J Sci Comput (2018) 74:70–96
https://doi.org/10.1007/s10915-017-0424-9

Eulerian Based Interpolation Schemes for Flow Map Construction and Line Integral Computation with Applications to Lagrangian Coherent Structures Extraction

Guoqiao You[1] · **Shingyu Leung**[2]

Received: 28 December 2016 / Revised: 19 March 2017 / Accepted: 21 March 2017 /
Published online: 27 March 2017

Abstract We propose and analyze a new class of Eulerian methods for constructing both the *forward* and the *backward* flow maps of sufficiently smooth dynamical systems. These methods improve previous Eulerian approaches so that the computations of the *forward* flow map can be done *on the fly* as one imports or measures the velocity field forward in time. Similar to typical Lagrangian or semi-Lagrangian methods, the proposed methods require an interpolation at each step. Having said that, the Eulerian method interpolates d components of the flow maps in the d dimensional space but does not require any $(d+1)$-dimensional spatial-temporal interpolation as in the Lagrangian approaches. We will also extend these Eulerian methods to compute line integrals along any Lagrangian particle. The paper gives a computational complexity analysis and an error estimate of these Eulerian methods. The method can be applied to a wide range of applications for flow map constructions including the finite time Lyapunov exponent computations, the coherent ergodic partition, and high frequency wave propagations using geometric optic.

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

1 Introduction

Given a smooth enough flow field and an integrable function, we propose and analyze numerical methods to solve the following two related yet different problems.

✉ Shingyu Leung
masyleung@ust.hk

Guoqiao You
270217@nau.edu.cn

[1] School of Science, Nanjing Audit University, Nanjing 211815, China

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

Problem F: Starting from an initial particle location at the initial time, we ask for the terminal location of the particle at the later time and also the line integral of the given function along the particle trajectory under the flow.
Problem B: Fixing a terminal particle location at the final time, we ask for the starting location of the particle at the initial time and also the line integral of the given function along the particle trajectory under the flow.

These two fundamental problems are important in many research fields such as modeling in the high frequency wave propagation [3,25,36], traveltime tomography in seismology [22], semiclassical solutions to the Schrödinger equations [23,24], and the level set equation in the level set method [8,32,33,37]. They are also building blocks of various recent numerical tools for flow visualizations including the finite time Lyapunov exponent (FTLE) [10–12, 16,21,38], the finite size Lyapunov exponent (FSLE) [1,2,4,13,19], the so-called coherent ergodic partition developed in [43], and some studies of chaotic mixing from atmospheric modelings [27,28,41].

There are in general two classes of methods to solve these two problems. Because information is required along each of the Lagrangian particle trajectories, one approach is the Lagrangian tracking methods in which we solve the ordinary differential equations (ODEs) governing both the dynamic of the Lagrangian particle and the line integral along the trajectory. These methods fit perfectly to Problem F. When applied to Problem B, we search for an initial takeoff location so that ray shooting from this position will reach the given terminal particle location. Therefore, the usual Lagrangian methods convert Problem B to a root-finding problem in the high dimensional space. Relating to this class of methods are the semi-Lagrangian schemes [6,17,18,41] which trace characteristics backward in time leading to the solution to Problem B. Since the ODE timestep in these methods is restricted solely by the stiffness of the ODE system based on the velocity field, one might seem to be able to apply a much larger ODE timestep $\Delta t > O(\Delta x)$ 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 behavior or the numerical integrator may even lead to unreliable solutions. For example, we consider the scalar ODE given by $x'(t) = u(t, x(t)) = -x(t)$. The exact solution of this system can be easily found $x(t) = x(0)e^{-t}$ and so all particle trajectories converge to $x(t) = 0$ as $t \to \infty$. Applying the second order Runge–Kutta (RK2) method, we have

$$x^{n+1} = \left(1 - \Delta t + \frac{\Delta t^2}{2}\right) x^n .$$

In particular, we have

$$x^{n+1} = \begin{cases} (0.905)^{n+1} x^0 \text{ for } \Delta t = 1.9, \\ (1.105)^{n+1} x^0 \text{ for } \Delta t = 2.1. \end{cases}$$

We note that when we pick a step size slightly larger than that constrained by the stability condition given by $\Delta t < 2$, the quantitative behavior of the solution is completely wrong in which all particle trajectories are diverging from the origin.

The second class of methods is the Eulerian approach which embeds the solutions to these problems in a higher dimensional space. The idea is to determine the evolution using the level set method [32,33,37] so that the flow map satisfies a Liouville equation. The resulting hyperbolic partial differential equations (PDEs) can be solved by any well-established robust and high order accurate numerical methods. In a series of studies [22–25,36], we have developed some Eulerian methods to solve Problem B. To solve Problem F, on the other hand, [20,21,43] have suggested to reverse the time and solve the PDE backward in time.

In particular, to solve Problem F 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 initial condition given at $t = T$. This implementation is numerically inconvenient, 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 field at all time steps have to be stored in the disk which might not be practical at all. In this paper, we are going to propose and analyze a class of Eulerian interpolation methods to solve Problem F *on the fly* so that the PDE is solved forward in time.

There are several main characteristics of our proposed Eulerian interpolation schemes. These methods are developed based on standard Eulerian approaches where the performance of the algorithm is rather well-studied. These methods require derivatives of the velocity only at mesh locations which can be easily computed using any well-developed finite difference algorithm including the ENO and WENO [26,39]. We, in this work, are not trying to improve these methods but are instead proposing a new formulation which can naturally solve both Problem B and Problem F. Indeed these schemes still require an interpolation at each step to construct the overall flow map. However, these interpolation problems are standard ones where we interpolate at non-grid locations using data given on a uniform Cartesian mesh. There are well-developed (monotone) numerical schemes for this interpolation step such as the function pchip/interp1/interp2/interp3/interpn developed in MATLAB. We refer the interested readers to [7,9,14,35,40] and the references thereafter.

The paper is organized as follows. In Sect. 2, we summarize the Lagrangian and the semi-Lagrangian approaches to both Problem F and Problem B. Our proposed Eulerian approaches will be given in Sect. 3. Some further analysis are given in Sect. 4. Finally, in Sect. 5 we will apply the techniques to various examples in dynamical system visualization and Lagrangian coherent structure (LCS) extraction to demonstrate the effectiveness and importance of the proposed Eulerian interpolation schemes for constructing the flow maps and line integrals.

2 Background

In this section, we first review the traditional Lagrangian-type approaches for the Problem F and also Problem B.

2.1 The Lagrangian Approach for the Problem F

We consider numerical methods for computing a Lagrangian quantity following the particle trajectory in a given smooth flow field. There are in general two different classes of mathematical formulation. The first category is the Lagrangian formulation which mathematically formulates the problem using the following system of ordinary differential equations (ODEs)

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

$$\dot{f}(t) = g(\mathbf{x}(t), t)\,, \tag{2}$$

with a given smooth enough velocity field $\mathbf{u} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and a given *initial* or *terminal* boundary condition which depends on the particular application. For example, one obtains the trajectory of a particle and the forward flow map $\Phi_0^T : \mathbb{R}^d \to \mathbb{R}^d$ by solving (1) up to $t = T$ with the initial condition $\mathbf{x}(t = 0) = \mathbf{x}_0$. In particular, the forward flow map is given by $\Phi_0^T(\mathbf{x}_0) = \mathbf{x}(T)$ with $\mathbf{x}(t)$ satisfying (1). Solving together with (2), we have

$$f(T) = f(0) + \int_0^T g(\mathbf{x}(s), s)\, ds$$

with some initial condition $f(0)$.

For example, if we are applying the simple first order Euler method, we have the updating formula

$$\frac{\mathbf{x}_{i,j}(t_{n+1}) - \mathbf{x}_{i,j}(t_n)}{\Delta t} = \mathbf{u}(\mathbf{x}_{i,j}(t_n), t_n)\,.$$

This implies that the forward flow map from t_n to t_{n+1} is given by

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

If TVD-RK2 is used instead to obtain high order accurate solution, we obtain

$$\begin{aligned}
\frac{\hat{\mathbf{x}}_{i,j}(t_{n+1}) - \mathbf{x}_{i,j}(t_n)}{\Delta t} &= \mathbf{u}(\mathbf{x}_{i,j}(t_n), t_n) \\
\frac{\hat{\mathbf{x}}_{i,j}(t^{n+2}) - \hat{\mathbf{x}}_{i,j}(t_{n+1})}{\Delta t} &= \mathbf{u}(\hat{\mathbf{x}}_{i,j}(t_{n+1}), t_{n+1}) \\
\mathbf{x}_{i,j}(t_{n+1}) &= \frac{1}{2}\left(\mathbf{x}_{i,j}(t_n) + \hat{\mathbf{x}}_{i,j}(t^{n+2})\right)\,.
\end{aligned}$$

If we need only the particle identity, we take $g(\mathbf{x}, t) = 0$ and simply use the flow map to construct

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

with the function $\mathbf{x}_{i,j}$ satisfying the ODE system (1) and the *initial* condition $\mathbf{x}_{i,j}(t_n) = \mathbf{x}_{i,j}$.

2.2 The Semi-Lagrangian Approach for the Problem B

One major issue of this ray tracing approach is that the solution at the final time $t = T$ will not be uniform since it depends on the distribution of the arrival location of rays, as shown in Fig. 1a. One resolution is the semi-Lagrangian methods which propose to trace the characteristics backward in time. Mathematically, we impose a *terminal* boundary condition to (1) at $t = T$ and solve the ODE backward in time to obtain the solution at $t = 0$, as shown in Fig. 1b. This gives the backward flow map $\Phi_T^0 : \mathbb{R}^d \to \mathbb{R}^d$ such that $\Phi_T^0(\mathbf{x}(T)) = \mathbf{x}(0)$ with $\mathbf{x}(t)$ satisfying (1). In particular, the backward line integral along the characteristic of a general function g is computed as

$$\begin{aligned}
f(\mathbf{x}_{i,j}, t_{n+1}) &= f(\Phi_{t_{n+1}}^{t_n}(\mathbf{x}_{i,j}), t_n) - \int_{t_{n+1}}^{t_n} g(\Phi_{t_{n+1}}^{s}(\mathbf{x}_{i,j}), s)\, ds \\
&= f(\Phi_{t_{n+1}}^{t_n}(\mathbf{x}_{i,j}), t_n) + \int_{t_n}^{t_{n+1}} g(\Phi_{t_{n+1}}^{s}(\mathbf{x}_{i,j}), s)\, ds\,.
\end{aligned}$$

Again, if we need only the particle identity, we take $g(\mathbf{x}, t) = 0$ and simply use the flow map to construct

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

with $\mathbf{x}_{i,j}$ satisfying the ODE system (1) with the *terminal* boundary condition $\mathbf{x}_{i,j}(t_{n+1}) = \mathbf{x}_{i,j}$. To solve this ODE backward in time, we can use any well-developed numerical integrators to first obtain the backward flow map $\Phi_{t_{n+1}}^{t_n}$ and then interpolate the solution at $t = t_n$. Interpolation plays an extremely important role in these Lagrangian and semi-Lagrangian

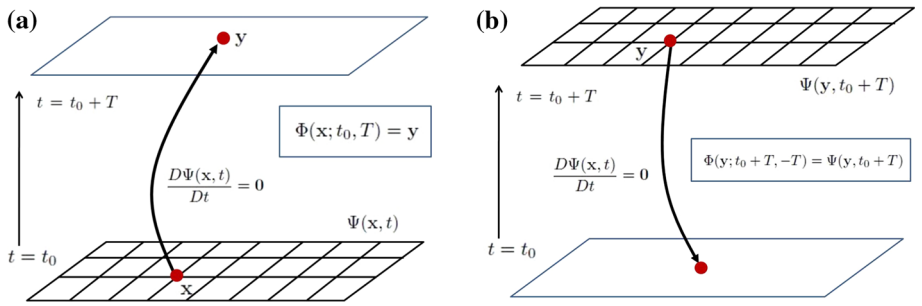

Fig. 1 Lagrangian and Eulerian interpretations of the function Ψ. **a** Lagrangian ray tracing from a given grid location $\mathbf{x}$ at $t = 0$. Note that $\mathbf{y}$ might be a non-grid point. **b** Eulerian values of Ψ at a given grid location $\mathbf{y}$ at $t = T$ gives the corresponding take-off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

approaches. When the velocity $\mathbf{u}$ is given at the grid locations, i.e. $\mathbf{u}(\mathbf{x}_{i,j}, t_n)$, we have to interpolate the flow (possibly using some high order methods like [15]) to obtain values at non-grid locations. This could be rather troublesome in practice.

3 Our Eulerian Approaches

In this section, we discuss the second class of mathematical formulation based on the Eulerian representation of the motion. We first review the Eulerian approach for Problem B developed earlier in [20,21,43]. The new approach for Problem F will be given in the next subsection.

3.1 An Eulerian Approach for the Problem B

Following the idea in [20,21,43], we define a vectored value 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 labelling for any particle in the phase space at $t = 0$. In particular, any particle initially located at $(\mathbf{x}_0, 0) = (x_0^1, x_0^2, \ldots, x_0^d, 0)$ in the extended phase space can be **implicitly** represented by the intersection of d codimension-1 surfaces represented by $\cap_{i=1}^{d} \{\Psi^i(\mathbf{x}, 0) = x_0^i\}$ in $\mathbb{R}^d$. At first glance this representation seems redundant. However, following the particle trajectory with $\mathbf{x} = \mathbf{x}_0$ as the initial condition in a given velocity field, any particle identity should be preserved in the Lagrangian framework and this implies that the material derivative of these level set functions is zero, i.e.

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

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

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

with the initial condition (3). This **implicit** representation therefore embeds all path lines in the extended phase space. For instance, the trajectory of a particle initially located at $(\mathbf{x}_0, 0)$ can be found by determining the intersection of d codimension-1 surfaces represented by

$\cap_{i=1}^{d}\{\Psi^i(\mathbf{x},t)=x_0^i\}$ in the extended phase space. Furthermore, this implicit representation provides a way to determine the *forward* flow map from $t=0$ to $t=T$. In particular, the *forward* flow map at a grid location $\mathbf{x}=\mathbf{x}_0$ is given by $\Phi(\mathbf{x}_0;0,T)=\mathbf{y}$ for $\mathbf{y}$ satisfies $\Psi(\mathbf{y},0+T)=\Psi(\mathbf{x}_0,0)\equiv\mathbf{x}_0$. Note that $\mathbf{y}$ in general is a non-mesh location. We have demonstrated a typical two dimensional scenario in Fig. 1a.

The solution to (4) contains much more information than what we have just interpreted above. Consider a given mesh location $\mathbf{y}$ in the phase space at time $t=T$, as shown in Fig. 1b, i.e. $(\mathbf{y},T)$ in the extended phase space. Since the intersection $\cap_{i=1}^{d}\{\Psi^i(\mathbf{x},t)=\Psi^i(\mathbf{y},T)\}$ represents the path line in the extended phase space passing through $(\mathbf{y},T)$, the level set functions $\Psi(\mathbf{y},T)$ represent the coordinates of the takeoff location at $t=0$ of a Lagrangian particle reaching $\mathbf{y}$ at $t=T$. Therefore, these level set functions defined on a uniform Cartesian mesh in fact give the *backward* flow map from $t=T$ to $t=0$, i.e. $\Phi(\mathbf{y};T,-T)=\Psi(\mathbf{y},T)$. Moreover, solution to the level set equations (4) for $t\in(0,T)$ provides also backward flow maps for all intermediate times, i.e. $\Phi(\mathbf{y};t,-t)=\Psi(\mathbf{y},t)$.

Numerically, we obtain the backward flow map by solving the Liouville equation forward in time. For example, if the forward Euler method is used, we have

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

Higher order extensions are possible using typical WENO-TVDRK approaches. For example, using the TVD-RK2 scheme we have

$$\Phi_{t_{n+1}}^{t_n}(\mathbf{x}_{i,j})=\mathbf{x}_{i,j}-\frac{\Delta t}{2}\left(\mathbf{u}(\mathbf{x}_{i,j},t_n)+\mathbf{u}(\mathbf{x}_{i,j},t_{n+1})\right)+\frac{\Delta t^2}{2}\mathbf{u}(\mathbf{x}_{i,j},t_{n+1})\cdot\nabla\mathbf{u}(\mathbf{x}_{i,j},t_n)\,. \quad (5)$$

Concerning the quantity evolved along the characteristics, we consider the following advection equation given by

$$\frac{\partial\hat{f}}{\partial t}+\mathbf{u}\cdot\nabla\hat{f}=g(\mathbf{x},t) \quad (6)$$

with the velocity field $\mathbf{u}=\mathbf{u}(\mathbf{x},t)$, the initial condition $\hat{f}(\mathbf{x},t_n)=0$ and a time-dependent source term $g(\mathbf{x},t)$. Numerically, we first solve the PDE forward in time from $t=t_n$ to $t=t_{n+1}$ together with the Liouville equation. The solution $\hat{f}(\mathbf{x}_{i,j},t_{n+1})$ represents the line integral of the function g along a trajectory reaching $\mathbf{x}_{i,j}$ at time $t=t_{n+1}$ starting from the time $t=t_n$. In particular, if the temporal direction is discretized using the TVD-RK2 scheme, then we have

$$\hat{f}(\mathbf{x}_{i,j},t_{n+1})=\frac{\Delta t}{2}[g(\mathbf{x}_{i,j},t_{n+1})+g(\mathbf{x}_{i,j},t_n)]-\frac{\Delta t^2}{2}\mathbf{u}(\mathbf{x}_{i,j},t_{n+1})\cdot\nabla g(\mathbf{x}_{i,j},t_n)\,. \quad (7)$$

Once we have the backward flow map, we interpolate and obtain

$$f(\mathbf{x}_{i,j},t_{n+1})=\hat{f}(\mathbf{x}_{i,j},t_{n+1})+f(\Phi_{t_{n+1}}^{t_n}(\mathbf{x}_{i,j}),t_n) \quad (8)$$

which represents the line integral of g along the particle trajectory reaching the location $\mathbf{x}_{i,j}$ at the time $t=t_{n+1}$ from the initial time $t=0$.

Finally, we want to point out that this is in fact an Eulerian formulation of the semi-Lagrangian scheme. Instead of solving the ODE backward in time, this method proposes to solve the corresponding PDE forward in time.

3.2 An Eulerian Approach for the Problem F

This is a significantly less developed area. To the best of our knowledge, we are not aware of any numerical algorithms to solve this problem in an Eulerian framework except those proposed recently in [20,21,43]. For the *forward* flow map, we have proposed to reverse the above process as in the *backward* flow map [20]. In particular, we initialize the level set functions at the final time $t = T$ by $\Psi(\mathbf{x}, T) = \mathbf{x}$, and solve the corresponding level set equations (4) *backward* in time. One disadvantage of such an Eulerian approach for forward flow map computation as discussed in [20,21,43] is that the level set equation (4) has to be first solved *backward* in time from $t = T$ to $t = 0$. Once we have the final solution at the time level $t = 0$, one can then identity the flow map $\Phi_0^T(\mathbf{x})$ by $\Psi(\mathbf{x}, 0)$. This could be inconvenient especially when we need to access the intermediate forward flow map. In this section, we extend the algorithm developed in [45] and propose a new class of algorithms to construct the forward flow map *on the fly*.

Consider the forward flow map from $t = 0$ to $t = T$. Suppose the time domain $[0, T]$ is discretized by $N + 1$ discrete points t_n, where $t_0 = 0$ and $t_N = T$. Then on each subinterval $[t_n, t_{n+1}]$ for $n = 0, 1, \ldots, N-1$, we would like to determine the forward flow map $\Phi_{t_n}^{t_{n+1}}$. Then, the overall flow map can be constructed using $\Phi_0^T = \Phi_{t_{N-1}}^{t_N} \circ \cdots \circ \Phi_0^{t_1}$. The composition of these flow maps from one intermediate stage to the next one involves interpolation. Yet, these interpolations can be easily constructed since we are evaluating flow maps at non-grid locations given a uniform Cartesian sampling. We can simply use the MATLAB function interp2 or interp3 for two- or three-dimensional implementations, respectively.

One way to determine the local flow map from t_n to t_{n+1} is based on the fact that any flow map is reversible, i.e.

$$\Phi_{t_n}^{t_{n+1}} = \left(\Phi_{t_{n+1}}^{t_n}\right)^{-1}.$$

We might first determine the *backward* flow map $\Phi_{t_{n+1}}^{t_n}$ using the technique developed in Sect. 3.1 since essentially we are solving a step of the Problem B. The numerical implementation of this step is straight-forward in the Eulerian framework. However, computing the numerical inverse of the flow map in general requires solving the following root-finding problem,

> "Given $\mathbf{x}_{i,j} \to \Phi_{t_{n+1}}^{t_n}(\mathbf{x}_{i,j})$ with $\mathbf{x}_{i,j}$ sampled on a uniform mesh, determine a location $\mathbf{y}_{i,j}$ such that $\mathbf{y}_{i,j} \to \mathbf{x}_{i,j}$ for each i and j."

Indeed, this root-finding problem is equivalent to the interpolation problem given by,

> "Given samplings $(\mathbf{y}_{i,j}, \mathbf{x}_{i,j})$ at some *scattered* locations $\mathbf{y}_{i,j}$, interpolate at uniform grid points $\mathbf{x}_{i,j}$."

which can be solved by first creating a Delaunay triangulation from the scattered sampling locations and then lifting the vertices of the triangles using $\mathbf{x}_{i,j}$. Efficient yet accurate implementation of this approach, on the other hand, is not as straight-forward.

Instead, we follow the same approach as described in [20]. But instead of constructing the *global* flow map by solving the PDE all the way from $t = T$ backward to $t = 0$, we construct only the local forward flow map $\Phi_{t_n}^{t_{n+1}}$. In particular, we solve the level set equation (4) *backward* in time from $t = t_{n+1}$ to $t = t_n$ with the terminal condition $\Psi(\mathbf{x}, t_{n+1}) = \mathbf{x}$ imposed on the time level $t = t_{n+1}$. Then the forward flow map is given by $\Phi_{t_n}^{t_{n+1}}(\mathbf{x}) = \Psi(\mathbf{x}, t_n)$. Once we have obtained this one step *forward* flow map $\Phi_{t_n}^{t_{n+1}}$, the *forward* flow map from $t = 0$ to $t = T$ can be obtained using the composition $\Phi_0^{t_{n+1}} = \Phi_{t_n}^{t_{n+1}} \circ \Phi_0^{t_n}$. And this can be

easily done by typical numerical interpolation methods. To prevent extrapolation, we could enforce $\Phi_{t_n}^{t_{n+1}} \subseteq [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. Here we emphasize again that even though we solve equation (4) *backward* in time for each subinterval, we access the velocity field data $\mathbf{u}^n$ prior to $\mathbf{u}^{n+1}$ and therefore the forward flow map is indeed obtained *on the fly*.

3.2.1 The First Order Euler method

As a simple example, we now consider the first order Euler method in the temporal direction. The forward flow map $\Phi_{t_n}^{t_{n+1}}$ can be obtained by first solving

$$\frac{\Psi_{i,j}^n - \Psi_{i,j}^{n+1}}{\Delta t} - \mathbf{u}_{i,j}^{n+1} \cdot \nabla \Psi_{i,j}^{n+1} = \mathbf{0}$$

with the terminal condition $\Psi_{i,j}^{n+1} = \mathbf{x}_{i,j}$, and then assigning $\Phi_{t_n}^{t_{n+1}} = \Psi_{i,j}^n$. The solution to this equation can be explicitly constructed and is given by

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

Even though looks similar, this numerical scheme is different from the Lagrangian approach to the Problem F where the forward Euler solution is given by

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

while the backward Euler solution is given implicitly by

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

3.2.2 TVD-RK

Higher order generalization is straight-forward. For example, if we use the TVD-RK2 method in the temporal direction [34], we can compute the forward flow map $\Phi_{t_n}^{t_{n+1}}$ by

$$\begin{aligned}
\frac{\hat{\Psi}_{i,j}^n - \Psi_{i,j}^{n+1}}{\Delta t} - \mathbf{u}_{i,j}^{n+1} \cdot \nabla \Psi_{i,j}^{n+1} &= \mathbf{0} \\
\frac{\hat{\Psi}_{i,j}^{n-1} - \hat{\Psi}_{i,j}^n}{\Delta t} - \mathbf{u}_{i,j}^n \cdot \nabla \hat{\Psi}_{i,j}^n &= \mathbf{0} \\
\Psi_{i,j}^n = \frac{1}{2}\left(\hat{\Psi}_{i,j}^{n-1} + \Psi_{i,j}^{n+1}\right)&
\end{aligned}$$

with the terminal condition $\Psi_{i,j}^{n+1} = \mathbf{x}_{i,j}$. Functions $\hat{\Psi}_{i,j}^{n-1}$ and $\hat{\Psi}_{i,j}^n$ are two intermediate predicted solutions at the time levels t_{n-1} and t_n, respectively. Then the flow map from the time level $t = t_n$ to $t = t_{n+1}$ is given by $\Phi_{t_n}^{t_{n+1}}(\mathbf{x}) = \Psi^n(\mathbf{x})$. More explicitly, we have

$$\Phi_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j}) = \mathbf{x}_{i,j} + \frac{\Delta t}{2}\left(\mathbf{u}(\mathbf{x}_{i,j}, t_n) + \mathbf{u}(\mathbf{x}_{i,j}, t_{n+1})\right) + \frac{\Delta t^2}{2}\left[\mathbf{u}(\mathbf{x}_{i,j}, t_n) \cdot \nabla\right]\mathbf{u}(\mathbf{x}_{i,j}, t_{n+1})\,. \tag{9}$$

A nice feature about this numerical approach is that the gradient of $\mathbf{u}$ is computed at grid points which can be easily computed using any well-developed approach such as ENO or WENO [26,39]. Unlike typical Lagrangian approach where the velocity field has to be interpolated at non-grid locations using high order reconstructions, the Eulerian approach to the local Problem F involves only approximating the derivatives of the velocity field at uniform grid

locations from sampling at the same set of mesh locations. This is a rather standard numerical problem nowadays.

Here we also give the expression for the TVD-RK3 for completeness,

$$\begin{aligned}
\hat{\Psi}^{n}_{i,j} &= \mathbf{x}_{i,j} + \Delta t\, \mathbf{u}^{n+1}_{i,j} \\
\hat{\Psi}^{n-1}_{i,j} &= \hat{\Psi}^{n}_{i,j} + \Delta t \left(\mathbf{u}^{n}_{i,j} \cdot \nabla\right) \hat{\Psi}^{n}_{i,j} \\
\hat{\Psi}^{n+\frac{1}{2}}_{i,j} &= \frac{1}{4}\left(\hat{\Psi}^{n-1}_{i,j} + 3\mathbf{x}_{i,j}\right) \\
\hat{\Psi}^{n-\frac{1}{2}}_{i,j} &= \hat{\Psi}^{n+\frac{1}{2}}_{i,j} + \Delta t \left(\mathbf{u}^{n+\frac{1}{2}}_{i,j} \cdot \nabla\right) \hat{\Psi}^{n+\frac{1}{2}}_{i,j} \\
\Psi^{n}_{i,j} &= \frac{1}{3}\left(2\hat{\Psi}^{n-\frac{1}{2}}_{i,j} + \mathbf{x}_{i,j}\right).
\end{aligned}$$

This implies

$$\begin{aligned}
K &= \mathbf{u}(\mathbf{x}_{i,j}, t_n) + \mathbf{u}(\mathbf{x}_{i,j}, t_{n+1}) + \Delta t \left[\mathbf{u}(\mathbf{x}_{i,j}, t_n) \cdot \nabla\right] \mathbf{u}(\mathbf{x}_{i,j}, t_{n+1}) \\
\Phi^{t_{n+1}}_{t_n}(\mathbf{x}_{i,j}) &= \mathbf{x}_{i,j} + \frac{\Delta t}{6}\left(K + 4\mathbf{u}\left(\mathbf{x}_{i,j}, t_{n+\frac{1}{2}}\right)\right) + \frac{\Delta t^2}{6}\left[\mathbf{u}\left(\mathbf{x}_{i,j}, t_{n+\frac{1}{2}}\right) \cdot \nabla\right] K.
\end{aligned}$$

Even though we have to provide $\mathbf{u}^{n+\frac{1}{2}}_{i,j}$ in this third order scheme, it can be approximated by a linear interpolation in the temporal direction given by

$$\mathbf{u}^{n+\frac{1}{2}}_{i,j} \approx \frac{1}{2}\left(\mathbf{u}^{n}_{i,j} + \mathbf{u}^{n+1}_{i,j}\right). \tag{10}$$

Finally, we have

$$\begin{aligned}
\Phi^{t_{n+1}}_{t_n}(\mathbf{x}_{i,j}) = \mathbf{x}_{i,j} &+ \frac{\Delta t}{2}\left(\mathbf{u}(\mathbf{x}_{i,j}, t_n) + \mathbf{u}(\mathbf{x}_{i,j}, t_{n+1})\right) \\
&+ \frac{\Delta t^2}{6}\left\{\left[\mathbf{u}(\mathbf{x}_{i,j}, t_n) \cdot \nabla\right] \mathbf{u}(\mathbf{x}_{i,j}, t_{n+1}) + \left[\frac{\mathbf{u}(\mathbf{x}_{i,j}, t_n) + \mathbf{u}(\mathbf{x}_{i,j}, t_{n+1})}{2} \cdot \nabla\right] K\right\}.
\end{aligned}$$

Considering the computation of the line integral along each characteristic, we follow a similar approach we introduced in the previous section, we consider an advection equation (6) with the terminal condition $\hat{f}(\mathbf{x}, t_{n+1}) = 0$. Numerically, we first solve the PDE backward in time from $t = t_{n+1}$ to $t = t_n$ together with the Liouville equation. The corresponding solution $\hat{f}(\mathbf{x}_{i,j}, t_n)$ represents the negative of the line integral of the function g along the characteristic starting at $\mathbf{x}_{i,j}$ at the time $t = t_n$ shooting forward in time to $t = t_{n+1}$. If we use the TVD-RK2 scheme to discretize (6) in the temporal direction, then $\hat{f}(\mathbf{x}_{i,j}, t_n)$ is approximated as

$$\hat{f}(\mathbf{x}_{i,j}, t_n) = -\frac{\Delta t}{2}[g(\mathbf{x}_{i,j}, t_{n+1}) + g(\mathbf{x}_{i,j}, t_n)] - \frac{\Delta t^2}{2}\mathbf{u}(\mathbf{x}_{i,j}, t_n) \cdot \nabla g(\mathbf{x}_{i,j}, t_{n+1}). \tag{11}$$

This implies

$$f(\mathbf{x}_{i,j}, t_{n+1}) = f(\mathbf{x}_{i,j}, t_n) - \hat{f}(\Phi^{t_n}_0(\mathbf{x}_{i,j}), t_n) \tag{12}$$

and it represents the line integral of g along the characteristics starting from $\mathbf{x}_{i,j}$ at time $t = 0$ to the final time $t = t_{n+1}$.

3.3 Comparisons of the Eulerian Interpolation Schemes and the Lagrangian Schemes

The proposed Eulerian methods have some similar properties like typical Lagrangian or the semi-Lagrangian methods. The proposed methods solve both Problem B and Problem F forward in time in a way that the dynamic is loaded forward in time. Unlike the Eulerian algorithms proposed in [20,21,43] and references thereafter, this gives an implementation-wise natural numerical algorithm in practice.

Interpolation plays an important role in many of these approaches. The semi-Lagrangian method for the Problem B requires interpolating each individual flow map to construct the flow map from the final time step back to the initial condition. In the semi-Lagrangian methods and the Lagrangian methods when the velocity field is available only at the gridded locations (e.g. from a computational fluid dynamic solver), extra interpolations are necessary when the numerical integrator requires velocity at arbitrary locations and, worst, at a non-mesh time. Such interpolation is done in the $(d+1)$-dimensions when $\mathbf{x} \in \mathbb{R}^d$.

Indeed, our proposed Eulerian approach for the Problem F requires interpolation at every step too. The first interpolation might be necessary in the construction of the local flow map when high order methods like TVD-RK3 is needed. In the above discussion, we applied the linear interpolation in the temporal direction given by (10). Of course, higher order interpolation might be necessary in practice. However, such an interpolation step is done only in the temporal direction at each individual mesh point. This interpolation is only a simple one-dimensional problem. The second interpolation in the Eulerian algorithm is in the composition of these local flow maps, which is simply a d-dimensional interpolation problem. Therefore, the proposed Eulerian method separates the temporal and the spatial interpolations.

3.4 Computational Complexities

In this section, we discuss the computational complexity of our proposed algorithms. We first consider Problem F. Let M and N be the discretization size of one spatial dimension and time dimension respectively. At each time step t_n, a short time flow map $\Phi_{t_{n-1}}^{t_n}$ is obtained by solving the Liouville equation (4) from t_n to t_{n-1}, the computational effort is $O(M^2)$. To construct the long time flow map $\Phi_0^{t_n}$, we require an interpolation $\Phi_0^{t_n} = \Phi_{t_{n-1}}^{t_n} \circ \Phi_0^{t_{n-1}}$ which takes $O(M^2)$ operations. Therefore the construction of flow map at each time step requires $O(M^2) + O(M^2) = O(M^2)$ operations. Computing $\hat{f}(\mathbf{x}_{i,j}, t_n)$ by solving Eq. (6) from t_{n+1} to t_n also needs $O(M^2)$ operations. Finally interpolating to obtain $\hat{f}(\Phi_0^{t_n}(\mathbf{x}_{i,j}), t_n)$ and hence the evaluation of $f(\mathbf{x}_{i,j}, t_{n+1})$ takes further $O(M^2)$ operations. Summing up this procedure in all time steps, the computational complexity is $N \cdot O(M^2)$. Finally, since the Eqs. (4) and (6) are both hyperbolic type, we have $\Delta t = O(\Delta x)$ due to the CFL stability condition. This implies $N = O(1/\Delta t) = O(1/\Delta x) = O(M)$. And, therefore, the overall computational complexity for solving Problem F is given by $O(M^3)$. Similarly, the overall computational complexity for solving Problem B is also $O(M^3)$.

4 Analysis of the Interpolation Schemes

In this section, we give some analytical results including a monotonicity property and some error estimates of our proposed Eulerian approaches.

4.1 Monotonicity

In the previous paper [20,21,43], we impose the following boundary conditions to the Liouville equation for computing the *backward* flow map by solving the time-dependent PDE *forward* in time. For $t > 0$,

$$\Psi(\mathbf{x}, t)|_{\mathbf{x}\in\partial\Omega} = \Psi(\mathbf{x}, 0)|_{\mathbf{x}\in\partial\Omega} = \mathbf{x} \tag{13}$$

if $\mathbf{n}\cdot\mathbf{u} < 0$ where $\mathbf{n}$ is the outward normal of the boundary, and $\mathbf{n}\cdot\nabla\Psi^i(\mathbf{x}, t)|_{\mathbf{x}\in\partial\Omega} = 0$ for $i = 1, \ldots, d$ if $\mathbf{n}\cdot\mathbf{u} > 0$, i.e. we fix the inflow boundary condition on $\partial\Omega_i = \{\mathbf{x} \in \partial\Omega : \mathbf{n}\cdot\mathbf{u}(\mathbf{x}) < 0\}$ where $\mathbf{n}$ is the outward normal of the domain Ω.

We follow the same procedure in this paper. Whenever necessary, we impose the boundary condition (13) on the inflow boundary when solving the Liouville equation. With this treatment, we have the following property.

Lemma 1 *([20]) With the boundary condition (13) on the inflow boundary, the exact flow map Ψ_0^t (and Ψ_t^0) satisfies $\Psi_0^t(x, y) \in \Omega$ (and $\Psi_t^0(x, y) \in \Omega$) for all $(x, y) \in \Omega$ and $t \in [0, T]$ with $\Omega = [x_{\min}, x_{\max}]$.*

To obtain a stable evolution in the flow map constructions, we require the interpolation scheme to be *monotone*, i.e. the interpolation scheme should preserve the monotonicity of the given data points. With this monotonicity constraint imposed on the interpolation scheme, we have the following property guaranteeing the availability of data for interpolation and avoiding extrapolation in the numerical algorithm. This would lead to the robustness in the proposed interpolation method. The proof of this property is similar to the idea as shown in [21] and therefore is omitted here.

Lemma 2 *We consider the proposed Eulerian approach for solving Problem F. If*

1. *both flow maps $\Phi_{t_{n-1}}^{t_n} \in \Omega$ and $\Phi_0^{t_{n-1}}(x_i, y_j) \in \Omega$ for $\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, and*
2. *the interpolation scheme is monotone,*

then we have $\Phi_0^{t_n}(x_i, y_j) \in \Omega$ for all $i = 1, 2, \ldots, I,\ j = 1, 2, \ldots, J$.

4.2 Accuracy

In this section, we consider the overall accuracy in the numerical solution obtained based on the Eulerian interpolation schemes.

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 $\mathbf{x} \in M$. 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 $t > 0$.

Proof Differentiating the expression

$$\frac{d\Phi_0^t(\mathbf{x})}{dt} = \mathbf{u}(\Phi_0^t(\mathbf{x}), t) \tag{14}$$

with respect to the i-th spatial variable x_i, we have

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

Because the norm of $\nabla\mathbf{u}$ is the Lipschitz constant L, we have

$$\frac{d}{dt}\left|\frac{\partial[\Phi_0^t(\mathbf{x})]}{\partial x_i}\right| \le \left|\frac{d}{dt}\frac{\partial[\Phi_0^t(\mathbf{x})]}{\partial x_i}\right| \le L\cdot\left|\frac{\partial[\Phi_0^t(\mathbf{x})]}{\partial x_i}\right|,$$

with

$$\left|\frac{\partial[\Phi_0^t(\mathbf{x})]}{\partial x_i}\right|_{t=0} = 1.$$

Let $r(t) \triangleq \left|\frac{\partial[\Phi_0^t(\mathbf{x})]}{\partial x_i}\right|$, then we have

$$\frac{d[r(t)e^{-Lt}]}{dt} \le 0$$

which implies that $r(t)e^{-Lt}$ is decreasing and thus $r(0) \ge r(t)e^{-Lt}$, i.e.

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

□

The following two lemmas extend those developed in [42] from autonomous flows to time-dependent velocity fields as discussed in the current work.

Lemma 4 *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*

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

Proof It can be easily obtained from Lemma 3 by using the Differential Mean Value Theorem of binary functions. □

Lemma 5 *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* $\mathbf{x}\in M$, *we have*

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

Proof Since $\mathbf{u}$ is smooth enough, there exists a constant C_s^1 such that for any multi-index γ with $|\gamma|\le s$, we have

$$|\partial^\gamma\mathbf{u}^i(\mathbf{x},t)| \le C_s^1,\quad \forall\mathbf{x}\in M,\quad \forall t>0,$$

where $\mathbf{u}^i$ with $i=1,2,\ldots,d$ are the d components of $\mathbf{u}$ and the partial derivatives are taken with respect to $\mathbf{x}$.

Now, we first show by induction that there exists a constant C_s^2 such that for any γ with $|\gamma| = s$, we have

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

Lemma 3 shows the case for $s = 1$. Now we assume that the statement is true for arbitrary multi-index of order less than s. Fix γ with $|\gamma| = s > 1$ and differentiate (14) on both sides to obtain

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

with $\partial^\gamma \phi^i|_{t=0} = 0$, ϕ^i denotes the i-th component of $\Phi_0^t(\mathbf{x})$, and $u^i_{k_1\ldots k_p} \triangleq \frac{\partial^p u^i}{\partial x_{k_1}\ldots\partial x_{k_p}}$ denotes the corresponding partial derivatives of the i-th component $u^i(\mathbf{x}, t)$ of the velocity field $\mathbf{u}(\mathbf{x}, t)$. The second summation is over all different choices of $(\gamma_1, \ldots, \gamma_p)$ such that $\gamma = \gamma_1 + \cdots + \gamma_p$ and $|\gamma_l| > 0$. In the case when $p = 1$, there is only a single term in the expansion, namely, $\sum_{k=1}^{d} u^i_k \partial^\gamma \phi^k$.

Now, since

$$\prod_{j=1}^{p} \partial^{\gamma_j}\phi^{k_j} \le Ce^{(2s-p)Lt}$$

for any $p > 1$ where C is a constant which depends on C_p^2 for $p < s$, it follows that

$$\frac{d|\partial^\gamma \Phi_0^t(\mathbf{x})|}{dt} \le \left|\frac{d[\partial^\gamma \Phi_0^t(\mathbf{x})]}{dt}\right| \le L|\partial^\gamma \Phi_0^t(\mathbf{x})| + C'e^{(2s-2)Lt}, \tag{15}$$

where C' is a constant depending on C_p^1 for $p \le s$ and C_p^2 for $p < s$.

Let $r(t) \triangleq |\partial^\gamma \Phi_0^t(\mathbf{x})|$, then we have

$$\frac{dr}{dt} \le Lr + C'e^{(2s-2)Lt}$$

where $r(0) = |\partial^\gamma \Phi_0^0(\mathbf{x})| = 0$. Let $q(t) = \frac{C'}{(2s-3)L}[e^{(2s-2)Lt} - e^{Lt}]$. We have

$$\frac{dq}{dt} = Lq + C'e^{(2s-2)Lt}$$

and thus $\frac{dh}{dt} \le Lh$ where $h(t) \triangleq r(t) - q(t)$. Equivalently, we have $\frac{d(he^{-Lt})}{dt} \le 0$ and thus $h(t)e^{-Lt} \le h(0) = r(0) - q(0) = 0$, i.e. $h(t) = r(t) - q(t) \le 0$. That is,

$$|\partial^\gamma \Phi_0^t(\mathbf{x})| \le \frac{C'}{(2s-3)L}\left[e^{(2s-2)Lt} - e^{Lt}\right] \le C_s^2 e^{(2s-1)Lt}$$

where C_s^2 depends only on C', s and L.

To prove the lemma, observe that the right hand side of (15) is bounded by $C_s e^{(2s-1)Lt}$ where the constant C_s depends on C_s^1 and C_s^2. Finally, integrating the expression from time 0 to t gives

$$\begin{aligned}|\partial^\gamma \Phi_0^t(\mathbf{x})| &\le C_s \int_0^t e^{(2s-1)L\tau} d\tau \le C_s e^{(2s-2)Lt} \int_0^t e^{L\tau} d\tau \\ &= C_s \cdot e^{(2s-2)Lt} \cdot \frac{e^{Lt} - 1}{L} \le C_s\, t\, e^{(2s-1)Lt}.\end{aligned}$$

□

With the above lemmas, we are now able to state and prove the following results on the accuracy of the proposed Eulerian algorithms.

Theorem 1 *Assuming that the interpolation is at least second order accurate and* $\Phi_{t_n}^{t_{n+1}}$ *is computed using* (9)*, the forward flow map* Φ_0^T *computed using the method proposed in* Sect. 3.2 *has second order accuracy in both temporal and spatial directions.*

Proof We use $\tilde{\cdot}$ to denote the numerical solution and $\mathcal{I}$ to denote the interpolation operator. Then formula (9) can be rewritten as

$$\tilde{\Phi}_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j}) = \mathbf{x}_{i,j} + \frac{\Delta t}{2}\left(\mathbf{u}(\mathbf{x}_{i,j}, t_n) + \mathbf{u}(\mathbf{x}_{i,j}, t_{n+1})\right) + \frac{\Delta t^2}{2}\mathbf{u}(\mathbf{x}_{i,j}, t_n)\cdot\nabla\mathbf{u}(\mathbf{x}_{i,j}, t_{n+1}),$$

which has an error

$$|\tilde{\Phi}_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j}) - \Phi_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j})| \le C_1\Delta t^3 + C_2\Delta t^2\Delta x^\alpha .$$

The term $C_1\Delta t^3$ comes from the temporal discretization to the PDE, α is the order of the discretization to $\nabla\mathbf{u}$. The constants C_1 and C_2 depend only on the $\mathbf{u}$. Then $\Phi_0^{t_{n+1}}$ is approximated by

$$\tilde{\Phi}_0^{t_{n+1}}(\mathbf{x}_{i,j}) = \mathcal{I}\tilde{\Phi}_{t_n}^{t_{n+1}} \circ \tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j})$$

with the error

$$\begin{aligned}
|\tilde{\Phi}_0^{t_{n+1}}(\mathbf{x}_{i,j}) - \Phi_0^{t_{n+1}}(\mathbf{x}_{i,j})| \le\ & |\mathcal{I}\tilde{\Phi}_{t_n}^{t_{n+1}}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j})) - \mathcal{I}\Phi_{t_n}^{t_{n+1}}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}))| \\
&+ |\mathcal{I}\Phi_{t_n}^{t_{n+1}}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j})) - \Phi_{t_n}^{t_{n+1}}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}))| \\
&+ |\Phi_{t_n}^{t_{n+1}}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j})) - \Phi_{t_n}^{t_{n+1}}(\Phi_0^{t_n}(\mathbf{x}_{i,j}))| \\
\triangleq\ & I_1 + I_2 + I_3.
\end{aligned}$$

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

$$I_1 \le N_I \cdot \max_{\mathbf{x}_{i,j}} |\tilde{\Phi}_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j}) - \Phi_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j})| \le C_1 N_I \Delta t^3 + C_2 N_I \Delta t^2 \Delta x^\alpha.$$

Suppose the interpolation operator is of order β, then I_2 is bounded by

$$I_2 \le C_3' \Delta x^\beta \max_{|\gamma|=\beta} \sup_{\mathbf{x}_{i,j}} |\partial^\gamma \Phi_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j})|.$$

According to Lemma 5, there exists a constant C_β such that

$$\max_{|\gamma|=\beta} \sup_{\mathbf{x}_{i,j}} |\partial^\gamma \Phi_{t_n}^{t_{n+1}}(\mathbf{x}_{i,j})| \le C_\beta \cdot e^{(2\beta-1)L\Delta t} \cdot \Delta t$$

which implies $I_2 \le C_3 e^{(2\beta-1)L\Delta t} \cdot \Delta t \Delta x^\beta$ where C_3 is a constant. Finally, by Lemma 4, I_3 is bounded by $I_3 \le e^{L\Delta t}|\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}) - \Phi_0^{t_n}(\mathbf{x}_{i,j})|$. Let $E^k \triangleq \max_{\mathbf{x}_{i,j}} |\tilde{\Phi}_0^{t_k}(\mathbf{x}_{i,j}) - \Phi_0^{t_k}(\mathbf{x}_{i,j})|$ and note the CFL condition $\Delta t = O(\Delta x)$. We have $E^{n+1} \le e^{L\Delta t}E^n + C_4\Delta t^3$ where C_4 is a constant which depends on C_1, C_2, C_3, N_I, L and β. By recurrence, we have

$$E^{n+1} \le e^{nL\Delta t}E^1 + C_4 \cdot n\Delta t^3 e^{nL\Delta t} \le e^{Lt}E^1 + C_4 \cdot t \cdot \Delta t^2 e^{Lt}.$$

On the other hand, E^1 is bounded by $E^1 \le C_1\Delta t^3 + C_2\Delta t^2\Delta x^\alpha$. As a result, we finally obtain $E^{n+1} \le C_5\Delta t^2$ where C_5 depends on C_4 and the final time T. □

Theorem 2 *Suppose that the interpolation scheme is of order $\beta \geq 2$ and $g(\mathbf{x}, t)$ is a sufficiently smooth function, and $\hat{f}$ is approximated using (11). Then the line integral of the function $g(\mathbf{x}, t)$ computed using (12) is second order accurate with respect to both Δx and Δt.*

Proof We first rewrite (12) in the following form

$$\tilde{f}(\mathbf{x}_{i,j}, t_{n+1}) = \tilde{f}(\mathbf{x}_{i,j}, t_n) - \mathcal{I}\hat{f}_a(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}), t_n)\,.$$

where $\hat{f}_a$ and $\tilde{\Phi}_0^{t_n}$ denote the numerical solution of $\hat{f}$ and $\Phi_0^{t_n}$, respectively. According to (11), $\hat{f}(\mathbf{x}_{i,j}, t_n)$ is approximated as

$$\hat{f}_a(\mathbf{x}_{i,j}, t_n) = -\frac{\Delta t}{2}[g(\mathbf{x}_{i,j}, t_{n+1}) + g(\mathbf{x}_{i,j}, t_n)] - \frac{\Delta t^2}{2}\mathbf{u}(\mathbf{x}_{i,j}, t_n) \cdot \nabla g(\mathbf{x}_{i,j}, t_{n+1})$$

with the error $|\hat{f}_a(\mathbf{x}_{i,j}, t_n) - \hat{f}(\mathbf{x}_{i,j}, t_n)| \leq C_1 \Delta t^3$ for any $n = 0, 1, \ldots, N-1$. Let $E^k(\mathbf{x}_{i,j}) \triangleq |\tilde{f}(\mathbf{x}_{i,j}, t_k) - f(\mathbf{x}_{i,j}, t_k)|$, then

$$\begin{aligned}
E^{n+1}(\mathbf{x}_{i,j}) &\leq E^n(\mathbf{x}_{i,j}) + |\mathcal{I}\hat{f}_a(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}), t_n) - \hat{f}(\Phi_0^{t_n}(\mathbf{x}_{i,j}), t_n)| \\
&\leq E^n(\mathbf{x}_{i,j}) + |\mathcal{I}\hat{f}_a(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}), t_n) - \mathcal{I}\hat{f}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}), t_n)| \\
&\quad + |\mathcal{I}\hat{f}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}), t_n) - \hat{f}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}), t_n)| \\
&\quad + |\hat{f}(\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}), t_n) - \hat{f}(\Phi_0^{t_n}(\mathbf{x}_{i,j}), t_n)| \\
&\triangleq E^n(\mathbf{x}_{i,j}) + I_1 + I_2 + I_3\,.
\end{aligned}$$

The term I_1 is bounded by $I_1 \leq N_I \cdot \max_{\mathbf{x}_{i,j}} |\hat{f}_a(\mathbf{x}_{i,j}, t_n) - \hat{f}(\mathbf{x}_{i,j}, t_n)| \leq C_1 N_I \Delta t^3$. Suppose that the interpolation operator has the β-th order accuracy. Then we have

$$I_2 \leq C_2' \cdot \Delta x^\beta \max_{|\gamma|=\beta} \sup_{\mathbf{x}_{i,j}} |\partial^\gamma \hat{f}(\mathbf{x}_{i,j}, t_n)|.$$

We differentiate $\hat{f}(\mathbf{x}, t_n) = -\int_{t_n}^{t_{n+1}} g(\Phi_{t_n}^\tau(\mathbf{x}), \tau) d\tau$ to obtain $\partial^\gamma \hat{f} = -\int_{t_n}^{t_{n+1}} \partial^\gamma g(\Phi_{t_n}^\tau(\mathbf{x}), \tau) d\tau$ where

$$\partial^\gamma g(\Phi_{t_n}^\tau(\mathbf{x}), \tau) = \sum_{p=1}^{\beta} \sum_{(\gamma_1, \ldots, \gamma_p): \gamma = \sum_{l=1}^{p} \gamma_l} \left(\sum_{k_1, \ldots, k_p = 1}^{d} g_{k_1 \ldots k_p} \prod_{j=1}^{p} \partial^{\gamma_j} \phi^{k_j} \right)$$

and ϕ^i denotes the i-th component of $\Phi_{t_n}^\tau(\mathbf{x})$ and $g_{k_1 \ldots k_p} \triangleq \frac{\partial^p g}{\partial x_{k_1} \ldots \partial x_{k_p}}$. From Lemmas 3 and 5, we obtain

$$\left| \prod_{j=1}^{p} \partial^{\gamma_j} \phi^{k_j} \right| \leq C_2$$

for any $p \geq 1$ where C_2 is a constant depends only on L. Since these terms are all finite, it then follows that $|\partial^\gamma g(\Phi_{t_n}^\tau(\mathbf{x}), \tau)| \leq C_3'$ where C_3' depends on C_2, β, d and $\max_{|\gamma| \leq \beta} |\partial^\gamma g|_{L_\infty}$. As a result, $|\partial^\gamma \hat{f}| \leq C_3' \cdot \Delta t$ which implies that $I_2 \leq C_3 \cdot \Delta x^\beta \cdot \Delta t$ where $C_3 = C_2' C_3'$.

On the other hand,

$$I_3 \leq \sup_{\mathbf{x} \in M} |\nabla \hat{f}(\mathbf{x}, t_n)| |\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}) - \Phi_0^{t_n}(\mathbf{x}_{i,j})|\,.$$

Since we have shown that $|\partial^\gamma \hat{f}| \le C_3' \cdot \Delta t$ for all $|\gamma| \le \beta$, then $|\nabla \hat{f}(\mathbf{x}, t_n)| \le C_4 \Delta t$ for all $\mathbf{x} \in M$. From Theorem 1, we can have $|\tilde{\Phi}_0^{t_n}(\mathbf{x}_{i,j}) - \Phi_0^{t_n}(\mathbf{x}_{i,j})| \le C_5 \Delta t^2$ for some constant C_5. As a result, $I_3 \le C_4 \cdot C_5 \cdot \Delta t^3$.

Due to the CFL condition, we have $\Delta t = O(\Delta x)$. Denote $E^k = \max_{\mathbf{x}_{i,j}} E^k(\mathbf{x}_{i,j})$, then

$$E^{n+1} \le E^n + C_1 N_I \Delta t^3 + C_3 \cdot \Delta x^\beta \cdot \Delta t + C_4 C_5 \Delta t^3 \le E^n + C \Delta t^3$$

where C depends on C_1, C_3, C_4, C_5 and N_I. Finally, by recursion we have $E^{n+1} \le E^0 + C \cdot t \cdot \Delta t^2 \le C \cdot T \cdot \Delta t^2$. □

Theorem 3 *Suppose that the interpolation scheme is of order $\beta \ge 2$. If $\Phi_{t_{n+1}}^{t_n}$ and $\hat{f}$ are approximated by solving their corresponding PDEs using the TVD-RK2 in the temporal direction, i.e. (5) and (7) are used to approximate $\Phi_{t_{n+1}}^{t_n}$ and $\hat{f}$ respectively, then the line integral of a sufficiently smooth function $g(\mathbf{x}, t)$ computed using (8) is second order accurate with respect to both Δx and Δt.*

Proof The proof is similar to the one corresponding to Theorem 2 and is omitted here. □

5 Numerical Examples

5.1 The Double Gyre Flow

This example is taken from [38] 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

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

In this example, we follow [38] and use $A = 0.1$, $\omega = 2\pi/10$.

We first use (8) to compute the line integral of $g(x, y) = \cos(12\pi x)\cos(2\pi y)$ along the particle trajectory reaching the location $\mathbf{x}_{i,j}$ at the time $t = 10$ from the initial time $t = 0$ as shown in Fig. 2a where $\Delta x = \Delta y = 1/256$. In the implementation, both $\hat{f}$ and $\Phi_{t_{n+1}}^{t_n}$ are approximated by solving corresponding PDEs using the TVD-RK2 in the temporal direction, i.e. $\hat{f}$ and $\Phi_{t_{n+1}}^{t_n}$ are computed using (7) and (5) respectively. As a result, the solution of f must have second order accuracy with respect to Δx according to Theorem 3. In Fig. 2b we plot the L_2 errors of f using different Δx's ranging from $1/32$ to $1/512$. Since we do not have the exact solution for this flow, the exact solution is computed using the Lagrangian ray tracing with a very small time step. As can be seen, the solution of f is really approximately second order accurate satisfying Theorem 3. We have to emphasize that only velocity data on the mesh is used in the whole implementation, that is, no interpolation on $\mathbf{u}(\mathbf{x}, t)$ is needed which is impossible in both the Lagrangian and Semi-Lagrangian implementations.

Then we use the new approach proposed in Sect. 3.2 to compute the forward flow map $\Phi_0^{10}(\mathbf{x})$. Figure 3a, b show the solutions of the two components ϕ and ψ respectively where $\Delta x = \Delta y = 1/256$. We also use the new approach proposed in Sect. 3.2 to compute the line integral of $g(x, y) = \cos(12\pi x)\cos(2\pi y)$ along the particle trajectory starting from $\mathbf{x}_{i,j}$ at time $t = 0$ to the final time $t = 10$. The solution is shown in Fig. 4a where $\Delta x = \Delta y = 1/256$. In the implementation, both $\hat{f}$ and $\Phi_{t_n}^{t_{n+1}}$ are approximated by solving corresponding PDEs

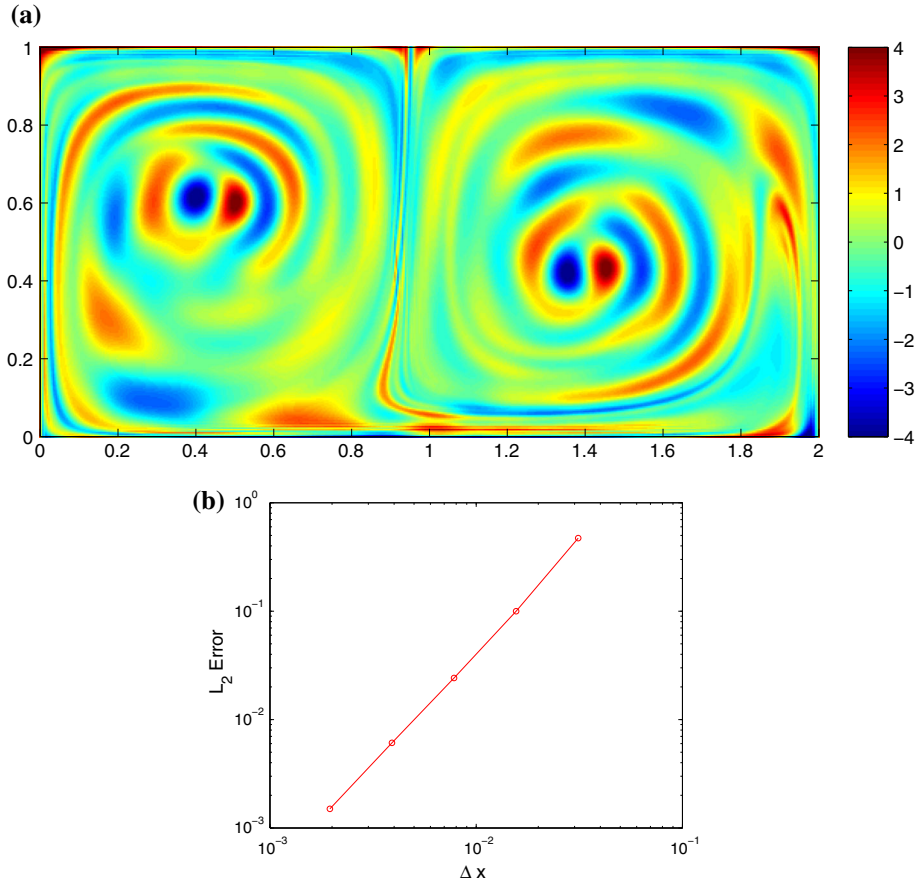

Fig. 2 (Section 5.1) Eulerian approach to Problem B. **a** The line integral of $g(x, y) = \cos(12\pi x)\cos(2\pi y)$ along the particle trajectory reaching the location $\mathbf{x}_{i,j}$ at $t = 10$ from $t = 0$. **b** L_2 error of the solution f with different Δx's using TVD-RK2

using the TVD-RK2 in the temporal direction. As a comparison, Fig. 4b, c show the the line integral of $g(x, y) = \cos(12\pi x)\cos(2\pi y)$ along the particle trajectory starting from $\mathbf{x}_{i,j}$ at $t = 0$ to $t = 10$ computed using the Lagrangian approach with TVD-RK2. The velocity field is interpolated with the bi-linear interpolation and the cubic spline interpolation, respectively, in Fig. 4b, c.

To verify Theorem 2, we compute the L_2 errors in the flow map and the line integral versus Δx from $1/32$ to $1/512$. Figure 5a shows the error in the two components of the flow map. As can be seen, our numerical solution has approximately second order accuracy which matches with the claim in Theorem 1. In Fig. 5b we plot the L_2 errors of the line integral f using different Δx's ranging from $1/32$ to $1/512$. As can be seen, the solution of f is also approximately second order accurate satisfying Theorem 2. This figure also compares the error of our proposed Eulerian approach with the errors in the Lagrangian solutions using different interpolation methods. All these approaches are second-order accurate. Note that since we have no exact solution for this example, all these errors are computed by comparing with the Eulerian solution with a very small time step and mesh size.

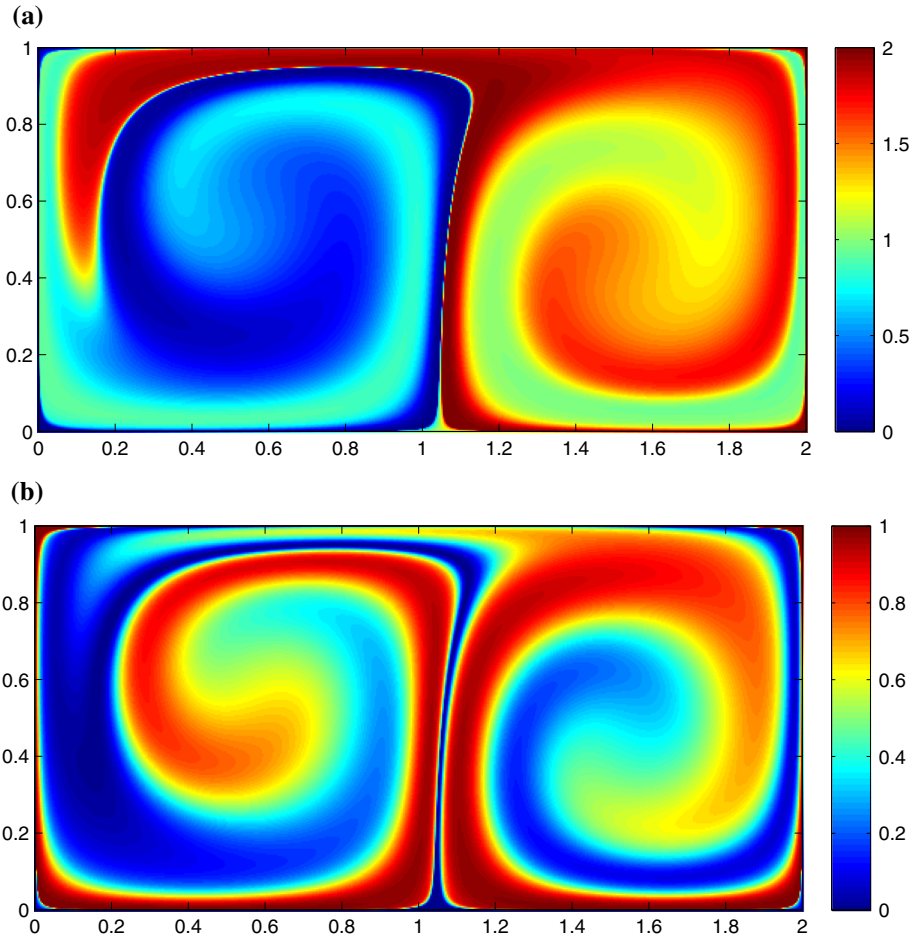

Fig. 3 (Section 5.1) Eulerian approach to Problem F using TVD-RK2 and *cubic spline*. The numerical solution of Φ_0^{10} and two components are shown in **a** and **b**, respectively

In Fig. 6 we compare the errors in our proposed Eulerian methods for Problem F using different interpolation methods and different PDE methods for the temporal direction. We use only simple bi-linear interpolation in Fig. 6a, while keeping the cubic spline interpolation in Fig. 6b. In each of these two figures, we show the errors in the first order Euler method, the TVD-RK2 and also the TVD-RK3 method. Using the simple Euler method, we find that the solution is always approximately first order accurate no matter whether bi-linear or cubic spline is used in the interpolation step, as shown in the red solid lines in both figures. This is because Euler method is first order accurate in time and, therefore, first order accurate in space (due to the CFL stability condition). When TVD-RK2 is used in solving the PDEs, the solutions become second order accurate as shown in blue dash dot lines. This verifies the claim in Theorem 2. Note also that even though solutions from TVD-RK2 show second order accuracy for both bi-linear and cubic spline interpolation, the errors in using cubic spline are almost one order of magnitude smaller than that from the bi-linear interpolation.

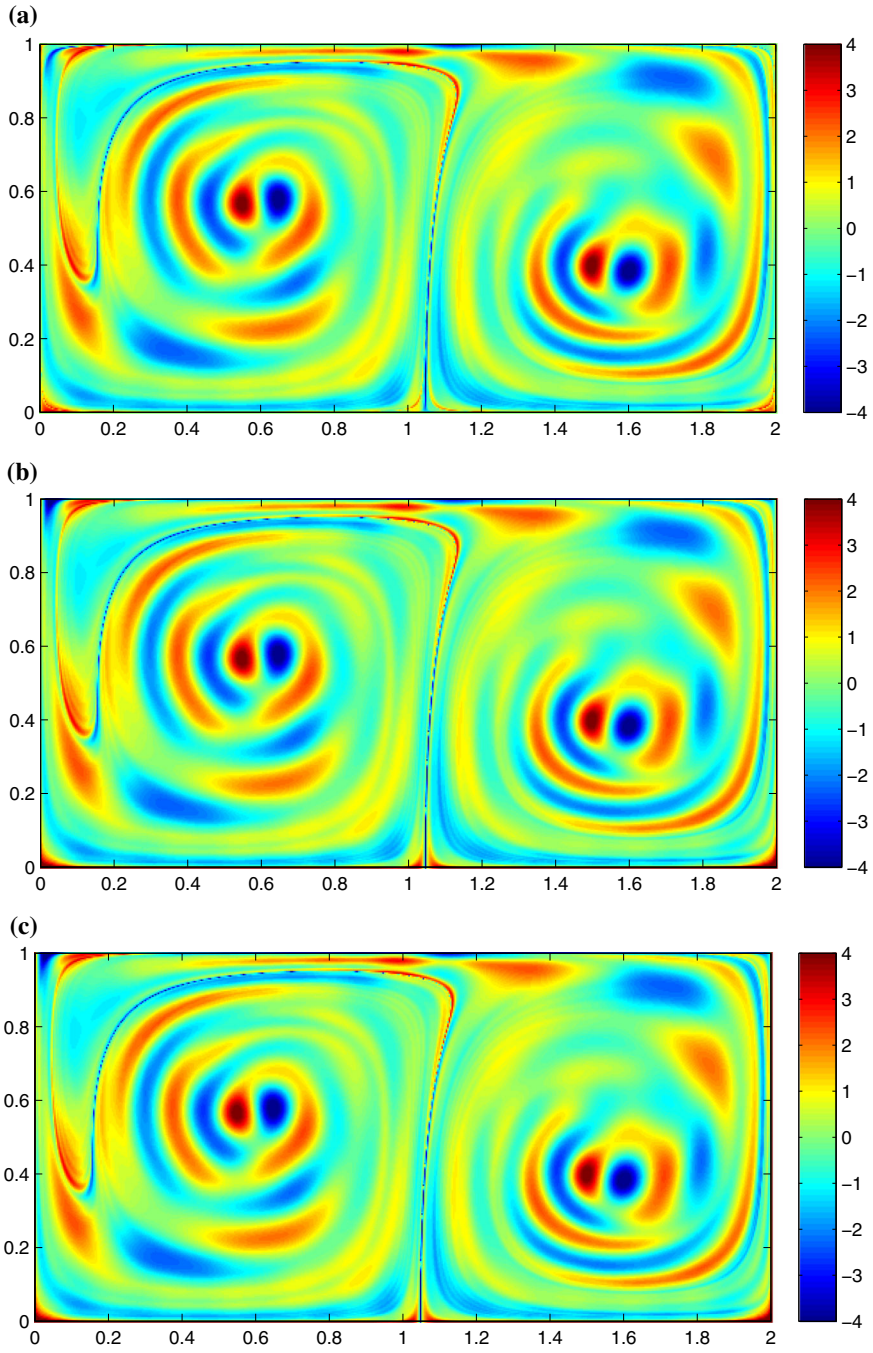

Fig. 4 (Section 5.1) **a** Eulerian approach to Problem F. The line integral of $g(x, y) = \cos(12\pi x)\cos(2\pi y)$ along the particle trajectory starting from $\mathbf{x}_{i,j}$ at $t = 0$ to $t = 10$. The solution is computed using TVD-RK2 with cubic spline interpolation. **b**, **c** Lagrangian approach to Problem F using TVD-RK2 where the velocity field is interpolated with **b** bi-linear interpolation and **c** cubic spline interpolation

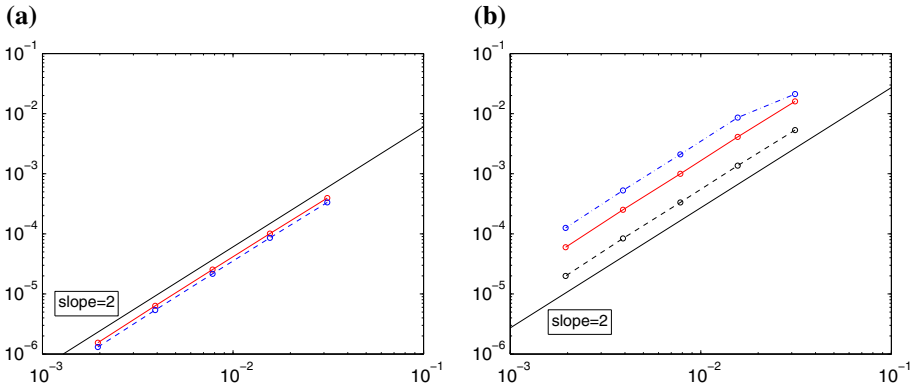

Fig. 5 (Section 5.1) Eulerian approach to Problem F. **a** L_2 error of the solution Φ_0^{10} with different Δx's using TVD-RK2 and cubic spline interpolation. **b** The L_2 error of the solution f with different Δx's computed using our proposed Eulerian approach with cubic spline (*black dashed line*), the Lagrangian approach with the cubic spline interpolation (*red solid line*) and the bi-linear interpolation (*blue dashed line*). Reference *lines* with slope 2 are plotted in *black solid line* (Color figure online)

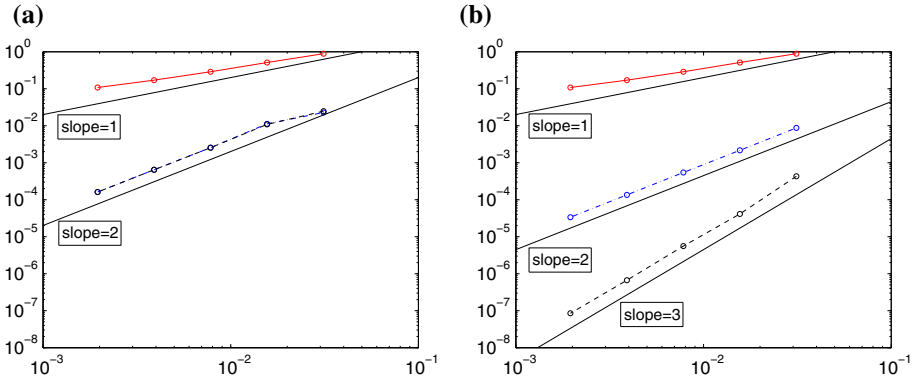

Fig. 6 (Section 5.1) Eulerian approach to Problem F. The L_2 error of the solution f with different Δx's using the proposed Eulerian scheme with **a** bi-linear interpolation and **b** cubic spline. *Red solid line* First order Euler method. *Blue dash dot line* TVD-RK2. *Black dash line* TVD-RK3. Reference lines of slope 1, 2 and 3 are plotted in *solid black* (Color figure online)

Finally, we have shown the errors when using the TVD-RK3 in black dash lines. With the bi-linear interpolation (which is second order accurate in space), we are not able to observe the third order convergence in the solution. The errors in the solution are clearly dominated by the interpolation error, as shown in Fig. 6a. When cubic spline is used instead, as shown in Fig. 6b, the errors in the solution do show an approximately third order convergence.

5.2 An Application to the Ocean Surface Current Analyses Real-Time (OSCAR) Dataset

As discussed in Sect. 3, the proposed approaches require only discrete velocity data at mesh points in computing the forward (backward) flow map and the corresponding forward (backward) line integrals of certain function along particle trajectories. To better demonstrate this

point, in this section we consider the Ocean Surface Current Analyses Real-time (OSCAR) data which includes velocity data only at discrete locations. The OSCAR data were obtained from JPL Physical Oceanography DAAC and developed by ESR. The data covers $-80°$ to $80°$ latitude and $0°$ to $360°$ longitude. The time resolution is about 5 days and the spatial resolution is $1/3°$ in each direction. An ocean region near the Line Islands is chosen as our computational domain, which is enclosed by $E180°$ (180 degrees East) to $E230°$ longitude and $S17°$ to $N8°$ latitude. To have a better visualization, we use the interpolation techniques to obtain the velocity data with a finer resolution of 0.25 days in the temporal direction and $1/3°$ in each spatial direction. We have to emphasize that the interpolation step is done before the whole algorithm only for a better visualization and no interpolation on velocity data is needed within the algorithm. In our numerical experiments, we use our proposed approach in Sect. 3.2 to simulate the ocean surface current within the first 50 days in year 2014.

One important quantity to visual a dynamical system is the finite time Lyapunov exponent (FTLE). This quantity studies the growth of an infinitesimal perturbation in an initial condition over a finite time period T. Mathematically, we have

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

where $\mathcal{D}\Phi(\mathbf{x}; t_0, T)$ is the spatial gradient or the Jacobian of the flow map. The leading order of the magnitude of this perturbation is given by

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

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

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

where $\mathbf{e}(0)$ aligns with the eigenvector associated with the largest eigenvalue of the deformation tensor $\lambda_{\max}[\Delta(\mathbf{x}; t_0, T)]$. Using this quantity, we define the FTLE using $\sigma^T(\mathbf{x}, t_0) = \ln\sqrt{\lambda_{\max}[\Delta(\mathbf{x}; t_0, T)]}/|T|$.

Figure 7a shows the forward FTLE field $\sigma_0^{50}(\mathbf{x})$ while Fig. 7b gives the forward line integral of the function $f(x, y) = \cos(0.5\pi x)\cos(\frac{1}{12}\pi y)$ starting from the point (x, y) at $t = 0$ along the particle trajectory up to $t = 50$.

5.3 An Application to the Coherent Ergodic Partition

We have proposed a numerical approach in [43] to extract invariant sets in a continuous dynamical system in the extended phase space. We call it the coherent ergodic partition of the system. To obtain the solution, we need to numerically compute the long time flow map $\Phi_0^T(\mathbf{x})$ for large enough $T > 0$ and also the required time averages of observables along particle trajectories. For a particular observable $g(\mathbf{x}, t) = g(\mathbf{x})$, the time average $g^\star$ is computed as

$$g^\star(\mathbf{x}) = \lim_{T\to\infty}\frac{1}{T}\int_0^T (g\circ\Phi_0^\tau)(\mathbf{x})d\tau.$$

These quantities required in computing the coherent ergodic partition can be more efficiently approximated with the new approaches proposed in Sect. 3. We take the double-gyre flow for example and use $A = 0.1$, $\epsilon = 0.1$, $\omega = 2\pi/10$. We discretize the domain

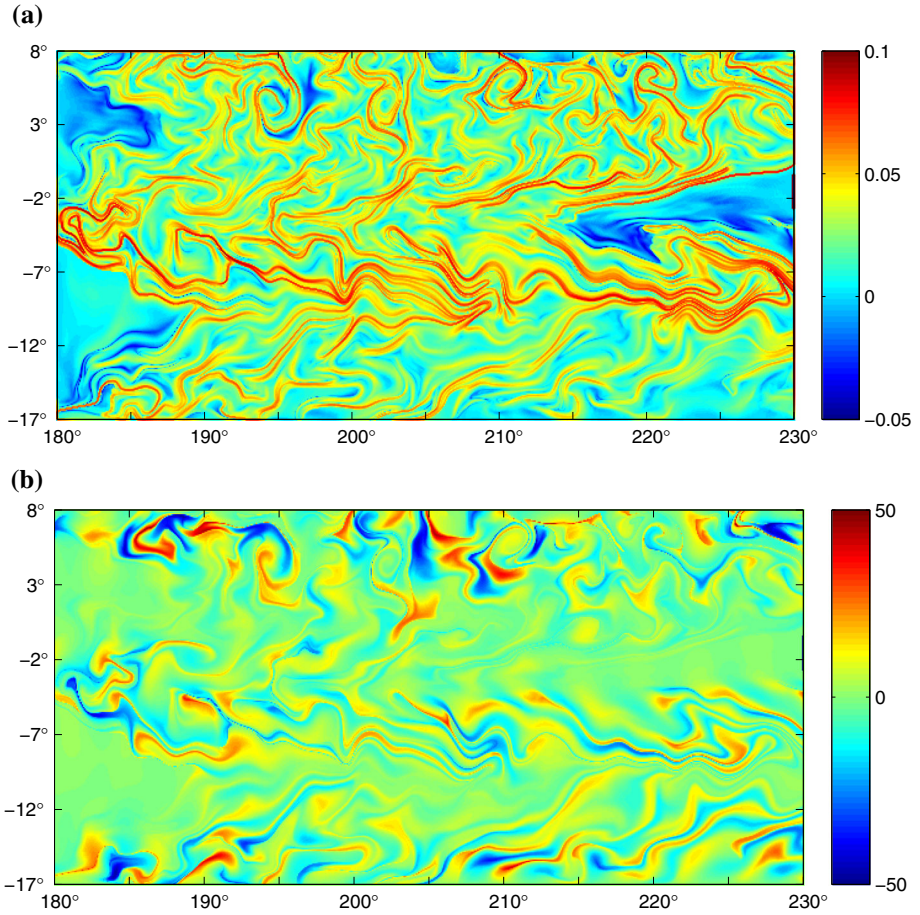

Fig. 7 (Section 5.2) **a** The FTLE field $\sigma_0^{50}(\mathbf{x})$ of OSCAR dataset. **b** The line integral of $f(x, y) = \cos(0.5\pi x)\cos(\frac{1}{12}\pi y)$ from $t = 0$ to $t = 50$ along particle trajectories

$[0, 2] \times [0, 1]$ using 513 grid points in the x-direction and 257 grid points in the y-direction. This gives $\Delta x = \Delta y = 1/256$. We first solve the level set equations from $t = t_0 = 0$ to $t = T_m = 10$ using the approach in Sect. 3.2 to obtain $\Phi_0^{T_m}$ and use (12) to obtain $\int_0^{T_m}(g \circ \Phi_0^\tau)(\mathbf{x})d\tau$. Then we iterate the flow map and also the line integral 5 times to obtain $\Phi_0^T(\mathbf{x})$ and $\int_0^T(g \circ \Phi_0^\tau)(\mathbf{x})d\tau$ for $T = T_m \cdot 2^5 = 320$.

In Fig. 8a, we have shown the quantity $\frac{1}{T}\int_0^T(g \circ \Phi_0^\tau)(\mathbf{x})d\tau$ for $T = 320$ using $g(\mathbf{x}) = \cos(2\pi y)$. The whole computational domain is roughly partitioned into several regions including two circular regions centered near $(0.5, 0.5)$ and $(1.5, 0.5)$, two arrow regions slightly below and slightly above these two circular regions respectively, and also the background region which contains many isolated dots. Within each of these regions, the approximated value of $g^\star$ shows similar quantity. To better distinguish these regions, we use four independent test functions $g_1 = \cos(2\pi y)$, $g_2 = \cos(12\pi x)\cos(2\pi y)$, $g_3 = \sin(2\pi y)$, and $g_4 = \frac{x}{2}$, and segment these $\mathbb{R}^4$ data into several clusters with the geometrical constraint in $\mathbb{R}^2$ as shown in Fig. 8. For grid points within the same cluster, we color it using the same

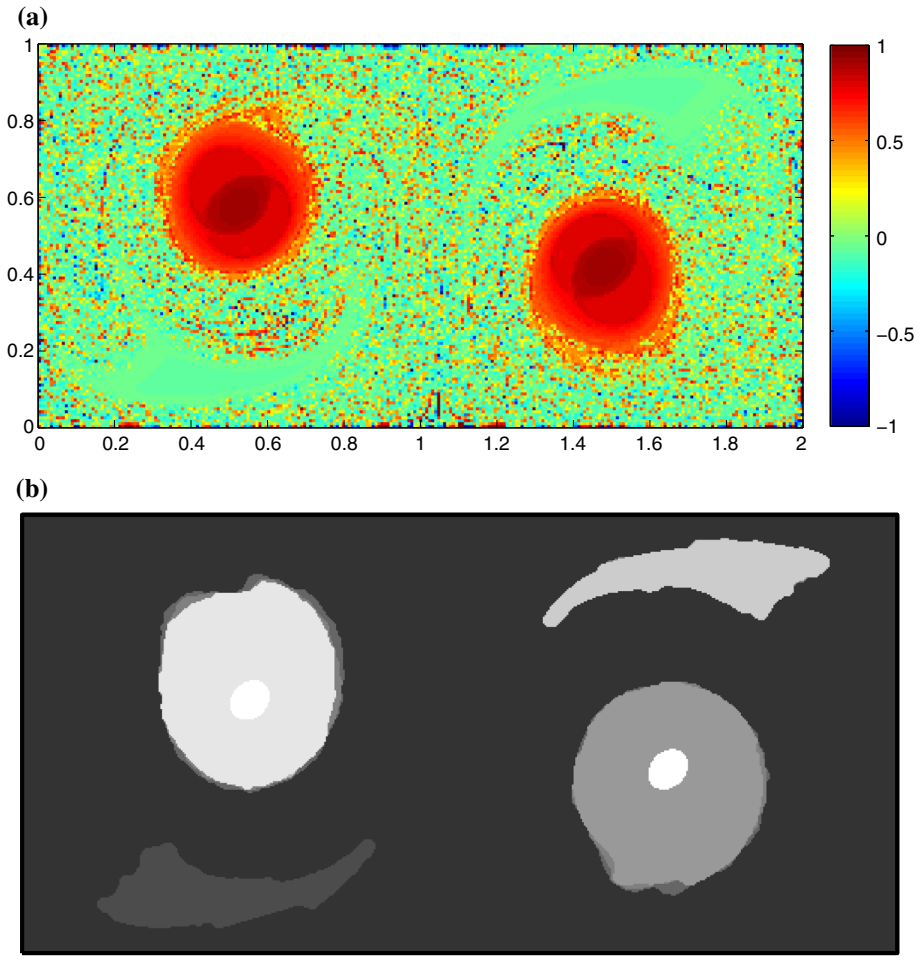

Fig. 8 (Section 5.3) **a** The time average of the function $g(\mathbf{x}) = \cos(2\pi y)$. **b**The coherent ergodic partition at $t = 0$ with (g_1, g_2, g_3, g_4) using $T = 320$

gray-level intensity, i.e. each group is colored in the same intensity. All these results match well with those in [43].

5.4 An Application to Geometrical Optics

Geometrical optics is an important class of asymptotic approximation to high frequency wave propagation. In the high frequency regime when $\omega \to \infty$, we approximate the phase function by the eikonal equation $|\nabla T| = \frac{1}{c}$. To obtain the multivalued solution to the traveltime field T in two dimensions, we reformulate the problem in the phase space by solving the eikonal equation using the Lagrangian formulation by the method of characteristics [5],

$$\frac{dx}{dt} = c \sin\theta \,,$$
$$\frac{dy}{dt} = c \cos\theta \,,$$

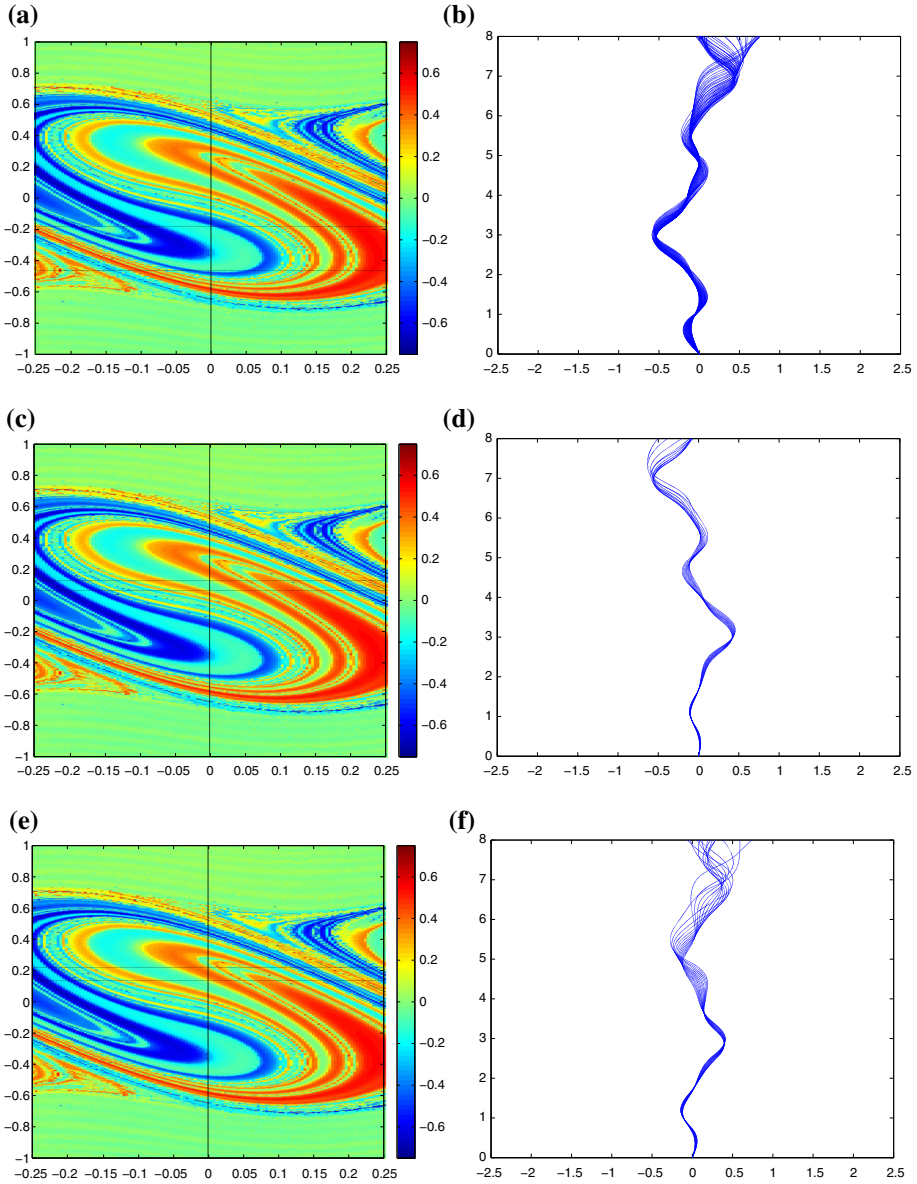

Fig. 9 (Section 5.4) The angle subintervals sharing the same *color* are shown in **a** $[-0.4625, -0.18125]$, **c** $[0.0625, 0.125]$ and **e** $[0.1375, 0.21875]$; **b**, **d**, **f** clusters of rays emitted from the origin corresponding to **a**, **c** and **e**, respectively (Color figure online)

$$\frac{d\theta}{dt} = c_y \sin\theta - c_x \cos\theta \,. \tag{16}$$

If we use depth as the running parameter, then we have a reduced system

$$\frac{dx}{dy} = \tan\theta \,,$$

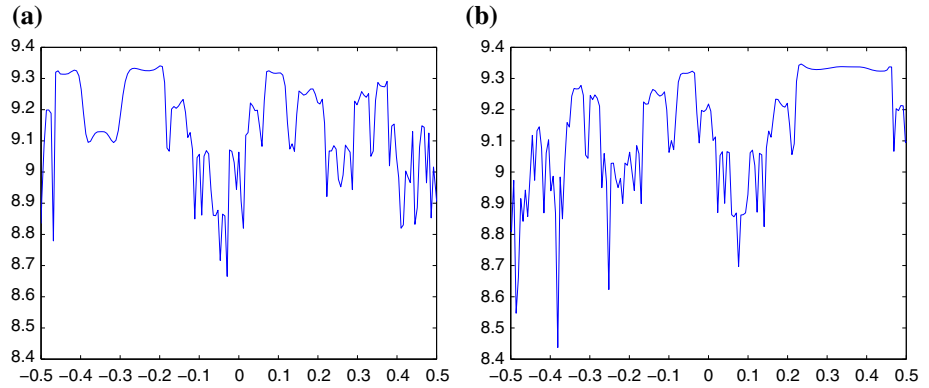

Fig. 10 (Section 5.4) The travel time of rays emitted from **a** $x = 0$, **b** $x = -0.123$ with departure angles in $[-0.5, 0.5]$

$$\frac{d\theta}{dy} = \frac{c_y}{c}\tan\theta - \frac{c_x}{c}\,. \tag{17}$$

We used the coherent ergodic partition as a tool to study the ray spreading in [43]. In particular, we succeeded to couple the emitted rays into clusters and rays classified into the same cluster have very close behaviors. Therefore, theoretically in the adaptive method, one can simply refine the angle space near the boundaries of these clusters.

In [43], we used $c = 1 + 0.2\sin(0.5\pi y)\sin(3\pi(x + 0.55))$ and $g = \sin x$. Then we solved the above reduced ray tracing system up to $y = 8$ and computed the value of

$$F(x, \theta, 8) \triangleq \int_0^8 f \circ \phi_y(x, \theta)\, dy$$

to form our coherent ergodic clusters.

In this section, we use (12) to recompute $F(x, \theta, 8)$ using another test function $g = \sin(5x)$. As can be seen from Fig. 9a, the solution shares similar structures with the solution corresponding to $g = \sin(x)$ and thus the classification of the clusters is almost the same. For example, if we focus on the cross section $x = 0$ which corresponds to the rays emitting from the origin with different departure angles, we observe that there exists three subintervals $\theta \in [-0.4625, -0.18125]$, $[0.0625, 0.125]$ and $[0.1375, 0.21875]$ which share the same color. This suggests that rays emitting from one of these subintervals may travel as a patch in the y-direction, as demonstrated in Fig. 9. This result matches exactly with our observation in [43].

We also compute the line integral of $g(x, \theta, y) = 1/[c(x, y)\cos\theta]$ which gives the travel time of each individual ray. Figure 10a, b show the travel time of rays emitted from $x = 0$ and $x = -0.123$ respectively where the horizontal axis denotes the departure angles.

6 Conclusion

We have developed a new class of Eulerian methods for constructing both the *forward* and the *backward* flow maps of sufficiently smooth dynamical systems. Unlike those previous Eulerian methods, the proposed method can determine the *forward* flow map *on the fly*

without the need of storing all intermediate flow fields. The article also gives a computational complexity analysis and an error estimate of these Eulerian methods.

Because of the simplicity in the implementation, this class of methods can be applied to various applications in computational dynamical systems such as the computations of the so-called VIALS [44], the FTLE, FSLE and the infinitesimal size Lyapunov exponent (ISLE) [45]. These methods can also be easily coupled with different Eulerian methods from the level set community. For example, one possibility is to follow the adaptive mesh refinement level-set methods [29–31] to develop an adaptive strategy for LCS extraction.

Acknowledgements The work of You was supported by the Natural Science Foundation of Jiangsu Higher Education Institutions of China (No. 16KJB110012) and the National Natural Science Foundation of China (61673221). The work of Leung was supported in part by the Hong Kong RGC Grants 16303114 and 16309316.

References

1. Artale, V., Boffetta, G., Celani, A., Cencini, M., Vulpiani, A.: Dispersion of passive tracers in closed basins: beyond the diffusion coefficient. Phys. Fluids **9**(11), 3162–3171 (1997)
2. Aurell, E., Boffetta, G., Crisanti, A., Paladin, G., Vulpiani, A.: Predictability in the large: an extension of the concept of Lyapunov exponent. J. Phys. A: Math. Gen. **30**, 1–26 (1997)
3. Candès, E.J., Ying, L.: Fast geodesics computation with the phase flow method. J. Comput. Phys. **220**, 6–18 (2006)
4. Cencini, M., Vulpiani, A.: Finite size Lyapunov exponent: review on applications. J. Phys. A: Math. Theor. **46**, 254019 (2013)
5. Cerveny, V., Molotkov, I.A., Psencik, I.: Ray Method in Seismology. Univerzita Karlova Press, Praha (1977)
6. Courant, R., Issacson, E., Rees, M.: On the solution of nonlinear hyperbolic differential equations by finite differences. Commun. Pure Appl. Math. **5**, 243–255 (1952)
7. de Boor, C., Swartz, B.: Piecewise monotone interpolation. J. Approx. Theory **21**, 411–416 (1977)
8. Enright, D., Losasso, F., Fedkiw, R.: A fast and accurate semi-Lagrangian particle level set method. Comput. Struct. **83**, 479–490 (2005)
9. Fritsch, F.N., Carlson, R.E.: Monotone piecewise cubic interpolation. SIAM J. Numer. Anal. **17**, 238–246 (1980)
10. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Phys. D **149**, 248–277 (2001)
11. Haller, G.: Lagrangian structures and the rate of strain in a partition of two-dimensional turbulence. Phys. Fluids A **13**, 3368–3385 (2001)
12. Haller, G., Yuan, G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. Phys. D **147**, 352–370 (2000)
13. Hermandez-Carrasco, I., Lopex, C., Hernansez-Garcia, E., Turiel, A.: How reliable are finite-size Lyapunov exponents for the assessment of ocean dynamics? Ocean Model. **36**(3–4), 208–218 (2011)
14. Huynh, H.T.: Accurate monotone cubic interpolation. NASA Technical Memorandum 103789 (1991)
15. Lekien, F., Marsden, J.E.: Tricubic interpolation in three dimensions. Int. J. Numer. Methods Eng. **63**, 455–471 (2005)
16. Lekien, F., Shadden, S.C., Marsden, J.E.: Lagrangian coherent structures in n-dimensional systems. J. Math. Phys. **48**, 065404 (2007)
17. Lentine, M., Gretarsson, J.T., Fedkiw, R.: An unconditionally stable fully conservative semi-Lagrangian method. J. Comput. Phys. **230**, 2857–2879 (2011)
18. Leslie, L.M., Pursuer, R.J.: Three-dimensional mass-conserving semi-Lagrangian scheme employing forward trajectories. Mon. Weather Rev. **123**, 2551–2566 (1995)
19. Letz, T., Kantz, H.: Characterization of sensitivity to finite perturbations. Phys. Rev. E. **61**, 2533 (2000)
20. Leung, S.: An Eulerian approach for computing the finite time Lyapunov exponent. J. Comput. Phys. **230**, 3500–3524 (2011)
21. Leung, S.: A backward phase flow method for the finite time Lyapunov exponent. Chaos **23**, 043132 (2013)
22. Leung, S., Qian, J.: Transmission traveltime tomography based on paraxial Liouville equations and level set formulations. Inverse Probl. **23**, 799–821 (2007)

23. Leung, S., Qian, J.: Eulerian Gaussian beams for Schrödinger equations in the semi-classical regime. J. Comput. Phys. **228**, 2951–2977 (2009)
24. Leung, S., Qian, J.: The backward phase flow and FBI-transform-based Eulerian Gaussian beams for the Schrödinger equation. J. Comput. Phys. **229**, 8888–8917 (2010)
25. Leung, S., Qian, J., Burridge, R.: Eulerian Gaussian beams for high frequency wave propagation. Geophysics **72**, SM61–SM76 (2007)
26. Liu, X.D., Osher, S.J., Chan, T.: Weighted essentially nonoscillatory schemes. J. Comput. Phys. **115**, 200–212 (1994)
27. Mills, P.: Following the vapour trail: A study of chaotic mixing of water vapour in the upper troposphere. Thesis, University of Bremen, Germany (2004)
28. Mills, P.: Isoline retrieval: an optimal sounding method for validation of advected contours. Comput. Geosci. **35**, 2020–2031 (2009)
29. Min, C.: Local level set methods in high dimension and codimension. J. Comput. Phys. **200**(1), 368–382 (2004)
30. Min, C., Gibou, F.: A second order accurate level set method on non-graded adaptive cartesian grids. J. Comput. Phys. **225**, 300–321 (2007)
31. Mirzadeh, M., Guittet, A., Burstedde, C., Gibou, F.: Parallel level-set methods on adaptive tree-based grids. J. Comput. Phys. **322**, 345–364 (2016)
32. Osher, S.J., Fedkiw, R.P.: Level Set Methods and Dynamic Implicit Surfaces. Springer, New York (2003)
33. 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)
34. Osher, S.J., Shu, C.W.: High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations. SIAM J. Numer. Anal. **28**, 907–922 (1991)
35. Passow, E.: Piecewise monotone spline interpolation. J. Approx. Theory **12**, 240–241 (1974)
36. Qian, J., Leung, S.: A level set based Eulerian method for paraxial multivalued traveltimes. J. Comput. Phys. **197**, 711–736 (2004)
37. Sethian, J.A.: Level Set Methods. Cambridge University Press, Cambridge (1996)
38. 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. Phys. D **212**, 271–304 (2005)
39. 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)
40. Smolarkiewicz, P.K., Grell, G.A.: A class of monotone interpolation schemes. J. Comput. Phys. **101**, 431–440 (1992)
41. Staniforth, A., Cote, J.: Semi-Lagrangian integration schemes for atmospheric model—a review. Mon. Weather Rev. **119**, 2206–2223 (1991)
42. Ying, L., Candès, E.J.: The phase flow method. J. Comput. Phys. **220**, 184–215 (2006)
43. You, G., Leung, S.: An Eulerian method for computing the coherent ergodic partition of continuous dynamical systems. J. Comput. Phys. **264**, 112–132 (2014)
44. 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)
45. You, G., Wong, T., Leung, S.: Eulerian methods for visualizing continuous dynamical systems using Lyapunov exponents. SIAM J. Sci. Comput. **39**(2), A415–A437 (2017)