exponent*, Journal of Scientific Computing, 2018, 76, 1407–1435.

[37] G. You and S. Leung, *Fast construction of forward flow maps using Eulerian based interpolation schemes*, Journal of Scientific Computing, 2020, 82, 32.

[38] G. You and S. Leung, *Computing the finite time Lyapunov exponent for flows with uncertainties*, J. Comp. Phys., 2021, 405, 109905.

[39] G. You, Y. Shan and Y. Xu, *Fast computations for the Lagrangian-averaged vorticity deviation based on the Eulerian formulations*, International Journal of Computational Methods, 2020, 17, 1950078.

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