J Sci Comput (2018) 76:1407–1435
https://doi.org/10.1007/s10915-018-0669-y

An Improved Eulerian Approach for the Finite Time Lyapunov Exponent

Guoqiao You[1] · **Shingyu Leung**[2]

Received: 13 September 2017 / Revised: 18 January 2018 / Accepted: 7 February 2018 /
Published online: 17 February 2018

Abstract We propose a new Eulerian numerical approach to compute the Jacobian of flow maps in continuous dynamical systems and subsequently the so-called finite time Lyapunov exponent (FTLE) for Lagrangian coherent structure extraction. The original approach computes the flow map and then numerically determines the Jacobian of the map using finite differences. The new algorithm improves the original Eulerian formulation so that we first obtain partial differential equations for each component of the Jacobian and then solve these equations to obtain the required Jacobian. For periodic dynamical systems, based on the time doubling technique developed for computing the longtime flow map, we also propose a new efficient iterative method to compute the Jacobian of the longtime flow map. Numerical examples will demonstrate that our new proposed approach is more accurate than the original one in computing the Jacobian and thus the FTLE field, especially near the FTLE ridges.

Keywords Partial differential equations · Flow maps · Coherent structures · Finite time Lyapunov exponent · Flow visualization

1 Introduction

Tools are needed to visualize, understand and then extract useful information in complex continuous dynamical systems including ocean flows [15,27], hurricane structures [26], flight path [3,29], gravity waves [30], blood mixing in cardiovascular flows [1] and some other bio-inspired fluid flows [10,21,23]. A lot of work in dynamical systems focuses on

✉ Guoqiao You
270217@nau.edu.cn

Shingyu Leung
masyleung@ust.hk

[1] School of Statistics and Mathematics, Nanjing Audit University, Nanjing 211815, China

[2] Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

understanding different types of behaviors in a model, such as elliptical zones, hyperbolic trajectories, chaotic attractors, and mixing regions. 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). There is clearly more than one way to precisely define the LCS. One possible way to to extract the structure based on the so-called finite time Lyapunov exponent (FTLE) [11–13,17,27]. 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.

Since FTLE is long treated as a Lagrangian property of a continuous dynamical system, most numerical methods are developed based on the traditional Lagrangian ray tracing method by solving the ODE system using any well-developed numerical integrator. These approaches, however, require the velocity field defined at arbitrary locations in the whole space depending on the location of each individual particle. This implies that one has to in general implement some interpolation routines in the numerical code. Unfortunately, it could be a numerically challenging task to develop an interpolation approach which is computationally cheap, high order accurate yet monotone (due to the numerical stability constraint for nonsmooth velocity fields).

Incorporating the level set method [24], an Eulerian approach has been first proposed in [18] to compute the flow map and the FTLE of continuous dynamical systems. Based on the phase flow method [2,20], we have developed a backward phase flow method for the Eulerian FTLE computations in [19]. In particular, a doubling technique is incorporated to efficiently compute the longtime flow map and corresponding FTLE for periodic dynamical systems.

No matter in the traditional Lagrangian approach or the Eulerian approach developed in [18,19,33], one needs to first compute the flow map and then use certain finite difference scheme, e.g. the central difference scheme, to compute the corresponding FTLE. However, the numerical dissipation in the standard finite difference scheme might lead to large error when computing the Jacobian of the flow map. Having said that, the computed FTLE values near the exact FTLE ridge still seem larger than those of nearby locations. As a result, the computed location of the FTLE ridge will not deviate a lot from the exact ridge location and this leads people to ignore the fact that the computed FTLE values near the ridge might have large errors compared to their exact values. Indeed, there are various high order numerical methods for the advection equation. In all of our previous work, we have been applying some rather standard numerical approaches like WENO5-TVDRK2 [9,22,28] to obtain highly accurate numerical solutions. It is of course possible to further improve the accuracy by replacing the finite difference scheme by other numerical approaches such as the Runge–Kutta discontinuous galerkin (RKDG) methods [4–6]. Based on the Lagrangian interpretation, various adaptive methods have also been proposed to improve both the computational efficiency and the numerical accuracy. For example, [7] has proposed adaptively refining the underlying Cartesian mesh locally so that the mesh size is $h/2^l$ for some adaptive level l where h is the grid size on the coarsest level. Adaptive methods based on a triangular mesh have also been implemented in [16]. Graphics Processing Unit (GPU) has also been applied for such computations for efficient parallel computations [8].

In this paper, however, we are going to develop an improved Eulerian-based approach to more accurately compute the FTLE. Instead of using a finite difference step after obtaining the flow map as in the original Eulerian approach [18], we first derive a PDE system regarding

the components of the Jacobian of the flow map, and then solve the PDE system to directly obtain the Jacobian and also the required FTLE field. Similar to some observations in [25], we find that the numerical Jacobian is in general more accurate. Numerical experiments show that the FTLE values near the FTLE ridge are more accurate using this new approach. For periodic dynamical systems, an efficient Eulerian algorithm was proposed to compute the longtime FTLE [19]. In particular, a doubling technique was introduced to compute the longtime flow map first and then a finite difference step was imposed to obtain the corresponding Jacobian for the FTLE computations. In this paper, we are also going to propose an efficient algorithm to compute the longtime FTLE for periodic dynamical systems based on our new Eulerian approach. Based on the doubling technique, we will directly derive a formula linking the Jacobian of longtime flow maps to the Jacobian of short-time flow maps and then compute the FTLE based on the Jacobian without any finite difference step.

This paper is organized as follows. In Sect. 2 we will give a summary of some important concepts and also our original Eulerian formulations for computing the flow maps and the FTLE. In Sect. 3, we give our proposed Eulerian algorithms, including the new algorithm to compute the longtime FTLE for periodic flows. After that, we use the idea of our new approach to numerically solve the linear advection equation to show that our new approach behaves better than the original Eulerian approach in computing the derivative of the solution to the linear advection equation. For completeness, a complexity analysis of our new algorithm is given at the end of Sect. 3. Finally, some numerical examples will be given in Sect. 4.

2 Background

In this section, we will summarize several useful concepts and methods, which will be useful for the developments that we are proposing. We first introduce the definition of the finite time Lyapunov exponent (FTLE) [11–13,17,27], then we summarize the original Eulerian approach to compute the FTLE as in [18] and also the original Eulerian algorithm to compute the longtime FTLE for periodic flows as in [19].

2.1 Finite Time Lyapunov Exponent (FTLE)

We consider a continuous dynamical system governed by the following ordinary differential equation (ODE)

$$\mathbf{x}'(t) = \mathbf{u}(\mathbf{x}(t), t) \tag{1}$$

with the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$. The velocity field $\mathbf{u} : \Omega \times \mathbb{R} \to \mathbb{R}^d$ is a time dependent Lipschitz function where $\Omega \subset \mathbb{R}^d$ is a bounded domain in the d-dimensional space. To simplify the notation in the later sections, we collect the solutions to this ODE for all initial conditions in Ω at all time $t \in \mathbb{R}$ and introduce the flow map

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

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 ODE (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 $t = b$.

FTLE [11–13,17,27] measures the rate of separation between adjacent particles over a finite time interval with an infinitesimal perturbation in the initial location. Mathematically, consider the initial time to be 0 and the final time to be t, we have the change in the initial infinitesimal perturbation given by

$$\begin{aligned}\delta\mathbf{x}(t) &= \Phi_0^t(\mathbf{x} + \delta\mathbf{x}(0)) - \Phi_0^t(\mathbf{x}) \\ &= \nabla\Phi_0^t(\mathbf{x})\delta\mathbf{x}(0) + \text{ higher order terms.}\end{aligned}$$

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

$$\|\delta\mathbf{x}(t)\| = \sqrt{\langle\delta\mathbf{x}(0), [\nabla\Phi_0^t(\mathbf{x})]^*\nabla\Phi_0^t(\mathbf{x})\delta\mathbf{x}(0)\rangle}.$$

Here $\nabla\Phi_0^t(\mathbf{x})$ is the spatial derivatives or the **Jacobian of the flow map**. With the strain tensor matrix $\Delta_0^t(\mathbf{x}) = [\nabla\Phi_0^t(\mathbf{x})]^*\nabla\Phi_0^t(\mathbf{x})$, the finite time Lyapunov exponent (FTLE) $\sigma_0^t(\mathbf{x})$ is defined as

$$\sigma_0^t(\mathbf{x}) = \frac{1}{|t|}\ln\sqrt{\lambda_{\max}[\Delta_0^t(\mathbf{x})]} = \frac{1}{|t|}\ln\sqrt{\lambda_0^t(\mathbf{x})}. \tag{2}$$

where $\lambda_0^t(\mathbf{x}) = \lambda_{\max}(\Delta_0^t(\mathbf{x}))$ denotes the largest eigenvalue of the Cauchy-Green tensor. The absolute value of t in the expression reflects the fact that we can trace the particles either *forward* or *backward* in time. In the case when $t < 0$, we are measuring the maximum stretch *backward* in time and this corresponds to the maximum compression *forward* in time. To distinguish different measures, we call $\sigma_0^t(\mathbf{x})$ the *forward* FTLE if $t > 0$ and the *backward* FTLE if $t < 0$.

2.2 The Original Eulerian Approach for Computing the FTLE

According to the definition of the FTLE, we can see that the key point to compute FTLE is to accurately approximate the Jacobian of the flow map. The typical Lagrangian approach first solves ODE system (1) by some high-order numerical integration methods to obtain the flow map and then uses certain finite difference method, e.g. the central difference method, to compute the required Jacobian.

In contrast to the Lagrangian approach, we proposed an Eulerian approach to compute the FTLE of given velocity fields [18]. We briefly summarize the ideas as follows and we refer interested readers to [18] and thereafter.

We define a vector-valued function $\Psi = (\Psi^1, \Psi^2, \ldots, \Psi^d) : \Omega \times \mathbb{R} \to \mathbb{R}^d$. At $t = 0$, we initialize these functions by

$$\Psi(\mathbf{x}, 0) = \mathbf{x} = (x^1, x^2, \ldots, x^d). \tag{3}$$

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

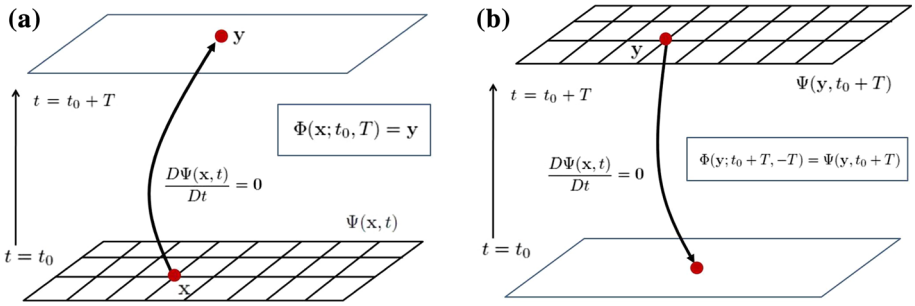

Fig. 1 Lagrangian and Eulerian interpretations of the function Ψ [18]. **a** Lagrangian ray tracing from a given grid location $\mathbf{x}$ at $t = 0$. Note that $\mathbf{y}$ might be a non-grid point. **b** Eulerian values of Ψ at a given grid location $\mathbf{y}$ at $t = T$ gives the corresponding take-off location at $t = 0$. Note the take-off location might not be a mesh point

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,

$$\frac{\partial\Psi(\mathbf{x}, t)}{\partial t} + (\mathbf{u} \cdot \nabla)\Psi(\mathbf{x}, t) = \mathbf{0} \tag{4}$$

with the initial condition (3).

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 Fig. 1a.

The solution to (4) 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 Fig. 1b, 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 (4) 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, [18] 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 (4) backward in time. Based on the forward flow map, the Jacobian of the flow map and hence the forward FTLE can be easily computed. A typical algorithm of this type in 2D case is given in Algorithm 1.

Algorithm 1: Computing the forward FTLE $\sigma_0^T(\mathbf{x})$:

1. Discretize the computational domain to get x_i, y_j, t_k.
2. Initialize the level set functions on the last time level $t = T = t_K$
$$\Psi^1(x_i, y_j, t_K) = x_i$$
$$\Psi^2(x_i, y_j, t_K) = y_j.$$
3. Solve the Liouville equations for each individual level set function $l = 1, 2$
$$\frac{\partial \Psi^l}{\partial t} + (\mathbf{u} \cdot \nabla)\Psi^l = 0$$
from $t = t_K$ down to $t = 0$ using any well-developed high order numerical methods like WENO5-TVDRK2 [9,22,28] with the boundary conditions
$$\Psi(\mathbf{x}, t)|_{\mathbf{x}\in\partial\Omega} = \mathbf{x} \quad \text{if } \mathbf{n}\cdot\mathbf{u} < 0 \tag{5}$$
$$\mathbf{n} \cdot \nabla\Psi^l(\mathbf{x}, t)|_{\mathbf{x}\in\partial\Omega} = 0 \quad \text{if } \mathbf{n}\cdot\mathbf{u} > 0 \tag{6}$$
where $\mathbf{n}$ is the outward normal of the boundary.
4. Assign $\Phi_0^T(x_i, y_j) = \Psi(x_i, y_j, 0)$ and compute the Cauchy-Green deformation tensor
$$\Delta_0^T(x_i, y_j) = \left[\nabla\Phi_0^T(x_i, y_j)\right]^* \nabla\Phi_0^T(x_i, y_j).$$
5. Determine the *forward* FTLE at $t = 0$ by computing the largest eigenvalue of the deformation tensor at each grid point (x_i, y_j)
$$\sigma_0^T(x_i, y_j) = \frac{1}{T} \ln\sqrt{\lambda_{max}\left[\Delta_0^T(x_i, y_j)\right]}.$$

Remark 1 In this Eulerian algorithm, one also needs to first compute the flow map in step 3 and then use certain finite difference method to obtain the Jacobian in step 4.

2.3 The Doubling Technique to Compute the Longtime Flow Map

In [19], we have proposed an efficient method 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 (4) 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 $\Phi_T^0(\mathbf{x}) = \Psi(\mathbf{x}, T)$.

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.

3 An Improved Eulerian Approach to Compute the FTLE

As mentioned in the previous section, most, if not all, approaches for computing the FTLE first determine the flow map, then use certain finite difference method to obtain the Jacobian and the deformation tensor, and finally determine its eigenvalues. Since these eigenvalues are in general very sensitive to any perturbation in the deformation tensor, accurate computations in the Jacobian matrix is crucial in determining the FTLE fields.

In this section, we will give an improved Eulerian approach to compute the FTLE where the Jacobian can be obtained by directly solving a PDE system, rather than by applying the finite difference step to the computed flow map. Corresponding to this new Eulerian approach, we are also proposing an efficient way to compute the longtime FTLE for periodic dynamical systems. After that, we will use the idea of our new approach to numerically solve the linear advection equation to show that our new approach behaves better than the original Eulerian approach by computing the derivatives of the solution to the linear advection equation. For completeness, a complexity analysis of our new algorithm is given at the end of this section.

3.1 An Improved Eulerian Algorithm to Compute the FTLE

Based on the Eulerian-formulation, we here propose a new method to compute the Jacobian of the flow map and hence the FTLE. Since the flow map has the same order of regularity as the velocity field, $\Psi(\mathbf{x}, t)$, solution to Liouville equations (4), has continuous partial derivatives as long as the velocity field $\mathbf{u}$ is smooth enough. Taking partial derivatives to both sides of equations (4) with respect to each spatial variable gives (in two-dimensional case for example):

$$\begin{cases} \dfrac{\partial \phi_x}{\partial t} + (\mathbf{u} \cdot \nabla)\phi_x = -u_x \phi_x - v_x \phi_y \\ \dfrac{\partial \phi_y}{\partial t} + (\mathbf{u} \cdot \nabla)\phi_y = -u_y \phi_x - v_y \phi_y \end{cases} \tag{7}$$

and similarly

$$\begin{cases} \dfrac{\partial \psi_x}{\partial t} + (\mathbf{u} \cdot \nabla)\psi_x = -u_x \psi_x - v_x \psi_y \\ \dfrac{\partial \psi_y}{\partial t} + (\mathbf{u} \cdot \nabla)\psi_y = -u_y \psi_x - v_y \psi_y. \end{cases} \tag{8}$$

where ϕ and ψ are used to respectively replace Ψ^1 and Ψ^2 for notational simplicity.

Given initial conditions $(\phi_x(x, y, 0), \phi_y(x, y, 0)) = (1, 0)$ and $(\psi_x(x, y, 0), \psi_y(x, y, 0)) = (0, 1)$, solving PDE systems (7) and (8) from $t = 0$ up to $t = T$ will give us the Jacobian $J_T^0(x, y)$ of the backward flow map $\Phi_T^0(x, y)$ and subsequently the FTLE can be easily computed without any finite difference step upon the flow map.

3.2 An Improved Algorithm to Compute the Longtime FTLE for Periodic Dynamical Systems

Corresponding to the new Eulerian approach for the FTLE proposed in Sect. 3.1, we here also present an efficient algorithm to compute the longtime FTLE for periodic dynamical systems.

3.2.1 The Algorithm

The first step of the algorithm is to solve PDE systems (7) and (8) together with (4) from $t = 0$ to $t = T_m$ with initial conditions $(\phi_x(x, y, 0), \phi_y(x, y, 0)) = (1, 0)$, $(\psi_x(x, y, 0), \psi_y(x, y, 0)) = (0, 1)$ and $\Psi(x, y, 0) = (x, y)$, respectively, where T_m is the period of the dynamical system. After that, we can obtain the backward flow map $\Phi^0_{T_m}(x, y) = \Psi(x, y, T_m)$ from $t = T_m$ to $t = 0$ and its spatial derivatives $(\phi_x(x, y, T_m), \phi_y(x, y, T_m))$ and $(\psi_x(x, y, T_m), \psi_y(x, y, T_m))$, i.e. components of the Jacobian $J^0_{T_m}(x, y)$.

Lemma 1 *Suppose* $\mathbf{x}(t) = (x(t), y(t))$ *is the trajectory of the particle located at* $\mathbf{x}_0 = (x_0, y_0)$ *initially at* $t = 0$, *then* $\begin{cases} f(t) = \phi_x(\Phi^t_0(\mathbf{x}_0), t) \\ g(t) = \phi_y(\Phi^t_0(\mathbf{x}_0), t) \end{cases}$ *and* $\begin{cases} f(t) = \psi_x(\Phi^t_0(\mathbf{x}_0), t) \\ g(t) = \psi_y(\Phi^t_0(\mathbf{x}_0), t) \end{cases}$ *are solutions to the following ODE system*

$$\begin{cases} \dfrac{df}{dt} = -u_x(\mathbf{x}(t), t) f - v_x(\mathbf{x}(t), t) g \\ \dfrac{dg}{dt} = -u_y(\mathbf{x}(t), t) f - v_y(\mathbf{x}(t), t) g \end{cases} \tag{9}$$

with the initial conditions $\begin{cases} f(0) = 1 \\ g(0) = 0 \end{cases}$ *and* $\begin{cases} f(0) = 0 \\ g(0) = 1 \end{cases}$, *respectively.*

Proof It can be easily seen from PDE systems (7) and (8).

As ODE system (9) is linear, we have the following lemma.

Lemma 2 *The solution to ODE system* (9) *with the initial condition* $(f(0), g(0)) = (f_0, g_0)$ *is given by*

$$\begin{aligned} f(t) &= f_0 \phi_x(\Phi^t_0(\mathbf{x}_0), t) + g_0 \psi_x(\Phi^t_0(\mathbf{x}_0), t) \\ g(t) &= f_0 \phi_y(\Phi^t_0(\mathbf{x}_0), t) + g_0 \psi_y(\Phi^t_0(\mathbf{x}_0), t) . \end{aligned}$$

Corollary 1 *For any positive integer k, we have*

$$J^0_{2^k T_m}(x, y) = J^0_{2^{k-1} T_m}(x, y)\, J^0_{2^{k-1} T_m}(\Psi(\mathbf{x}, 2^{k-1} T_m)) \tag{10}$$

where $J^0_t(x, y) \triangleq \begin{pmatrix} \phi_x(x, y, t) & \psi_x(x, y, t) \\ \phi_y(x, y, t) & \psi_y(x, y, t) \end{pmatrix}$.

Proof Due to the periodicity of the velocity field, $(\phi_x(x, y, 2T_m), \phi_y(x, y, 2T_m))$ is the solution to ODE system (9) at $t = T_m$ with the initial condition $\begin{cases} f(0) = \phi_x(\Psi(\mathbf{x}, T_m), T_m) \\ g(0) = \phi_y(\Psi(\mathbf{x}, T_m), T_m) \end{cases}$ and $\mathbf{x}_0 = \Psi(\mathbf{x}, T_m)$.

With Lemma 2, we have

$$\begin{aligned}(\phi_x(x,y,2T_m),\phi_y(x,y,2T_m)) = \;& \phi_x(\Psi(\mathbf{x},T_m),T_m)\cdot(\phi_x(x,y,T_m),\phi_y(x,y,T_m))\\ &+\phi_y(\Psi(\mathbf{x},T_m),T_m)\cdot(\psi_x(x,y,T_m),\psi_y(x,y,T_m))\end{aligned}$$

and similarly

$$\begin{aligned}(\psi_x(x,y,2T_m),\psi_y(x,y,2T_m)) = \;& \psi_x(\Psi(\mathbf{x},T_m),T_m)\cdot(\phi_x(x,y,T_m),\phi_y(x,y,T_m))\\ &+\psi_y(\Psi(\mathbf{x},T_m),T_m)\cdot(\psi_x(x,y,T_m),\psi_y(x,y,T_m))\end{aligned}$$

which implies

$$J^0_{2T_m}(x,y) = J^0_{T_m}(x,y)\,J^0_{T_m}(\Psi(\mathbf{x},T_m))$$

and, therefore, (10).

As a result, once we have computed $J^0_{T_m}(x,y)$, we can easily obtain $J^0_{2T_m}(x,y)$ based on formula (10) where $J^0_{T_m}(\Psi(\mathbf{x},T_m))$ can be obtained by interpolation. After that we can iteratively obtain the Jacobian $J^0_{2^kT_m}(x,y)$ for any $k\geq 2$. An algorithm of this approach is given in Algorithm 2.

Algorithm 2: Computing the backward FTLE $\sigma^0_T(\mathbf{x})$ for periodic flows where $T=2^nT_m$

where $n\in\mathbb{N}+$:

1. Solve PDE systems (4), (7) and (8) together from $t=0$ to $t=T_m$ with initial conditions $\Psi(x_i,y_j,0)=(x_i,y_j)$, $(\phi_x(x_j,y_j,0),\phi_y(x_i,y_j,0))=(1,0)$ and $(\psi_x(x_i,y_j,0),\psi_y(x_i,y_j,0))=(0,1)$, respectively. Then $\Psi(x_i,y_j,T_m)$ and $J^0_{T_m}(x_i,y_j)$ are obtained.
2. For $k=0,1,\ldots,n-1$, interpolate to obtain $\Psi(x_i,y_j,2^kT_m)$, $J^0_{2^kT_m}(\Psi(x_i,y_j,2^kT_m))$ and thus
$$J^0_{2^{k+1}T_m}(x_i,y_j) = J^0_{2^kT_m}(x_i,y_j)\cdot J^0_{2^kT_m}(\Psi(x_i,y_j,2^kT_m))$$
can be obtained.
3. Compute the Cauchy-Green deformation tensor $\Delta^0_T(x_i,y_j) = [J^0_T(x_i,y_j)]^*\,J^0_T(x_i,y_j)$ and thus the backward FTLE is computed as $\sigma^0_T(x_i,y_j)=\frac{1}{T}\ln\sqrt{\lambda_{max}[\Delta^0_T(x_i,y_j)]}$.

To end this section, we discuss the Lagrangian implementation of directly solving the ODE system (9). The stiffness of the system clearly depends on the property of the flow velocity. We let $M=\begin{pmatrix}u_x & v_x\\ u_y & v_y\end{pmatrix}$ such that the evolution of the Jacobian is governed by $\frac{dJ}{dt}=-MJ$. The eigenvalue of the matrix M is given by

$$\lambda_\pm = \frac{u_x+v_y\pm\sqrt{(u_x+v_y)^2-4(u_xv_y-u_yv_x)}}{2} = \frac{1}{2}\left[\nabla\cdot\mathbf{u}\pm\sqrt{(\nabla\cdot\mathbf{u})^2-4\det(M)}\right].$$

If the underlying flow is potential so that the velocity $\mathbf{u}$ is given by $(-\psi_y,\psi_x)$ for some stream function ψ, the eigenvalues $\lambda_\pm$ is given by

$$\lambda_\pm = \pm\sqrt{-\det(M)} = \pm\sqrt{-\det[H(\psi)]}$$

where $H(\psi)$ is the hessian of the stream function. Therefore, if the stream function has a settle point, the hessian matrix has one positive and one negative eigenvalues which implies that the determinant of such matrix would be negative. As a result, the matrix M has a positive eigenvalue and so the ODE system (9) will be stiff. Such a stiff ODE system will therefore require a smaller timestep to guarantee numerical stability.

3.2.2 The Influence of the Interpolation Scheme on the Accuracy of Algorithm 2

In step 2 of Algorithm 2, we have proposed to use the interpolation method to obtain the longtime flow map and the corresponding Jacobian for periodic dynamical systems. In this part we will show some results about how the interpolation scheme will influence the accuracy of Algorithm 2. We first give the following two lemmas which are natural extensions of Lemma 4 and Lemma 5 from [32] and will be used in our proof.

Lemma 3 *Suppose that the velocity field* $\mathbf{u}(\mathbf{x}, t)$ *is smooth enough and has the Lipschitz constant* L *on the computational domain* $\mathbf{x} \in M$. *We have*

$$|\Phi_0^t(\mathbf{x}_1) - \Phi_0^t(\mathbf{x}_2)| \leq e^{Lt}|\mathbf{x}_1 - \mathbf{x}_2|, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in M.$$

Lemma 4 *Suppose that the velocity field* $\mathbf{u}$ *is smooth enough. For each* $s \geq 2$, *there exists a constant* C_s *such that for any multi-index* γ *with* $|\gamma| = s$ *and any* $\mathbf{x} \in M$, *we have*

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

In the following Theorem 1 and Lemma 5, we assume that the interpolation operator has a bounded norm which is independent of the mesh size.

Theorem 1 *Assuming that* $\Psi(x_i, y_j, T_m)$ *computed in* step 1 *of* Algorithm 2 *has second order accuracy and the interpolation scheme in* step 2 *is at least second order accurate,* $\Psi(x_i, y_j, 2T_m)$ *is also second order accurate.*

Proof We use $\tilde{\cdot}$ to denote the numerical solution and $\mathcal{I}$ to denote the interpolation operator. Then $\Psi(x_i, y_j, 2T_m)$ is approximated by

$$\tilde{\Psi}(x_i, y_j, 2T_m) = \tilde{\Phi}^0_{2T_m}(\mathbf{x}_{i,j}) = \mathcal{I}\tilde{\Phi}^0_{T_m}\left(\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j})\right)$$

with the error

$$\begin{aligned}\left|\tilde{\Phi}^0_{2T_m}(\mathbf{x}_{i,j}) - \Phi^0_{2T_m}(\mathbf{x}_{i,j})\right| \leq & \left|\mathcal{I}\tilde{\Phi}^0_{T_m}\left(\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j})\right) - \mathcal{I}\Phi^0_{T_m}\left(\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j})\right)\right| \\ & + \left|\mathcal{I}\Phi^0_{T_m}\left(\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j})\right) - \Phi^0_{T_m}\left(\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j})\right)\right| \\ & + \left|\Phi^0_{T_m}\left(\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j})\right) - \Phi^0_{T_m}\left(\Phi^0_{T_m}(\mathbf{x}_{i,j})\right)\right| \\ \triangleq & \; I_1 + I_2 + I_3.\end{aligned}$$

Since $\Psi(x_i, y_j, T_m)$ is supposed to have second order accuracy, there exists a constant C_0 such that

$$|\tilde{\Psi}(x_i, y_j, T_m) - \Psi(x_i, y_j, T_m)| = \left|\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j}) - \Phi^0_{T_m}(\mathbf{x}_{i,j})\right| \leq C_0 \Delta x^2.$$

Suppose the interpolation operator has a Δx-independent norm N_I. Then I_1 is bounded by

$$I_1 \leq N_I \cdot \max_{\mathbf{x}_{i,j}} \left|\tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j}) - \Phi^0_{T_m}(\mathbf{x}_{i,j})\right| \leq C_1 \Delta x^2.$$

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

$$I_2 \le C_2' \Delta x^2 \max_{|\gamma|=2} \sup_{\mathbf{x}_{i,j}} \left| \partial^\gamma \Phi^0_{T_m}(\mathbf{x}_{i,j}) \right|.$$

According to Lemma 4, there exists a constant C_2 such that

$$\max_{|\gamma|=2} \sup_{\mathbf{x}_{i,j}} \left| \partial^\gamma \Phi^0_{T_m}(\mathbf{x}_{i,j}) \right| \le C_2 \cdot e^{3LT_m} \cdot T_m$$

which implies $I_2 \le C_3 \Delta x^2$ where C_3 is a constant. Finally, by Lemma 3, I_3 is bounded by

$$I_3 \le e^{LT_m} \left| \tilde{\Phi}^0_{T_m}(\mathbf{x}_{i,j}) - \Phi^0_{T_m}(\mathbf{x}_{i,j}) \right| \le C_4 \Delta x^2$$

where $C_4 = C_0 e^{LT_m}$ is a constant. As a result, $\Psi(x_i, y_j, 2T_m) = \Phi^0_{2T_m}(\mathbf{x}_{i,j})$ is second order accurate.

Lemma 5 *Assuming that $\Psi(x_i, y_j, T_m)$ and $J^0_{T_m}(x_i, y_j)$ computed in* step 1 *of* Algorithm 2 *have second order accuracy and the interpolation scheme in* step 2 *is at least second order accurate, then $J^0_{T_m}(\Psi(x_i, y_j, T_m))$ obtained by interpolating on $J^0_{T_m}(x_i, y_j)$ is also second order accurate.*

Proof Since $J^0_{T_m}(\Psi(x_i, y_j, T_m)) \triangleq \begin{pmatrix} \phi_x(\Psi(x_i, y_j, T_m), T_m) & \psi_x(\Psi(x_i, y_j, T_m), T_m) \\ \phi_y(\Psi(x_i, y_j, T_m), T_m) & \psi_y(\Psi(x_i, y_j, T_m), T_m) \end{pmatrix}$, we here only prove that $\phi_x(\Psi(x_i, y_j, T_m), T_m)$ is second order accurate and the accuracy of the other three terms can be similarly proved.

In fact, $\phi_x(\Psi(x_i, y_j, T_m), T_m)$ is approximated by $\mathcal{I}\tilde{\phi}_x(\tilde{\Psi}(\mathbf{x}_{i,j}, T_m), T_m)$ with the error

$$\begin{aligned}
&\left| \mathcal{I}\tilde{\phi}_x(\tilde{\Psi}(\mathbf{x}_{i,j}, T_m), T_m) - \phi_x(\Psi(\mathbf{x}_{i,j}, T_m), T_m) \right| \\
&\le \left| \mathcal{I}\tilde{\phi}_x(\tilde{\Psi}(\mathbf{x}_{i,j}, T_m), T_m) - \mathcal{I}\phi_x(\tilde{\Psi}(\mathbf{x}_{i,j}, T_m), T_m) \right| \\
&\quad + \left| \mathcal{I}\phi_x(\tilde{\Psi}(\mathbf{x}_{i,j}, T_m), T_m) - \phi_x(\tilde{\Psi}(\mathbf{x}_{i,j}, T_m), T_m) \right| \\
&\quad + \left| \phi_x(\tilde{\Psi}(\mathbf{x}_{i,j}, T_m), T_m) - \phi_x(\Psi(\mathbf{x}_{i,j}, T_m), T_m) \right| \\
&\triangleq I_1 + I_2 + I_3.
\end{aligned}$$

Suppose the interpolation operator has a Δx-independent norm N_I. Then I_1 is bounded by

$$I_1 \le N_I \cdot \max_{\mathbf{x}_{i,j}} \left| \tilde{\phi}_x(\mathbf{x}_{i,j}, T_m) - \phi_x(\mathbf{x}_{i,j}, T_m) \right|.$$

Since $J^0_{T_m}(x_i, y_j)$ is supposed to be second order accurate, we have $|\tilde{\phi}_x(\mathbf{x}_{i,j}, T_m) - \phi_x(\mathbf{x}_{i,j}, T_m)| \le C_0 \Delta x^2$ and thus $I_1 \le C_1 \Delta x^2$ where C_0, C_1 are constants. Since the interpolation scheme is at least second order accurate, then there exist a constant C_2 such that

$$\begin{aligned}
I_2 &\le C_2 \Delta x^2 \max_{|\gamma|=2} \sup_{\mathbf{x}_{i,j}} |\partial^\gamma \phi_x(\mathbf{x}_{i,j}, T_m)| \\
&\le C_2 \Delta x^2 \max_{|\gamma|=3} \sup_{\mathbf{x}_{i,j}} |\partial^\gamma \phi(\mathbf{x}_{i,j}, T_m)|.
\end{aligned}$$

According to Lemma 4, there exists a constant C_3 such that

$$\max_{|\gamma|=3} \sup_{\mathbf{x}_{i,j}} |\partial^\gamma \phi(\mathbf{x}_{i,j}, T_m)| \le C_3 \cdot e^{5LT_m} \cdot T_m$$

which implies $I_2 \leq C_4 \Delta x^2$ where $C_4 = C_2 C_3 \cdot e^{5LT_m} \cdot T_m$ is a constant. Finally, I_3 is bounded by

$$I_3 \leq \sup_{\mathbf{x} \in M} |\nabla \phi_x(\mathbf{x}, T_m)||\tilde{\Psi}(\mathbf{x}, T_m) - \Psi(\mathbf{x}, T_m)|$$

where M is the computational domain. Also from Lemma 4, we have

$$\sup_{\mathbf{x} \in M} |\nabla \phi_x(\mathbf{x}, T_m)| \leq C_5$$

where C_5 is a constant only depending on L, T_m. Furthermore, $|\tilde{\Psi}(\mathbf{x}, T_m) - \Psi(\mathbf{x}, T_m)| \leq C_6 \Delta x^2$ due to the assumption that $\Psi(\mathbf{x}, T_m)$ is second order accurate. As a result, $I_3 \leq C_5 C_6 \Delta x^2$ and thus $\phi_x(\Psi(x_i, y_j, T_m), T_m)$ is second order accurate.

Theorem 2 *Assuming that $\Psi(x_i, y_j, T_m)$ and $J^0_{T_m}(x_i, y_j)$ computed in* step 1 *of* Algorithm 2 *have second order accuracy and the interpolation scheme in* step 2 *is at least second order accurate, $J^0_{2T_m}(x_i, y_j)$ is also second order accurate.*

Proof $J^0_{T_m}(x_i, y_j)$ is supposed to be second order accurate and from Lemma 5 we know that $J^0_{T_m}(\Psi(x_i, y_j, T_m))$ is also second order accurate. As a result, the numerical solution of

$$J^0_{2T_m}(x_i, y_j) = J^0_{T_m}(x_i, y_j)\, J^0_{T_m}(\Psi(x_i, y_j, T_m))$$

is still second order accurate.

With similar derivation, we have the following corollary:

Theorem 3 *Suppose that $\Psi(x_i, y_j, T_m)$ and $J^0_{T_m}(x_i, y_j)$ computed in* step 1 *of* Algorithm 2 *have second order accuracy and the interpolation scheme in* step 2 *is at least second order accurate, then $\Psi(x_i, y_j, 2^k T_m)$ and $J^0_{2^k T_m}(x_i, y_j)$ are also second order accurate for any positive integer k.*

3.3 A Simple Analysis on the Linear Advection Equation

In this subsection, we use a slightly easier 1-dimensional case to show the idea of our new Eulerian approach proposed in Sect. 3.1 and demonstrate its effectiveness. Suppose that $\phi(x, t)$ is the smooth solution satisfying the linear advection equation in one-dimension

$$\phi_t + a\phi_x = 0 \tag{11}$$

with the initial condition $\phi(x, 0) = \phi_0(x)$ where a is a constant. We consider two different numerical strategies to compute $\phi_x(x, t)$ which is the spatial derivative of $\phi(x, t)$.

In the original approach, we first use certain numerical scheme to solve PDE (11) to approximate $\phi(x, t)$ and then use a finite difference method to obtain the numerical approximation to $\phi_x(x, t)$. Using the idea of our new approach as proposed in Sect. 3.1, we first take the partial derivatives of both sides of Eq. (11) with respect to x and let $p = \phi_x$, then we obtain

$$p_t + a p_x = 0\,. \tag{12}$$

After that, we also use certain numerical scheme to solve PDE (12) and obtain the numerical approximation to $p(x, t)$, i.e. $\phi_x(x, t)$.

In this subsection, we will give detailed discussion about the accuracies of these two different strategies for computing $\phi_x(x, t)$. For the original approach, in particular, we will

use the central difference scheme to obtain $\phi_x(x,t)$ based on $\phi(x,t)$. And we will use the Lax–Wendroff scheme and the TVDRK2-WENO5 scheme, respectively in Sects. 3.3.1 and 3.3.2, to solve both PDEs (11) and (12).

3.3.1 The Lax–Wendroff Discretization

Theorem 4 *Let ϕ_j^n be the numerical solution to Eq. (11) discretized by the Lax–Wendroff scheme, i.e.*

$$\phi_j^{n+1} = \phi_j^n - \frac{a\Delta t}{2\Delta x}\left(\phi_{j+1}^n - \phi_{j-1}^n\right) + \frac{a^2\Delta t^2}{2\Delta x^2}\left(\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n\right), \tag{13}$$

then $(\phi_x)_j^n$ is a second-order accurate approximation to $\phi_x(x,t)$ where $(\phi_x)_j^n$ is obtained by using the central difference based on ϕ_j^n.

Proof According to the Taylor's expansion we have

$$\begin{aligned}\phi(x_j,t^{n+1}) = \phi(x_j,t^n) + \Delta t\phi_t(x_j,t^n) + \frac{\Delta t^2}{2}\phi_{tt}(x_j,t^n) + \frac{\Delta t^3}{6}\phi_{ttt}(x_j,t^n)\\ + \frac{\Delta t^4}{24}\phi_{tttt}(x_j,t^n) + O(\Delta t^5).\end{aligned} \tag{14}$$

On the other hand, Eq. (11) gives

$$\phi_t = -a\phi_x, \phi_{tt} = a^2\phi_{xx}, \phi_{ttt} = -a^3\phi_{xxx}, \phi_{tttt} = a^4\phi_{xxxx}. \tag{15}$$

Plugging (15) into (14) gives

$$\begin{aligned}\phi(x_j,t^{n+1}) = \phi(x_j,t^n) - a\Delta t\phi_x(x_j,t^n) + \frac{a^2\Delta t^2}{2}\phi_{xx}(x_j,t^n) - \frac{a^3\Delta t^3}{6}\phi_{xxx}(x_j,t^n)\\ + \frac{a^4\Delta t^4}{24}\phi_{xxxx}(x_j,t^n) + O(\Delta t^5).\end{aligned} \tag{16}$$

Also from the Taylor's expansion, we can obtain

$$\phi_x(x_j,t^n) = \frac{\phi(x_{j+1},t^n) - \phi(x_{j-1},t^n)}{2\Delta x} - \frac{\Delta x^2}{6}\phi_{xxx}(x_j,t^n) + O(\Delta x^4) \tag{17}$$

and

$$\phi_{xx}(x_j,t^n) = \frac{\phi(x_{j+1},t^n) - 2\phi(x_j,t^n) + \phi(x_{j-1},t^n)}{\Delta x^2} - \frac{\Delta x^2}{12}\phi_{xxxx}(x_j,t^n) + O(\Delta x^4). \tag{18}$$

Substituting (17–18) into (16) and letting $\lambda = \frac{a\Delta t}{\Delta x}$ be the CFL number which should be less than 1, we have

$$\begin{aligned}\phi(x_j,t^{n+1}) = \phi(x_j,t^n) - \frac{\lambda}{2}[\phi(x_{j+1},t^n) - \phi(x_{j-1},t^n)]\\ + \frac{\lambda^2}{2}[\phi(x_{j+1},t^n) - 2\phi(x_j,t^n) + \phi(x_{j-1},t^n)]\\ + \frac{\lambda(1-\lambda^2)\Delta x^3}{6}\phi_{xxx}(x_j,t^n)\\ + \frac{\lambda^2(\lambda^2-1)\Delta x^4}{24}\phi_{xxxx}(x_j,t^n) + O(\Delta x^5).\end{aligned} \tag{19}$$

Denoting $e_j^n = \phi_j^n - \phi(x_j, t^n)$, then (13)-(19) implies

$$\begin{aligned} e_j^{n+1} = &\left(\frac{\lambda^2}{2} + \frac{\lambda}{2}\right) e_{j-1}^n + (1-\lambda^2) e_j^n + \left(\frac{\lambda^2}{2} - \frac{\lambda}{2}\right) e_{j+1}^n \\ &- \frac{\lambda(1-\lambda^2)\Delta x^3}{6} \phi_{xxx}(x_j, t^n) \\ &- \frac{\lambda^2(\lambda^2-1)\Delta x^4}{24} \phi_{xxxx}(x_j, t^n) + O(\Delta x^5) \end{aligned} \tag{20}$$

and thus

$$\begin{aligned} e_{j+1}^n - e_{j-1}^n = &\left(\frac{\lambda^2}{2} + \frac{\lambda}{2}\right)(e_j^{n-1} - e_{j-2}^{n-1}) + (1-\lambda^2)\left(e_{j+1}^{n-1} - e_{j-1}^{n-1}\right) \\ &+ \left(\frac{\lambda^2}{2} - \frac{\lambda}{2}\right)\left(e_{j+2}^{n-1} - e_j^{n-1}\right) \\ &+ \frac{\lambda(\lambda^2-1)}{3} \Delta x^4 \phi_{xxxx}(x_j, t^{n-1}) + O(\Delta x^5). \end{aligned} \tag{21}$$

Let $K = \frac{\lambda(\lambda^2-1)}{3}$. Using the mathematical induction, we can prove that

$$e_{j+1}^n - e_{j-1}^n = K \Delta x^4 \sum_{i=0}^{n-1} \phi_{xxxx}(x_j, t^i) + O(\Delta x^5) \tag{22}$$

is true for arbitrary $n \in \mathbb{N}_+$.

Now suppose that $|\phi_{xxx}(x,t)| \le M_{xxx}$, $|\phi_{xxxx}(x,t)| \le M_{xxxx}$ uniformly in the whole computational domain, then from (22) we can know

$$|e_{j+1}^n - e_{j-1}^n| \le |K| n M_{xxxx} \Delta x^4 + O(\Delta x^5).$$

Together with (17), the error of numerical approximation to $\phi_x(x_j, t^n)$ can be expressed as

$$\begin{aligned} |E_j^n| &\triangleq \left| \frac{\phi_{j+1}^n - \phi_{j-1}^n}{2\Delta x} - \phi_x(x_j, t^n) \right| \\ &\le \frac{|K|}{2} n M_{xxxx} \Delta x^3 + \frac{\Delta x^2}{6} M_{xxx} + O(\Delta x^4). \end{aligned}$$

Since $n\Delta x = n\frac{a\Delta t}{\lambda} \le |\frac{a}{\lambda} T|$ where $|T|$ is the final time level, we finally have

$$|E_j^n| \le \frac{|1-\lambda^2|}{6} M_{xxxx} |aT| \Delta x^2 + \frac{1}{6} M_{xxx} \Delta x^2 + O(\Delta x^4). \tag{23}$$

□

Now we use the idea of our new approach as proposed in Sect. 3.1 to approximate $\phi_x(x,t)$, i.e. we need to solve Eq. (12). Similar derivation as in the proof of Theorem 4 will give us a formula similar to (20):

$$\begin{aligned} E_j^{n+1} = &\left(\frac{\lambda^2}{2} + \frac{\lambda}{2}\right) E_{j-1}^n + (1-\lambda^2) E_j^n + \left(\frac{\lambda^2}{2} - \frac{\lambda}{2}\right) E_{j+1}^n \\ &- \frac{\lambda(1-\lambda^2)\Delta x^3}{6} p_{xxx}(x_j, t^n) + O(\Delta x^4). \end{aligned} \tag{24}$$

We can also use the mathematical induction to prove that

$$E_j^n = \frac{K}{2}\Delta x^3 \sum_{i=0}^{n-1} p_{xxx}(x_j, t^i) + O(\Delta x^4).$$

Since $p_{xxx}(x_j, t^i) = \phi_{xxxx}(x_j, t^i)$, then

$$|E_j^n| \leq \frac{|1-\lambda^2|}{6} M_{xxxx} |aT| \Delta x^2 + O(\Delta x^4). \tag{25}$$

Using the idea of our proposed approach, we first derive a PDE regarding ϕ_x itself and then solve it to directly obtain its numerical approximation avoiding an extra finite difference step. As we can see from formula (23) and (25), both these two numerical schemes for approximating $\phi_x(x, t)$ are second order accurate in Δx. However, the proposed approach might be more accurate than the original approach under the same Δx. $|E_j^n|$ is bounded by $I_1 + I_2$ in the original approach and only by I_1 using the proposed approach where $I_1 \triangleq \frac{|1-\lambda^2|}{6} M_{xxxx} |aT| \Delta x^2$ and $I_2 \triangleq \frac{1}{6} M_{xxx} \Delta x^2$. As a result, the larger the ratio of I_2 over I_1, the more accurate our proposed approach than the original approach. In summary, our new Eulerian approach seems to be an improved version of the original Eulerian approach when PDEs are solved using the Lax–Wendroff scheme.

3.3.2 The TVDRK2-WENO5 Discretization

We then discuss the accuracies of those two strategies for computing $\phi_x(x, t)$ assuming that both PDEs (11) and (12) are solved with the TVDRK2-WENO5 scheme. We first review the fifth order WENO scheme for approximating the derivatives of given functions. In particular, to approximate the derivative $f_x(x_i)$ of the function $f(x)$ at a grid point x_i, the WENO scheme proposes to first compute three possible approximations

$$\left(f_x^1\right)_i = \frac{v_1}{3} - \frac{7v_2}{6} + \frac{11v_3}{6}, \quad \left(f_x^2\right)_i = -\frac{v_2}{6} + \frac{5v_3}{6} + \frac{v_4}{3}$$

and

$$\left(f_x^3\right)_i = \frac{v_3}{3} + \frac{5v_4}{6} - \frac{v_5}{6}$$

where $v_1 = D^- f_{i-2}, v_2 = D^- f_{i-1}, v_3 = D^- f_i, v_4 = D^- f_{i+1}, v_5 = D^- f_{i+2}$ for $(f_x)_i^-$ and $v_1 = D^+ f_{i+2}, v_2 = D^+ f_{i+1}, v_3 = D^+ f_i, v_4 = D^+ f_{i-1}, v_5 = D^+ f_{i-2}$ for $(f_x)_i^+$. Here $(f_x)_i^-$ and $(f_x)_i^+$ are appropriately chosen depending on the direction of the characteristic line when solving a Hamiltonian-Jacobian equation. For simplicity, we only consider $(f_x)_i^-$ here.

After that, one approximates $f_x(x_i)$ by

$$(f_x)_i = \omega_1 \left(f_x^1\right)_i + \omega_2 \left(f_x^2\right)_i + \omega_3 \left(f_x^3\right)_i$$

where the $0 \leq \omega_k \leq 1$ are the weights with $\omega_1 + \omega_2 + \omega_3 = 1$ and $(f_x)_i$ denotes the approximation to the exact value of $f_x(x_i)$. In smooth regions, setting $\omega_1 = 0.1, \omega_2 = 0.6$ and $\omega_3 = 0.3$ will give the optimal fifth-order accurate approximation to f_x. When the variation in the underlying function is not smooth enough, we might have other sets of parameters which leads to WENO5. However, for the region where the solution is not smooth enough, the analysis is too complicated and we would not be able to draw any conclusion in the current work. As a result, we have to emphasize that the analysis here is only for the case

when the solution is smooth enough and always use $\omega_1 = 0.1, \omega_2 = 0.6$ and $\omega_3 = 0.3$. With all terms written explicitly in one single formula, we then have

$$(f_x)_i^- = \frac{-0.2f_{i-3} + 1.5f_{i-2} - 6f_{i-1} + 2f_i + 3f_{i+1} - 0.3f_{i+2}}{6\Delta x}.$$

This is a fifth-order accurate approximation because

$$\begin{aligned} &\frac{-0.2f_{i-3} + 1.5f_{i-2} - 6f_{i-1} + 2f_i + 3f_{i+1} - 0.3f_{i+2}}{6\Delta x} \\ &= f_x(x_i) - \frac{1}{60} f_{xxxxxx}(x_i)\Delta x^5 + O(\Delta x^6) \end{aligned} \tag{26}$$

with the use of Taylor's expansion. Actually, formula (26) is true for arbitrary smooth enough function $f(x)$ and we will make use of it multiple times later.

Now we are able to show the accuracies of those two strategies for computing $\phi_x(x,t)$. In the original approach, one first uses the TVDRK2-WENO5 scheme to solve Eq. (11) to approximate $\phi(x,t)$ and then use certain finite difference scheme, e.g. central difference scheme, to obtain the approximation of $\phi_x(x,t)$. Let $\phi(x_i,t^n)$ and ϕ_i^n respectively be the exact value and the numerical approximation of the function $\phi(x,t)$ at the spatial mesh point x_i and time level $t = t^n$. Then ϕ_i^1 is approximated by first solving

$$\frac{\hat{\phi}_i^1 - \phi_i^0}{\Delta t} = -a(\phi_x)_i^0$$

and

$$\frac{\hat{\phi}_i^2 - \hat{\phi}_i^1}{\Delta t} = -a(\phi_x)_i^1$$

and then assigning

$$\phi_i^1 = \phi_i^0 - a\Delta t \cdot \frac{(\phi_x)_i^0 + (\phi_x)_i^1}{2}.$$

Here $\hat{\phi}_i^1$ and $\hat{\phi}_i^2$ represent the intermediate solutions at $t = t^1$ and $t = t^2$ respectively and $(\phi_x)_i^0$ and $(\phi_x)_i^1$ denote the approximation of $\phi_x(x_i,t^0)$ and $\phi_x(x_i,t^1)$, respectively, using the WENO5 discretization. It is obvious that $(\phi_x)_i^0 = \phi_x(x_i,t^0) - \frac{1}{60}\phi_{xxxxxx}(x_i,t^0)\Delta x^5 + O(\Delta x^6)$. The error analysis of $(\phi_x)_i^1$ is a little more complex since it is computed using the WENO5 scheme based on the values of $\hat{\phi}_j^1$ for different j's, rather than the exact values of $\phi(x_j,t^1)$. In particular,

$$(\phi_x)_i^1 = \frac{-0.2\hat{\phi}_{i-3}^1 + 1.5\hat{\phi}_{i-2}^1 - 6\hat{\phi}_{i-1}^1 + 2\hat{\phi}_i^1 + 3\hat{\phi}_{i+1}^1 - 0.3\hat{\phi}_{i+2}^1}{6\Delta x}.$$

Plugging

$$\begin{aligned} \hat{\phi}_j^1 &= \phi_j^0 - a\Delta t(\phi_x)_j^0 \\ &= \phi_j^0 - a\Delta t\left[\phi_x(x_j,t^0) - \frac{1}{60}\phi_{xxxxxx}(x_j,t^0)\Delta x^5 + O(\Delta x^6)\right] \end{aligned}$$

into the above formula gives

$$(\phi_x)_i^1 = \frac{-0.2\phi_{i-3}^0 + 1.5\phi_{i-2}^0 - 6\phi_{i-1}^0 + 2\phi_i^0 + 3\phi_{i+1}^0 - 0.3\phi_{i+2}^0}{6\Delta x}$$

$$
\begin{aligned}
&-a\Delta t\frac{-0.2\phi_x(x_{i-3},t^0)+1.5\phi_x(x_{i-2},t^0)-6\phi_x(x_{i-1},t^0)+2\phi_x(x_i,t^0)+3\phi_x(x_{i+1},t^0)-0.3\phi_x(x_{i+2},t^0)}{6\Delta x}\\
&+O(\Delta t\Delta x^6)\\
&=(\phi_x)_i^0-a\Delta t\left[\phi_{xx}(x_i,t^0)-\frac{1}{60}\phi_{xxxxxx}(x_i,t^0)\Delta x^5+O(\Delta x^6)\right]+O(\Delta t\Delta x^5)\\
&=(\phi_x)_i^0-a\phi_{xx}(x_i,t^0)\Delta t+O(\Delta t\Delta x^5).
\end{aligned}
$$

As a result,

$$
\begin{aligned}
\phi_i^1&=\phi_i^0-a\Delta t\frac{2(\phi_x)_i^0-a\Delta t\phi_{xx}(x_i,t^0)+O(\Delta t\Delta x^5)}{2}\\
&=\phi_i^0-a\Delta t\left[\phi_x(x_i,t^0)-\frac{1}{60}\phi_{x^6}(x_i,t^0)\Delta x^5+O(\Delta x^6)\right]\\
&\quad+\frac{a^2\phi_{xx}(x_i,t^0)}{2}\Delta t^2+O(\Delta t^2\Delta x^5)\\
&=\phi_i^0-a\phi_x(x_i,t^0)\Delta t+\frac{a^2\phi_{xx}(x_i,t^0)}{2}\Delta t^2+O(\Delta t\Delta x^5).
\end{aligned}
\tag{27}
$$

Taking $n=0$ in (16) and then subtracting it from (27) gives

$$
\begin{aligned}
\phi_i^1&=\phi(x_i,t^1)+\frac{a^3\phi_{xxx}(x_i,t^0)}{6}\Delta t^3-\frac{a^4\phi_{xxxx}(x_i,t^0)}{24}\Delta t^4+O(\Delta t\Delta x^5)+O(\Delta t^5)\\
&=\phi(x_i,t^1)+\frac{a^3\phi_{xxx}(x_i,t^0)}{6}\Delta t^3-\frac{a^4\phi_{xxxx}(x_i,t^0)}{24}\Delta t^4+O(\Delta t^5)
\end{aligned}
$$

where the last equality is due to the CFL condition $\Delta t=O(\Delta x)$.
With the mathematical induction we can easily prove that

$$
\begin{aligned}
\phi_i^n&=\phi(x_i,t^n)+\frac{na^3\phi_{xxx}(x_i,t^{n-1})}{6}\Delta t^3-\frac{na^4\phi_{xxxx}(x_i,t^{n-1})}{24}\Delta t^4+O(\Delta t^5)\\
&=\phi(x_i,t^n)+\frac{a\lambda^2T\phi_{xxx}(x_i,t^{n-1})}{6}\Delta x^2\\
&\quad-\frac{a\lambda^3T\phi_{xxxx}(x_i,t^{n-1})}{24}\Delta x^3+O(\Delta x^5)
\end{aligned}
\tag{28}
$$

where $T=n\Delta t$ and $\lambda=\frac{a\Delta t}{\Delta x}$. To approximate $\phi_x(x,t)$, in the original approach, we then use the central difference method and have

$$
\begin{aligned}
(\phi_x)_i^n&=\frac{\phi_{i+1}^n-\phi_{i-1}^n}{2\Delta x}=\frac{\phi(x_{i+1},t^n)-\phi(x_{i-1},t^n)}{2\Delta x}\\
&\quad+\frac{a\lambda^2T\Delta x^2}{6}\frac{\phi_{xxx}(x_{i+1},t^{n-1})-\phi_{xxx}(x_{i-1},t^{n-1})}{2\Delta x}+O(\Delta x^3)\\
&=\phi_x(x_i,t^n)+\frac{\phi_{xxx}(x_i,t^n)}{6}\Delta x^2+\frac{a\lambda^2T\phi_{xxxx}(x_i,t^{n-1})}{6}\Delta x^2+O(\Delta x^3).
\end{aligned}
\tag{29}
$$

However, if we use the idea of the new approach proposed in Sect. 3.1 to approximate $\phi_x(x,t)$, then we just solve Eq. (12) with the TVDRK2-WENO5 scheme and consider the error of $p(x,t)$. From (28), we can immediately have

$$
p_i^n=p(x_i,t^n)+\frac{a\lambda^2Tp_{xxx}(x_i,t^{n-1})}{6}\Delta x^2-\frac{a\lambda^3Tp_{xxxx}(x_i,t^{n-1})}{24}\Delta x^3+O(\Delta x^5).
$$

Noticing that $p = \phi_x$, the above formula can be rewritten as

$$(\phi_x)_i^n = \phi_x(x_i, t^n) + \frac{a\lambda^2 T \phi_{xxxx}(x_i, t^{n-1})}{6}\Delta x^2 + O(\Delta x^3). \tag{30}$$

From formula (29) and (30), we can see that both these two strategies for approximating $\phi_x(x,t)$ are second order accurate in Δx when the corresponding PDEs are solved with the TVDRK2-WENO5 scheme. However, the proposed approach might be more accurate than the original approach under the same Δx. In particular, $|(\phi_x)_i^n - \phi_x(x_i, t^n)|$ is bounded by $I_3 + I_4$ in the original approach and only by I_3 using the proposed approach where $I_3 \triangleq \frac{a\lambda^2 T M_{xxxx}}{6}\Delta x^2$ and $I_4 \triangleq \frac{M_{xxx}}{6}\Delta x^2$. As a result, the larger the ratio of I_4 over I_3, the more accurate our proposed approach than the original approach. This observation is quite similar to that in Sect. 3.3.1 where corresponding PDEs are solved with the Lax–Wendroff scheme. To end this subsection, we want to point out that we still use the WENO5 scheme in the spatial direction in most of our implementations, although the overall accuracy is only second order. It is because that in the region where the solution is not smooth, the WENO5 scheme will lead to a third order discretization rather than giving a fifth order linear discretization and it would better match with the RK2 or RK3 scheme used for the temporal discretization.

3.4 Computational Complexity

We consider the computational complexity of our new Eulerian approach in this section. Let N and M be the discretization size of one spatial dimension and time dimension respectively. In the original Eulerian approach, only the hyperbolic PDE system (4) needs to be solved. Supposing that the computational effort is $h \cdot N^2$ at each time step t_n and summing up the effort in all time steps, the overall computational complexity is $h \cdot M \cdot N^2$. For our new Eulerian approach, however, two hyperbolic PDE systems (7) and (8) need to be solved together. In this case, $2h \cdot N^2$ operations are required at each time step and the total computational complexity would be $2h \cdot M \cdot N^2$. That is, although the proposed approach needs to solve more PDEs, the computational complexity has the same order as that of the original Eulerian approach and both of them have kept the optimal complexity.

4 Numerical Examples

4.1 Linear Advection Equation

In this example, we consider numerical solutions to the linear advection equation with various parameters and situations to demonstrate the performance of the proposed method. The details of this example have been given in Sect. 3.3. For simplicity, we fix the parameters except M_{xxx} and M_{xxxx} which bound the third and the fourth derivative of the solution with respect to x. In particular, we set $\lambda = 0.7$, $a = 0.4$ and $T = 5$ and let M_{xxx} and M_{xxxx} vary by giving different initial conditions $\phi_0(x)$.

4.1.1 $M_{xxx}/M_{xxxx} = O(10^4)$

In this first case, we consider the initial condition $\phi_0(x) = 300x^2(x-1) + 0.001x^4$ whose graph is plotted in Fig. 2a. We then compare $\phi_x(x_j, 5)$ numerically solved by the original approach and the proposed approach, respectively.

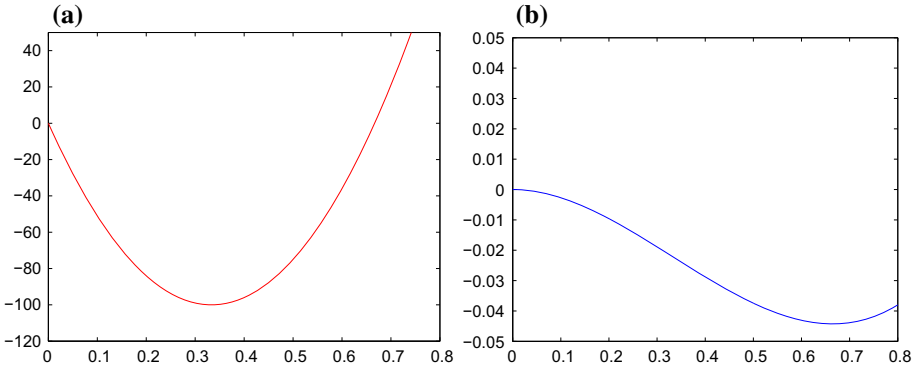

Fig. 2 (Example 4.1.1) The graphs of the function $\phi_0(x) = cx^2(x-1) + 0.001x^4$ with **a** c = 300, **b** 0.3

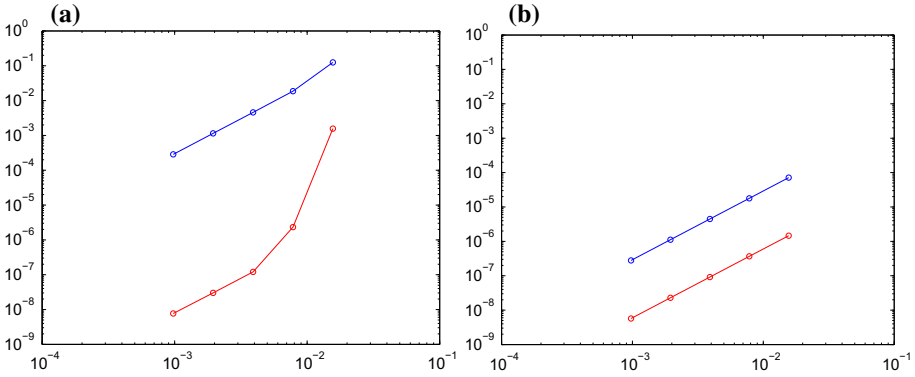

Fig. 3 (Example 4.1.1) L_2 error of $\phi_x(x_j, 5)$ computed using the original approach (in blue lines) and the proposed approach (in red lines) with **a** c = 300, **b** 0.3. In plotting this figure, all corresponding PDEs are solved using the Lax–Wendroff scheme (Color figure online)

In the first implementation, we use the Lax–Wendroff scheme to solve corresponding PDEs as discussed in Sect. 3.3.1. Here we have $M_{xxx} \approx 1800$ and $M_{xxxx} = 0.024$ and thus

$$\frac{I_2}{I_1} = \frac{\frac{1}{6}M_{xxx}}{\frac{1-\lambda^2}{6}M_{xxxx}aT} \approx 7.35 \times 10^4.$$

Figure 3a shows the variation of the L_2 error of $\phi_x(x_j, 5)$ using various Δx's. The blue line corresponds to the solution computed by the original approach and the red line shows the result of the proposed approach. We vary Δx from $1/64$ to $1/1024$ and find that both slopes of the two lines are approximately equal to 2 which indicates that both methods are second-order accurate as expected. As we can see, however, the proposed approach is much more accurate than the original approach under the same mesh size Δx.

We then use the TVDRK2-WENO5 scheme to solve PDEs wherever required as demonstrated in Sect. 3.3.2. Since we have $M_{xxx} \approx 1800$ and $M_{xxxx} = 0.024$, then

$$\frac{I_4}{I_3} = \frac{\frac{1}{6}M_{xxx}}{\frac{\lambda^2}{6}M_{xxxx}aT} \approx 7.65 \times 10^4.$$

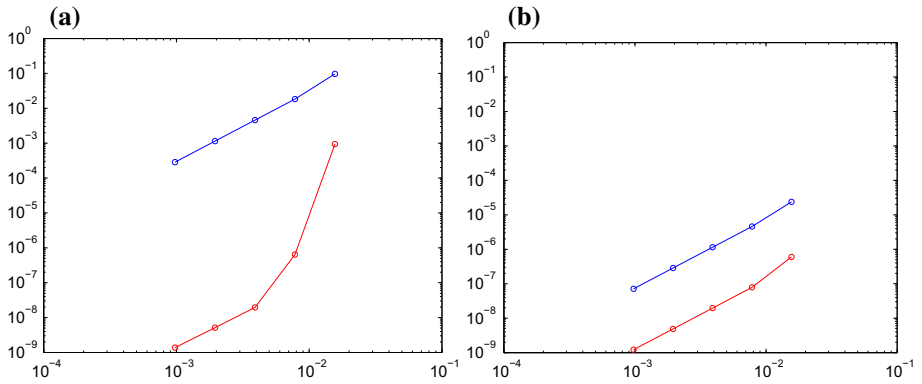

Fig. 4 (Example 4.1.1) L_2 error of $\phi_x(x_j, 5)$ computed using the original approach (in blue lines) and the proposed approach (in red lines) with **a** c = 300, **b** 0.3. In plotting this figure, all corresponding PDEs are solved using the TVDRK2-WENO5 scheme (Color figure online)

Figure 4a shows the variation of the L_2 error of $\phi_x(x_j, 5)$ using various Δx's. The blue line corresponds to the solution computed by the original approach and the red line shows the result of the proposed approach. With Δx varying from 1/64 to 1/1024, we find that both methods are second-order accurate. And as expected, the proposed approach is much more accurate than the original approach under the same mesh size Δx.

4.1.2 $M_{xxx}/M_{xxxx} = O(10^2)$

In this case, we consider $\phi_0(x) = 0.3x^2(x-1) + 0.001x^4$ and the corresponding graph is shown in Fig. 2b. Then we have $M_{xxx} \approx 18.0192$, $M_{xxxx} = 0.024$.

In the first implementation, we use the Lax–Wendroff scheme to solve corresponding PDEs as discussed in Sect. 3.3.1. In this case,

$$\frac{I_2}{I_1} = \frac{\frac{1}{6}M_{xxx}}{\frac{1-\lambda^2}{6}M_{xxxx}|aT|} \approx 7.36 \times 10^2,$$

where the ratio of I_2 over I_1 has greatly decreased compared to the previous example. The variation of the L_2 error of $\phi_x(x_j, 5)$ with respect to Δx is given in Fig. 3b. The red line and the blue line get much closer to each other compared to the previous example as expected.

Also we have used the TVDRK2-WENO5 scheme to solve PDEs in another implementation. Since we have $M_{xxx} \approx 18.0192$ and $M_{xxxx} = 0.024$, then

$$\frac{I_4}{I_3} = \frac{\frac{1}{6}M_{xxx}}{\frac{\lambda^2}{6}M_{xxxx}aT} \approx 7.66 \times 10^2.$$

Figure 4b shows the variation of the L_2 error of $\phi_x(x_j, 5)$ using various Δx's. The blue line corresponds to the solution computed by the original approach and the red line shows the result of the proposed approach. With Δx varying from 1/64 to 1/1024, we find that both methods are second-order accurate. And as expected, the red line and the blue line get much closer compared to the previous example in Sect. 4.1.1.

4.2 Spiral Focus Ridge

This example is taken from [14] which introduces a general benchmark for FTLE computation. In particular, several velocity fields with closed-form formulations of FTLE are given as a ground truth for measuring different numerical approaches for computing FTLE. Here we consider the *spiral focus ridge* to compare our proposed new Eulerian method and the original Eulerian method. Beginning from this example, all PDEs are solved using the TVDRK2-WENO5 scheme in our implementation.

The example starts with a trivial field $\mathbf{v}(\mathbf{x}, t) = (0, x)^T$ and its flow map is given by $(\Phi_{\mathbf{v}})_0^\tau(\mathbf{x}) = (x, y + \tau x)^T$. Let

$$\alpha(\mathbf{x}, t) = \begin{pmatrix} x\cos\gamma - y\sin\gamma \\ x\sin\gamma + y\cos\gamma \end{pmatrix} \text{ and } \beta(\mathbf{x}, t) = \begin{pmatrix} x\cos(-\gamma) - y\sin(-\gamma) \\ x\sin(-\gamma) + y\cos(-\gamma) \end{pmatrix}$$

with $\gamma = p_0/(1 + |x^2 + y^2|)$. Then the velocity field $\mathbf{w}$ of the spiral focus ridge flow is constructed as:

$$\mathbf{w}(\mathbf{x}, t) = (\nabla\beta)^{-1}(\mathbf{x}, t) \cdot \left(\mathbf{v}(\beta(\mathbf{x}, t), t) - \frac{\partial\beta}{\partial t}(\mathbf{x}, t)\right)$$

and the corresponding flow map is given by

$$(\Phi_{\mathbf{w}})_t^{t+\tau}(\mathbf{x}) = \alpha((\Phi_{\mathbf{v}})_t^{t+\tau}(\beta(\mathbf{x}, t)), t + \tau).$$

In our numerical implementation, we set $p_0 = 12$, $t = 0$, $\tau = 5$ and the computational domain by $[-1, 1] \times [-1, 1]$. The exact FTLE field is plotted in Fig. 5a. The FTLE field computed using the original Eulerian approach and our new Eulerian approach are shown in Fig. 5b, c, respectively. The computational domain is discretized by 257×257 grid points and thus gives a mesh size $\Delta x = \Delta y = 1/128$. To better compare the solutions, we plot the $y = 0$ cross-sections of Fig. 5b, c in Fig. 6a, b, respectively. The corresponding cross-sections along $y = -0.5$ of Fig. 5b, c are plotted in red lines in Fig. 6c, d, respectively. The exact solutions are plotted in blue lines in each subfigure of Fig. 6. We can see from Fig. 6 that the original Eulerian approach and our new approach both match well with the exact solution at locations with relatively small FTLE values. However, the proposed approach is much more accurate than the original approach at locations with relatively big FTLE values, e.g. near the FTLE ridge.

4.3 Double-Gyre Flow

This example is taken from [27] to describe a periodically varying double-gyre. The flow is modeled by the following stream-function $\psi(x, y, t) = A\sin[\pi f(x, t)]\sin(\pi y)$ where

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

In this example, we use $A = 0.1$, $\omega = 2\pi/10$, $\epsilon = 0.1$ and the computational domain Ω is $[0, 2] \times [0, 1]$. As a result, this flow is a periodic flow with the period $T_m = 10$. We use the new Eulerian approach proposed in Sect. 3.2 to compute the backward FTLE $\sigma_T^0(\mathbf{x})$ from $t = T$ to $t = 0$ where $T = 2^4 T_m = 160$.

We first solve PDE systems (4), (7) and (8) together from $t = 0$ to $t = T_m = 10$ with initial conditions $\Psi(\mathbf{x}, 0) = \mathbf{x}$, $(\phi_x(\mathbf{x}, 0), \phi_y(\mathbf{x}, 0)) = (1, 0)$ and $(\psi_x(\mathbf{x}, 0), \psi_y(\mathbf{x}, 0)) =$

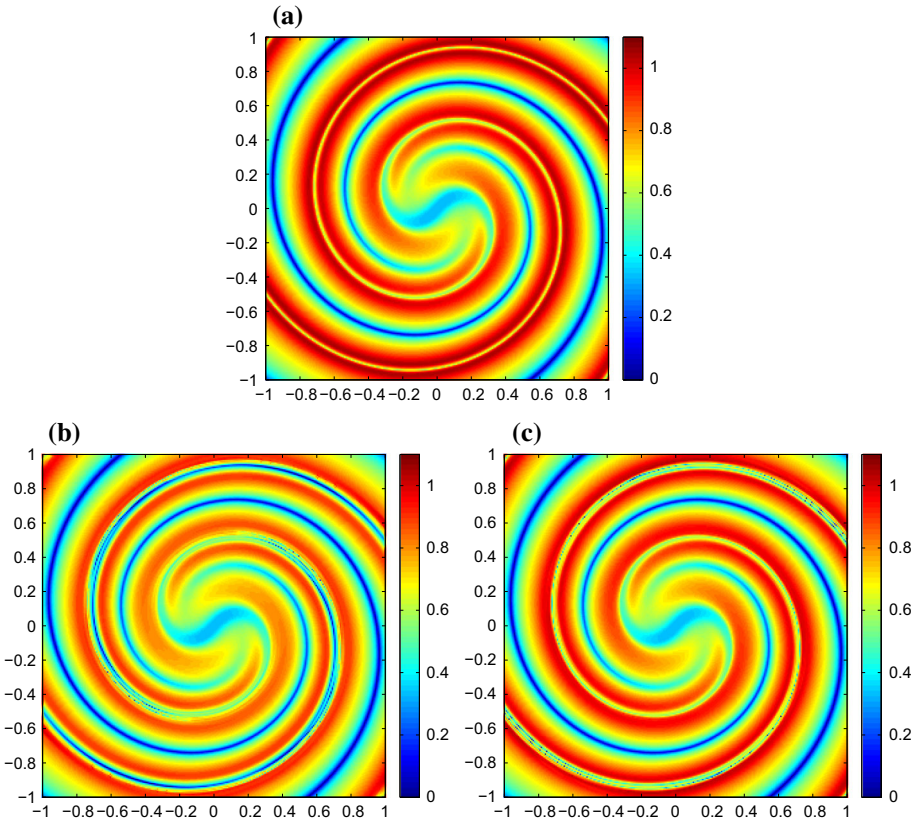

Fig. 5 (Example 4.2) The FTLE $\sigma_0^5(\mathbf{x})$. **a** The exact solution, **b** the numerical solution computed by the original approach and **c** the proposed approach

$(0, 1)$, respectively. Then $\Psi(\mathbf{x}, 10)$ and $J_{10}^0(\mathbf{x})$ are obtained and iterate 4 times as stated in step 2 of Algorithm 2 to obtain $J_{160}^0(\mathbf{x})$. The backward FTLE $\sigma_{160}^0(\mathbf{x})$ is then computed as $\sigma_{160}^0(\mathbf{x}) = \frac{1}{160} \ln \sqrt{\lambda_{max}[\Delta_{160}^0(\mathbf{x})]}$ where $\Delta_{160}^0(\mathbf{x}) = [J_{160}^0(\mathbf{x})]^* J_{160}^0(\mathbf{x})$.

In Fig. 7a, b we plot the backward FTLE σ_{160}^0 computed with the original Eulerian approach and our new Eulerian approach respectively. As we can see in those four connected subregions, the FTLE values are relatively small and numerical solutions from the two approaches are very close. However, the FTLE values computed with the two approaches differ a lot outside the four subregions where the FTLE values are relatively large. Take a cross-section $y = 0.4688$ as shown in Fig. 8a, then two line segments $[0.4, 0.65]$ and $[1.32, 1.65]$ lie within the connected subregions. Figure 8b, c respectively show the FTLE σ_{160}^0 on this cross-section computed with the original Eulerian approach and our new Eulerian approach. In Fig. 8b, red, green and blue lines represent the numerical solutions computed using the original Eulerian approach with the mesh size $\Delta x = \Delta y = 1/128$, $\Delta x = \Delta y = 1/256$ and $\Delta x = \Delta y = 1/512$, respectively. Figure 8c shows the solutions of our proposed approach. In particular, the blue line corresponds to the solution with the finest mesh size $\Delta x = \Delta y = 1/512$. The red and green lines represent the difference of the solutions with $\Delta x = \Delta y = 1/128$ and $\Delta x = \Delta y = 1/256$, respectively, from the solution of the finest mesh size. As can be seen

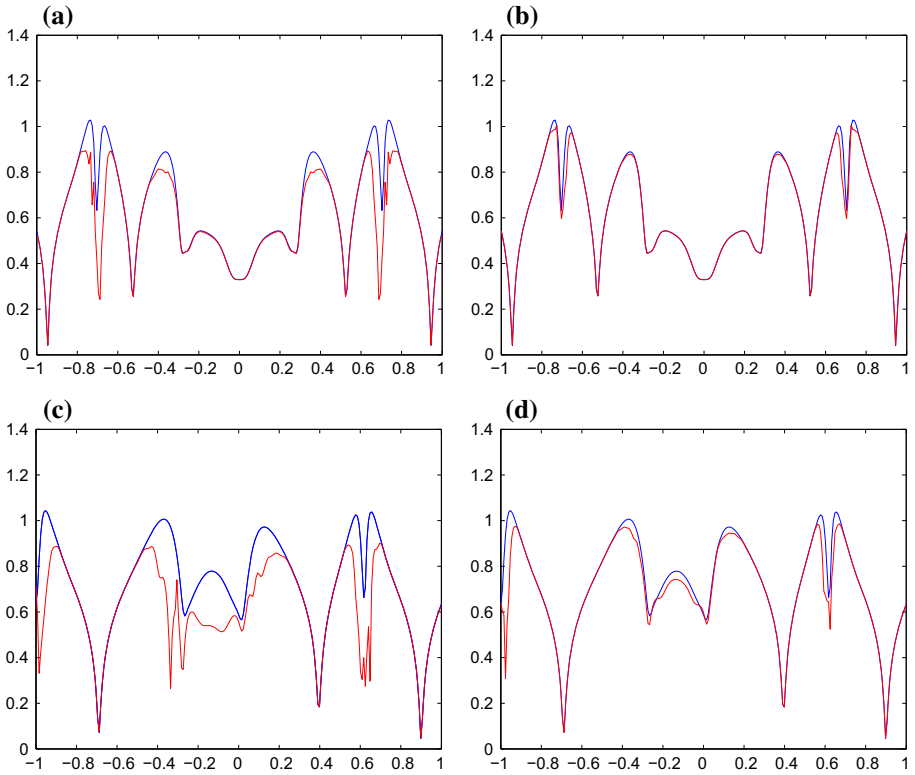

Fig. 6 (Example 4.2) **a** The solution computed by the original approach along $y = 0$. **b** The solution computed by the proposed approach along $y = 0$. **c** The solution computed by the original approach along $y = -0.5$. **d** The solution computed by the proposed approach along $y = -0.5$. The exact solution is plotted in blue (Color figure online)

from these figures, the FTLE values on the segments [0.4, 0.65] and [1.32, 1.65], which are within the connected subregions, are always approximated well with both the two approaches and all mesh sizes. However, except on the two segments the computed FTLE values differ a lot between the two approaches. From Fig. 8b we can see that as the underlying mesh size becomes smaller and smaller, the FTLE values outside the two segments grow larger and larger and surely get closer and closer to the exact solution. In Fig. 8c, however, as the mesh size changes, the FTLE values do not have obvious trend to which direction and we conclude that they are already near the exact solution. From this point of view, our new approach is much more accurate than the original Eulerian approach especially at locations with large FTLE values.

4.4 Velocity Field at Discrete Locations

In our new Eulerian approach to compute the FTLE, we need not only velocity data defined on mesh points, but also the Jacobian of the velocity field as inferred in the PDE systems (7) and (8). However, the Jacobian of the velocity field at each mesh point is not always available. As an alternative, we propose to use the central difference method to compute the Jacobian of the velocity field. The following numerical results will show the effectiveness.

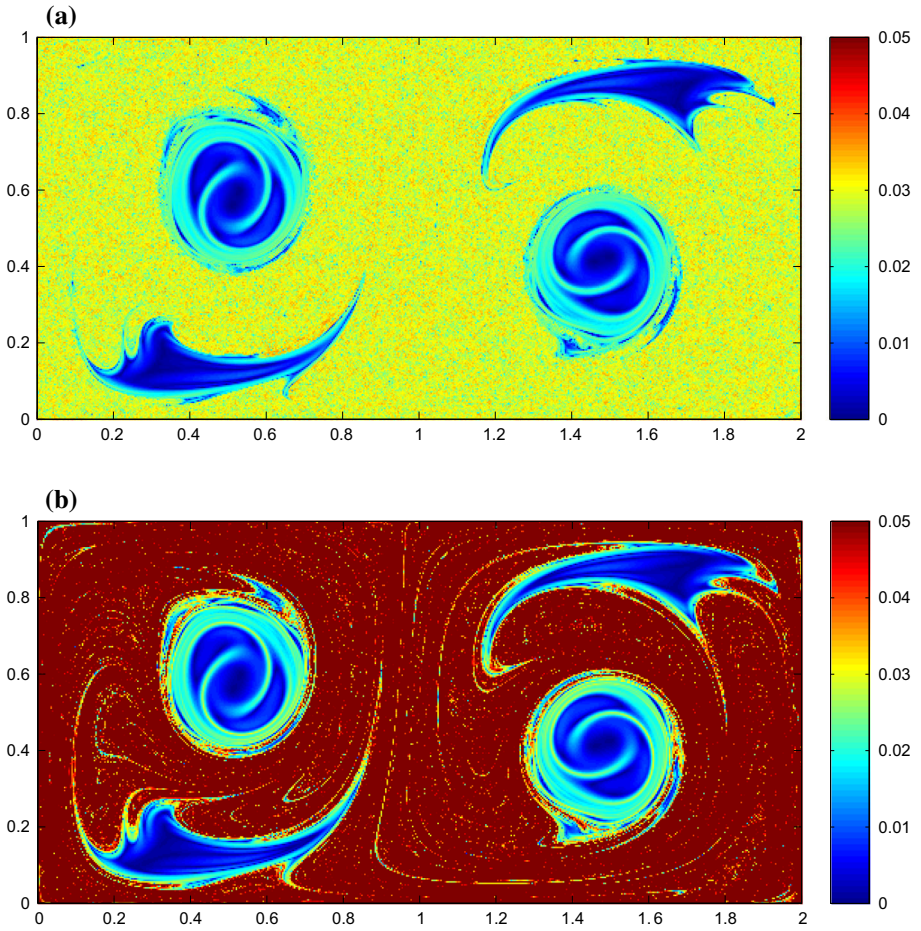

Fig. 7 (Example 4.3) The backward FTLE σ^0_{160} computed with mesh size $\Delta x = \Delta y = 1/256$ using **a** the original Eulerian approach, **b** our new Eulerian approach

Figure 9a shows the forward FTLE $\sigma^5_0(\mathbf{x})$ of the *spiral focus ridge* flow computed using our new Eulerian approach where the Jacobian of the velocity field is obtained by the central difference method. Here all the parameters are the same as those in Fig. 5. Figure 9a shows almost the same result with Fig. 5c where the Jacobian of the velocity field is exactly given. The red lines in Fig. 9b, c are respectively the $y = 0$ and $y = -0.5$ cross-sections of Fig. 9a while the blue lines in them are exact solutions. Compared with Fig. 6, we can find that the FTLE computed with our new Eulerian approach, no matter the Jacobian of the velocity field is given exactly or from central difference, is more closed to the exact solution than that of the original Eulerian approach. However, the two solutions of our new approach, although with different treatments of the Jacobian of the velocity field, are very close to each other. Figure 9d compares the numerical errors of these two solutions at the $y = 0$ cross-section and we can see that the one with exact Jacobian of velocity field (plotted in blue) is only a little more accurate than the other one (plotted in red).

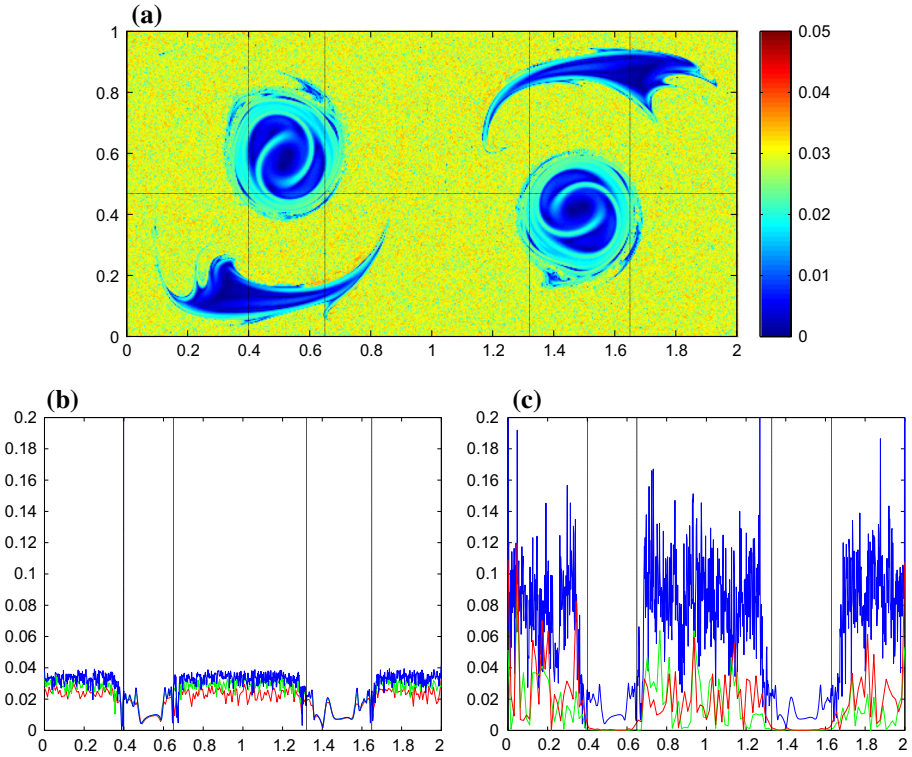

Fig. 8 (Example 4.3) **a** The horizontal black line is $y = 0.4688$ and the two line segments between vertical black lines are within the circular connected components; the FTLE σ^0_{160} on the $y = 0.4688$ cross-section computed with **b** the original Eulerian approach, **c** our new approach (Color figure online)

Also in Fig. 10a we plot the backward FTLE σ^0_{160} of the double gyre flow computed using our new Eulerian approach where the Jacobian of the velocity field is obtained by the central difference method and all the parameters are set the same as in Fig. 7. We can see that Fig. 10a is almost the same as Fig. 7b which is also the solution of our new Eulerian approach yet with the Jacobian of the velocity field exactly given at mesh points. As a result, these two solutions of our new Eulerian approach with different treatments of the Jacobian of the velocity field are both more accurate than the solution of the original Eulerian approach. To better demonstrate this, we plot the $y = 0.4688$ cross-sections of these altogether three different solutions in Fig. 10b where the red line corresponds to the original Eulerian approach, the blue line corresponds to our new Eulerian approach with exact Jacobian of the velocity field and the black line corresponds to the remaining solution. It's easy to see that the blue line and the black line almost coincide with each other which implies that the two solutions of our new Eulerian approach, although the Jacobian of the velocity field are treated differently, are almost the same accurate. The difference of them is plotted in Fig. 10c.

From these two examples, we can find that the solution of our new Eulerian approach, even when the Jacobian of the velocity field is obtained from the central difference method, is more accurate (especially at ridge locations) than that of the original Eulerian approach in which the finite difference step is used to obtain the Jacobian of the flow map.

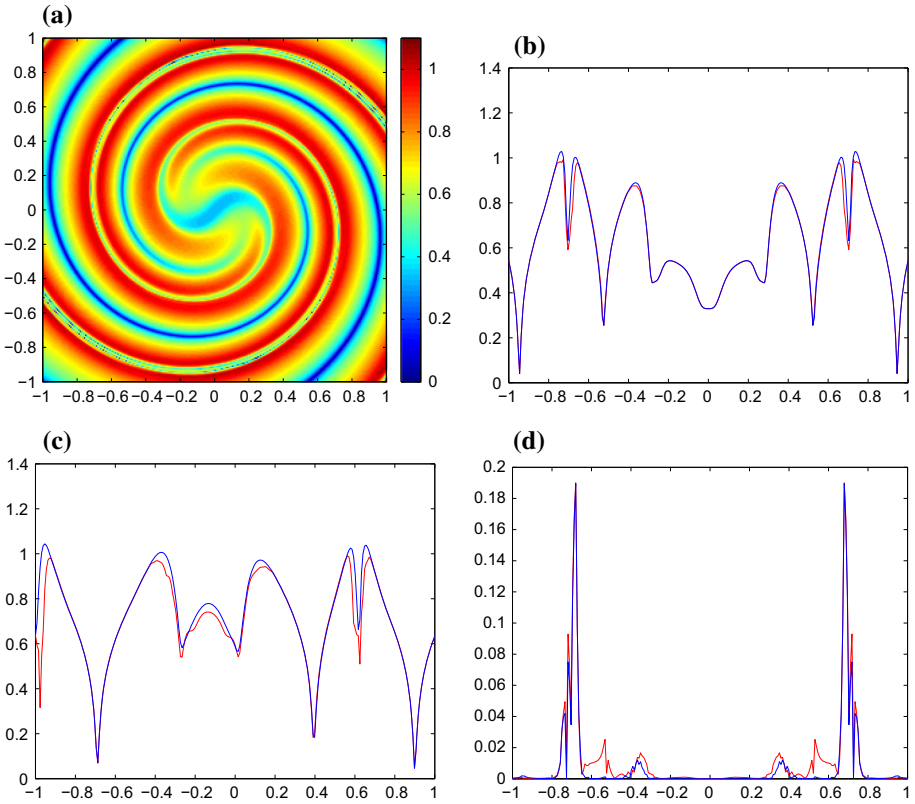

Fig. 9 (Example 4.4 for the spiral flow) **a** The FTLE $\sigma_0^5(\mathbf{x})$ computed with our new Eulerian approach where the Jacobian of the velocity field is from central difference; **b**, **c** plot respectively the $y = 0$ and $y = -0.5$ cross-sections of (**a**) in red lines while exact solutions are plotted in blue lines; **d** numerical errors of the two solutions using the new Eulerian approach at the $y = 0$ cross-section where the blue line corresponds to the one with exact Jacobian of the velocity field while the red line corresponds to the one with the Jacobian from central difference (Color figure online)

4.5 Application to the Variation of the Integral Over Area of Level Surfaces (VIALS)

In [31] we have proposed a novel Eulerian tool to study complicated dynamical systems based on the average growth in the surface area of a family of level surfaces represented implicitly by a level set function. Since this proposed quantity determines the temporal variation of the averaged surface area of all level surfaces, we named the quantity the Variation of the Integral over Area of Level Surfaces (VIALS). We have shown in [31] that the VIALS is a nice candidate for quantifying the level of short-time mixing of given dynamical systems. In particular, the VIALS is defined as

$$\mathcal{L}(t;\, f) = \frac{\displaystyle\int_{-\infty}^{+\infty} \int_{C(t,k;f)} dsdk}{\displaystyle\int_{-\infty}^{+\infty} \int_{C(0,k;f)} dsdk}. \tag{31}$$

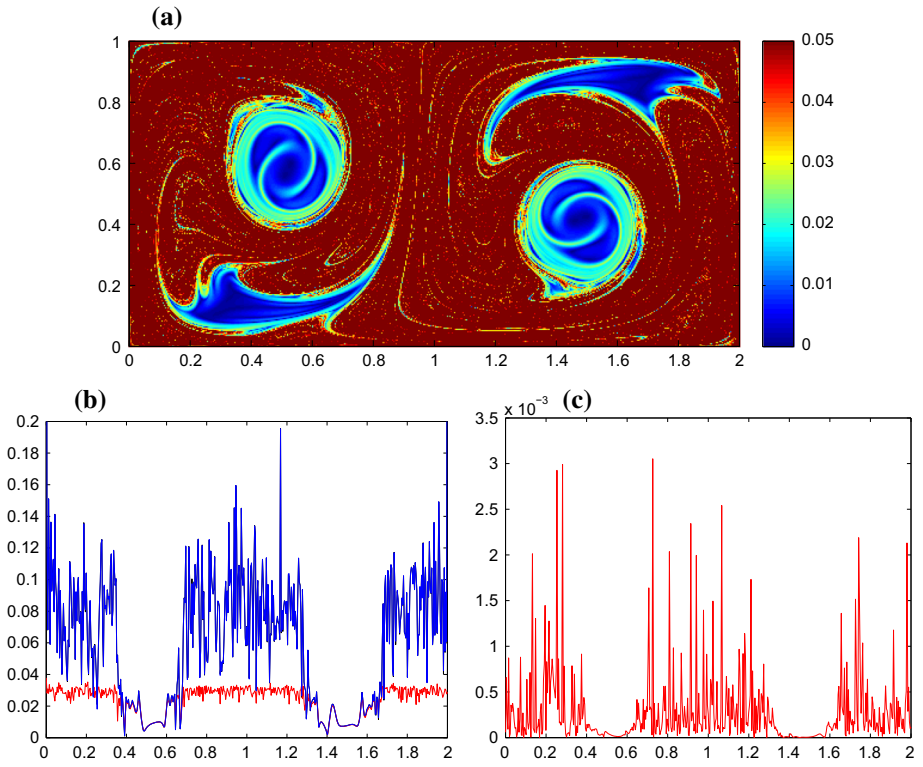

Fig. 10 (Example 4.4 for the double gyre flow) **a** The backward FTLE σ^0_{160} computed using our new Eulerian approach where the Jacobian of the velocity field is from central difference. **b** The $y = 0.4688$ cross-sections of σ^0_{160} computed using the original Eulerian approach (red) and the new Eulerian approach with different treatments of the Jacobian of the velocity field (blue and black). **c** The difference between the solutions from the original approach and the proposed approach (Color figure online)

where

$$C(t, k; f) = \{\mathbf{x} : f(\mathbf{x}, t) = k\}$$

and f is a level set function satisfying also the Liouville equations (4). With the use of the coarea formula, we can rewrite (31) as

$$\mathcal{L}(t; f) = \frac{\int_\Omega |\nabla f(\mathbf{x}, t)| d\mathbf{x}}{\int_\Omega |\nabla f(\mathbf{x}, 0)| d\mathbf{x}} \tag{32}$$

which is much easier to numerically compute in practice.

Since mesh refinement will reduce the numerical dissipation in the solution to the level set equation (4), the length of each individual level contour will increase in the number of mesh points in each spatial direction. Now, because the VIALS takes an average of all level surfaces of $f(\mathbf{x}, t)$, we do observe that the quality does increase in the number of mesh points in the computational domain, as shown in Fig. 11a. It shows the VIALS from $t = 0$ to $t = 10$ of the double gyre flow with $\epsilon = 1.2$ computed on various mesh sizes for the initial level set

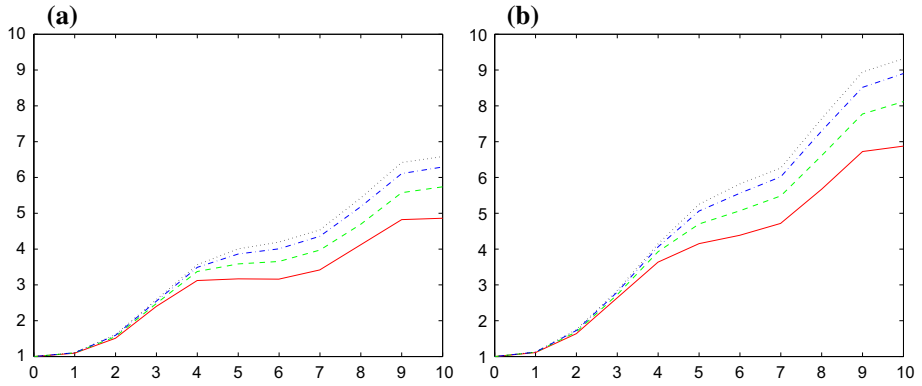

Fig. 11 (Example 4.5) The VIALS of the double-gyre flow with $t \in [0, 10]$ and $\epsilon = 1.2$ computed using **a** the original approach from [31] and **b** our proposed approach on various mesh sizes $\Delta x = \Delta y = 1/128$ (red color), $\Delta x = \Delta y = 1/256$ (green color), $\Delta x = \Delta y = 1/512$ (blue color) and $\Delta x = \Delta y = 1/1024$ (black color), respectively (Color figure online)

function $f(\mathbf{x}, 0) = x$. However, as can be seen from formula (32), the computation of the VIALS $\mathcal{L}(t; f)$ actually only depends on the partial derivatives of f, rather than f itself. As a result, we can use our new approach proposed in Sect. 3 to recompute $\mathcal{L}(t; f)$. Figure 11b shows the corresponding results computed using our proposed method. We can see that the VIALS computed with our new approach using even the coarsest mesh $\Delta x = \Delta y = 1/128$ has already achieved similar accuracy like the one computed by the original approach using the finest mesh $\Delta x = \Delta y = 1/1024$.

Acknowledgements The work of You was supported by the National Natural Science Foundation of China (No. 11701287) and the Natural Science Foundation of Jiangsu Province (No. BK20171071). The work of Leung was supported in part by the Hong Kong RGC Grants 16303114 and 16309316.

References

1. Badas, M.G., Domenichini, F., Querzoli, G.: Quantification of the blood mixing in the left ventricle using finite time Lyapunov exponents. Meccania **52**, 529–544 (2017)
2. Candès, E.J., Ying, L.: Fast geodesics computation with the phase flow method. J. Comput. Phys. **220**, 6–18 (2006)
3. Cardwell, B.M., Mohseni, K.: Vortex shedding over two-dimensional airfoil: where do the particles come from? AIAA J. **46**, 545–547 (2008)
4. Chavent, G., Cockburn, B.: The local projection p0 p1-discontinuous-Galerkin finite element method for scalar conservation laws. RAIRO Modél. Math. Anal. Numér. **23**, 565–592 (1989)
5. Cockburn, B., Shu, C.-W.: The Runge–Kutta discontinuous Galerkin finite element method for conservation laws V: multidimensional systems. J. Comput. Phys. **141**, 199–224 (1998)
6. Cockburn, B., Shu, C.-W.: Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. J. Sci. Comput. **16**, 173–261 (2001)
7. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Vis. Comput. Graph. **13**, 1464–1471 (2007)
8. Garth, C., Li, G.S., Tricoche, X., Hansen, C.D., Hagen, H.: Visualization of Coherent Structures in Transient 2D Flows. Springer, Berlin (2009)
9. Gottlieb, S., Shu, C.-W.: Total variation diminishing Runge–Kutta schemes. Math. Comput. **67**, 73–85 (1998)
10. Green, M.A., Rowley, C.W., Smiths, A.J.: Using hyperbolic Lagrangian coherent structures to investigate vortices in biospired fluid flows. Chaos **20**, 017510 (2010)

11. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D **149**, 248–277 (2001)
12. Haller, G.: Lagrangian structures and the rate of strain in a partition of two-dimensional turbulence. Phys. Fluids A **13**, 3368–3385 (2001)
13. Haller, G., Yuan, G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. Physica D **147**, 352–370 (2000)
14. Kuhn, A., Rossl, C., Weinkauf, T., Theisel, H.: A benchmark for evaluating FTLE computations. In: IEEE Pacific Visualization Symposium, pp. 121–128. IEEE Computer Society (2012)
15. Lekien, F., Leonard, N.: Dynamically consistent Lagrangian coherent structures. In: Experimental Chaos: 8-th Experimental Chaos Conference, pp. 132–139 (2004)
16. Lekien, F., Ross, S.D.: The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds. Chaos **20**, 017505 (2010)
17. Lekien, F., Shadden, S.C., Marsden, J.E.: Lagrangian coherent structures in n-dimensional systems. J. Math. Phys. **48**, 065404 (2007)
18. Leung, S.: An Eulerian approach for computing the finite time Lyapunov exponent. J. Comput. Phys. **230**, 3500–3524 (2011)
19. Leung, S.: The backward phase flow method for the Eulerian finite time Lyapunov exponent computations. Chaos **23**, 043132 (2013)
20. Leung, S., Qian, J.: Eulerian Gaussian beams for Schrödinger equations in the semi-classical regime. J. Comput. Phys. **228**, 2951–2977 (2009)
21. Lipinski, D., Mohseni, K.: Flow structures and fluid transport for the hydromedusae *Sarsia tubulosa* and *Aequorea victoria*. J. Exp. Biol. **212**, 2436–2447 (2009)
22. Liu, X.D., Osher, S.J., Chan, T.: Weighted essentially nonoscillatory schemes. J. Comput. Phys. **115**, 200–212 (1994)
23. Lukens, S., Yang, X., Fauci, L.: Using Lagrangian coherent structures to analyze fluid mixing by cillia. Chaos **20**, 017511 (2010)
24. Osher, S.J., Sethian, J.A.: Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations. J. Comput. Phys. **79**, 12–49 (1988)
25. Qian, J., Leung, S.: A local level set method for paraxial multivalued geometric optics. SIAM J. Sci. Comput. **28**, 206–223 (2006)
26. Sapsis, T., Haller, G.: Inertial particle dynamics in a hurricane. J. Atmos. Sci. **66**, 2481–2492 (2009)
27. Shadden, S.C., Lekien, F., Marsden, J.E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Physica D **212**, 271–304 (2005)
28. Shu, C.W.: Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In: Cockburn, B., Johnson, C., Shu, C.W., Tadmor, E. (eds.) Advanced Numerical Approximation of Nonlinear Hyperbolic Equations. Lecture Notes in Mathematics, vol. 1697, pp. 325–432. Springer, Berlin (1998)
29. Tang, W., Chan, P.W., Haller, G.: Accurate extraction of Lagrangian coherent structures over finite domains with application to flight data analysis over Hong Kong international airport. Chaos **20**, 017502 (2010)
30. Tang, W., Peacock, T.: Lagrangian coherent structures and internal wave attractors. Chaos **20**, 017508 (2010)
31. You, G., Leung, S.: VIALS: an Eulerian tool based on total variation and the level set method for studying dynamical systems. J. Comput. Phys. **266**, 139–160 (2014)
32. You, G., Leung, S.: Eulerian based interpolation schemes for flow map construction and line integral computation with applications to coherent structures extraction. J. Sci. Comput. **74**, 70–96 (2018)
33. You, G., Wong, T., Leung, S.: Eulerian methods for visualizating continuous dynamical systems using Lyapunov exponents. SIAM J. Sci. Comput. **39**(2), A415–A437 (2017)