Norwegian University of Science and Technology

Physics, Specialization Project

Sensitivity to Numerical Integration Scheme in Calculation of Lagrangian Coherent Structures

Author:
Arne Magnus Tveita LØKEN

Supervisor:
Assoc. Prof. Tor NORDAM

Department of Physics
Norwegian University of Science and Technology
N-7491 Trondheim, Norway

December 18, 2017

Abstract

A generic flow system can be described as a structure whose state depends on flowing streams of energy, material or information. Traditional examples of such systems are the transport of properties such as pressure, temperature or matter in fluids, and the transport of charge in electrical currents. However, valuable insight into various phenomena, including, but not limited to, the development of algae in the ocean and crowd patterns formed by humans, can be obtained by regarding them as flow systems.

Lagrangian coherent structures (henceforth abbreviated to LCSs) can be described as 'landscapes' within a multidimensional space, which exert a major influence upon the flow patterns in dynamical systems. Compared to the conventional approach of increasing the spatial and temporal resolution of the model(s) involved in numerical simulations, LCSs provide a simplified means of predicting the future states of complex systems. In this context, a complex system is a system which exhibits sensitive dependence to initial conditions. From their variational theory, hyperbolic LCSs are identified as locally most repelling or attractive material surfaces. Somewhat simplified, such surfaces can be considered as generalized trajectories, created by the underlying transport phenomenon. When investigating transport systems, hyperbolic LCSs are of particular interest, as they form the skeleton of observable flow patterns.

This project is centered around the dependence of the numerical identification of such LCSs upon which numerical integration scheme was used in order to simulate the underlying transport phenomenon. LCSs were computed and compared for an analytical double gyre velocity field, for which reference LCSs are documented in the literature. In particular, four traditional singlestep and four embedded, adaptive stepsize Runge-Kutta methods were investigated. A numerical solution obtained by means of a high order embedded method with a very small numerical tolerance — two orders of magnitude smaller than the smallest tolerance level utilized otherwise — and thus a very small expected numerical error, was utilized as a reference solution for comparing the performance of the different methods.

It turns out that the LCSs for the system under consideration are quite robust numerically. Thus, none of the numerical integration schemes stand out as the absolutely best-suited. However, based upon the errors of the computed flow maps, high order integration methods are generally advisable for generic flow systems, as they invariably yield more efficient calculations, which are less susceptible to numerical round-off error. Moreover, there is no apparent reason to expect the LCSs of a generic flow system to be as robust as they were for the system considered for this project work.

Note that the numerical implementation of one of the LCS existence conditions which arise from their variational theory has not yet (to the author's knowledge) been described completely in the literature. The approach considered in this project is based on careful inspection of properties of the underlying system, which might not be applicable to generic systems, depending on their scale. In short, there is currently a lot of room for research within the field.

Sammendrag

Et generisk strømningssystem kan beskrives som en struktur hvis tilstand avhenger av flyt av energi-, material- eller informasjonsstrømmer. Tradisjonelle eksempler på slike systemer er transport av trykk, temperatur og masse i fluidstrømninger, og ladningstransport som forårsakes av elektriske strømmer. Interessante sider ved andre fenomen, som algeoppblomstring og folkesamlinger, kan avdekkes ved å betrakte dem som strømningssystemer.

Lagrange-koherente strukturer (heretter forkortet til LKSer) kan beskrives som «landskap» i flerdimensjonale rom, som utøver stor innflytelse på strømningsmønstre i dynamiske system. Sammenlignet med å utføre simuleringer på modeller med økende grad av oppløsning i rom og tid – hvilket er den tradisjonelle tilnærmingen – kan LKSer benyttes til å på forenklet vis kunne forutsi fremtidige tilstander i komplekse system. I denne konteksten betegner et komplekst system et system som er svært sensitivt til dets initialbetingelser. Fra deres underliggende variasjonsteori, kjennetegnes hyperbolske LKSer som de lokalt sterkest frastøtende eller tiltrekkende materialoverflatene i systemet. Noe forenklet kan denne typen overflater betraktes som en generalisering av banene det underliggende transportfenomenet skaper. Ved analyse av transportsystemer er hyperbolske LKSer spesielt interessante, ettersom disse utgjør skjelettet av observable strømningsmønstre.

Dette prosjektet omhandler hvordan den numeriske identifikasjonen av LKSer avhenger av den numeriske integrasjonsmetoden som ble benyttet til å simulere det underliggende transportfenomenet. LKSer ble beregnet og sammenlignet ut fra et analytisk kjent hastighetsfelt, hvis LKSer er gjort kjent i litteraturen. Spesifikt ble åtte Runge-Kutta-metoder undersøkt, hvorav fire tradisjonelle enkeltstegsmetoder, og fire sammensatte, dynamisk steglengde-metoder. En løsning funnet via en høyere ordens sammensatt metode med svært liten numerisk toleranse – to størrelsesordener mindre enn det minste toleransenivået som ble benyttet for øvrig – og dermed veldig liten forventet numerisk feil, ble brukt som referanseløsning.

Det viser seg at LKSene i eksempelsystemet er nokså robuste, numerisk sett. Av den grunn sto ingen av de numeriske integrasjonsmetodene frem som den absolutt mest velegnede. Likevel, basert på de de beregnede feilene i flytdiagrammene, tilrådes bruk av høyere ordens numeriske integratorer generelt, fordi de uten unntak resulterer i mer effektive beregninger, som er mindre utsatt for numeriske avrundingsfeil. Videre foreligger ingen åpenbar grunn til å forvente at LKSene for et generelt strømningssystem er like robuste som de viste seg å være for modellsystemet undersøkt her.

Merk at den numeriske implementasjonen av en av eksistensbetingelsene for LKSer, som stammer fra deres variasjonsteori, til dags dato (så vidt undertegnede vet) ikke har blitt fullstendig beskrevet i litteraturen. Tilnærmingsmetoden som ble benyttet i dette prosjektet, er basert på nøye vurderinger av det underliggende systemets egenskaper, hvilket ikke nødvendigvis er en praktisk hensiktsmessig fremgangsmåte for generiske system, avhengig av de involverte skalaene. Med andre ord er det for øyeblikket stort rom for forskning innen feltet.

Preface

This thesis is submitted as part of the formal requirements of the subject TFY4510 — 'Physics, Specialization Project' at NTNU, accounting for a total of 15 ECTS. All of the underlying work was performed in Trondheim, during the fall semester of 2017, that is, the 9^{th} semester of my enrollment in the 5 year study programme culminating in a M.Sc. in 'Physics and Mathematics', with a specialization in Applied Physics.

I would like to extend my gratitude to my supervisor, Assoc. Prof. Tor Nordam. Our many discussions have provided me with great insight, while his great knowledge, and humour proved excellent tools in terms of relaxing my early-semester nerves. Coupled with his expertise with regards to the use supercomputers for scientific purposes — without the use of which, by the way, several of the computations constituting this thesis would not have been practically feasible — I could not be more satisfied. I am already looking forwards to the continuation of our cooperation, for the upcoming spring term.

My dear friend and fellow student Simon Nordgreen, who has also performed his project work under the guidance of Assoc. Prof. Nordam, definitely deserves a mention. His and my main research topics are closely correlated, which lead to the two of us to many laborious, yet fruitful, collaborations regarding our understanding of the underlying variational principles. I think it is safe to say that two heads think better than one. To all the friends and acquaintances I have had the pleasure of getting to know during my time in Trondheim, I extend my thanks.

Last, but most certainly not least, I gratefully thank my family for their unconditional support and encouragement throughout my time as a university student.

Trondheim, December 2017
Arne Magnus Tveita Løken

Table of Contents

List of Figures

List of Tables

Notation

Newton's notation is used for differentiation with respect to time, i.e.:

$$\dot{f}(t) \equiv \frac{\mathrm{d}f(t)}{\mathrm{d}t}.$$

Vectors are denoted by upright, bold letters, like this:

$$\boldsymbol{\xi} = \left(\xi_1, \xi_2, \ldots, \xi_n\right).$$

The Euclidean norm of a vector $\boldsymbol{\xi} \in \mathbb{R}^n$ is denoted by:

$$\|\boldsymbol{\xi}\| = \sqrt{\xi_1^2 + \xi_2^2 + \ldots + \xi_n^2}.$$

Matrices and matrix representations of rank-2 tensors are denoted by bold, italicized letters, as follows:

$$\boldsymbol{A} = (a_{i,j}) = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{pmatrix}.$$

1 Introduction

When analyzing complex dynamical systems, such as the nonlinear many-body problems arising in transport phenomena by virtue of oceanic currents or atmospheric winds, the conventional approach to predicting future states by means of simulating the trajectories of phase points is frequently insufficient. This is due to the resulting predictions being very sensitive to small changes in time and initial positions. One way of addressing the delicate dependence on initial conditions is to run different models for the same underlying physical systems, of increasing spatial and temporal resolution. This sort of approach is made possible because the fundamental dynamics are known — yet, for complex transport systems, the computational cost quickly grows beyond the available resources, in terms of computation time or memory. For many practical purposes, however, microscopic details matter little in comparison to the overarching trends in the system, which means that a less ambitious approach, merely aiming to understand the macroscopics of the transport phenomenon, is often justifiable.

At the turn of the millennium, the concept of Lagrangian coherent structures saw the light of day, emerging from the intersection between nonlinear dynamics, that is, the underlying mathematical principles of chaos theory, and fluid dynamics (Haller and Yuan 2000). These provide a new framework for understanding transport phenomena in conceptual fluid flow systems. Lagrangian coherent structures can be described as time-evolving 'landscapes' in a multidimensional space, which dictate macroscopical flow patterns in dynamical systems. In particular, such structures define the interfaces of dynamically distinct, invariant regions. An invariant region in fluid dynamics is characterized as a domain where all particle trajectories that originate within the region, remain in it, although the region itself can move and deform with time. So, simply put; Lagrangian coherent structures enable us to make predictions regarding the future states of flow systems.

There are two possible perspectives regarding the description of fluid flow. The Eulerian approach is to consider the properties of a flow field at a set of fixed points in time and space. An example is the concept of velocity fields, which produce the local and instantaneous velocities at all points within their domains. The Lagrangian point of view, on the other hand, concerns the developing velocity of each fluid element along their paths, as they are transported by the flow. Unlike the Eulerian perspective, the Lagrangian mindset is objective, as in frame-invariant. That is, properties of Lagrangian fields are unchanged by time-dependent translations and rotations of the reference frame. For unsteady flow systems, which are more common than steady flow systems in nature, there exists no self-evident preferred frame of reference. Thus, any transport-dictating dynamic structures should hold for *any* choice of reference frame. This is the main rationale for which *Lagrangian*, rather than *Eulerian*, coherent structures have been pursued.

A generic flow system can be described as a structure whose state depends on flowing streams of energy, material or information. Conventional examples of flow systems include the transport of physical properties such as pressure, temperature or matter in fluids, and the transport of

charge in electrical currents. A large variety of phenomena can reasonably be modelled as flow systems, such as the classical harmonic oscillator, or the interaction between predator and prey in closed systems (Strogatz 2014, parts I–II). In doing so, valuable pieces of insight can be obtained from well-understood properties of generic flows. In recent years, analyses based upon Lagrangian coherent structures have been conducted for a variety of naturally occuring phenomena which are not commonly considered as flow systems. Two prominent examples are how Olascoaga et al. (2008) used Lagrangian coherent structures in order to forecast the development of toxic algae in the ocean, and Ali and Shah (2007) used Lagrangian coherent structures to predict the formation and stability of human crowd patterns. As these examples emphasize, the theory of Lagrangian coherent structures is applicable to a wide range of systems.

Although the framework for detecting Lagrangian coherent structures is mathematically valid for any number of dimensions, the focus of this project work has been two-dimensional flow systems. Many natural processes can reasonably be described as two-dimensional, perhaps most notably the transport of debris and contaminations, such as garbage patches or the remnants of an oil spill, by means of oceanic surface currents. Being able to successfully predict where such particles will be taken by the flow could enable us to isolate them and accelerate the cleanup process before the particles are able to reach the coastline, thus mitigating potential humanitarian and natural calamities.

At the heart of detecting Lagrangian coherent structures lies the advection of (generalized) fluid elements by means of a velocity field, which describes the system under consideration. This is true both for test cases, where the velocity profile is known analytically, and for real-life systems, typically described by means of some model for the instantaneous velocity field. For this project work, the topic of interest is how the detection of Lagrangian coherent structures depends on the choice of numerical integration method used in order to compute the aforementioned particle transport. In particular, four singlestep methods and four embedded, adaptive step length methods, each with different properties, were used in order to advect a collection of fluid elements by means of an analytically known, two-dimensional, unsteady velocity field.

This thesis is structured based on the idea that readers possessing at least an undergraduate level of knowledge of physics and mathematics, in addition to a rudimentary understanding of programming, should be able to understand and repeat the numerical investigations which have been conducted. To this end, the immediately forthcoming chapter contains a description of the various numerical integration schemes that were utilized, in addition to a generic yet brief mathematical description of the kind of flow systems considered and the Lagrangian coherent structures situated therein, the latter based on variational theory. In the ensuing chapter, we present a transport model frequently used in literature — whose Lagrangian coherent structures are documented — in addition to a description of how the variational principles of Lagrangian coherent structure detection were implemented numerically. Lastly, the results are presented and discussed, before the conclusions of the project as a whole are drawn.

2 Theory

2.1 Solving systems of ordinary differential equations

In physics, like other sciences, modeling a system often equates to solving an initial value problem. An initial value problem can be described in terms of an ordinary differential equation (hereafter abbreviated to ODE) of the form

$$\dot{x}(t) = f\Big(t, x(t)\Big), \quad x(t_0) = x_0, \tag{2.1}$$

where x is an unknown function (scalar or vector) of time t. The function f is defined on an open subset Ω of $\mathbb{R} \times \mathbb{R}^n$, where n is the number of spatial dimensions, that is, the number of components of x. The initial condition (t_0, x_0) is a point in the domain of f, i.e., $(t_0, x_0) \in \Omega$. In higher dimensions (that is, $n > 1$) the differential equation (2.1) generally extends to a coupled family of ODEs

$$\dot{x}_i(t) = f_i\Big(t, x_1(t), x_2(t), \ldots, x_n(t)\Big), \quad x_i(t_0) = x_{i,0}, \quad i = 1, \ldots, n. \tag{2.2}$$

The system is nonlinear if the function f in equation (2.1), or, if at least one of the functions f_i in equation (2.2), is nonlinear in one or more of its arguments. For the sake of notational simplicity, the discussion to follow in the rest of this chapter is based on the one-dimensional case, that is, system (2.1), for $n = 1$. However, all of the considerations also hold for $n > 1$.

Say that the solution of system (2.1) is sought at some time t_f. In order to approximate said solution numerically, the time variable must be discretized first. This is frequently done by defining

$$t_j = t_0 + j \cdot h, \tag{2.3}$$

where t_j is the time level j for integer j, and h is some time increment which is smaller than $t_f - t_0$. Typically, the time increment is chosen such that an integer number of step lengths h equals the difference $t_f - t_0$. With the discretized time, the numerical solution of system (2.1) is found by means of successive applications of some numerical integration method. The Runge-Kutta family of numerical methods for ODE systems is a common choice, and will be described in greater detail in section 2.1.1.

Generally, all numerical integration schemes fall into one of two categories, that is, explicit and implicit methods. Explicit methods are characterized by computing the state of a system at a later time, based on the state of the system at the current time (in some cases, the state at earlier times are also considered). Implicit methods, however, involve the solution of an equation where both the current and the later state of the system are involved. Thus, a generic, explicit method for computing the state of the system at time $t + h$ given its state at t can be expressed as

$$x(t + h) = F\Big(x(t)\Big), \tag{2.4a}$$

while, for implicit methods, an equation of the sort

$$G\Big(x(t), x(t+h)\Big) = 0, \tag{2.4b}$$

is solved to find $x(t+h)$.

Generally, implicit methods require the solution of a linear system at every time step. Clearly, implicit methods are more computationally demanding than explicit methods. The main selling point of implicit methods is that they are more numerically stable than explicit methods. This property means that implicit methods are particularly well-suited for *stiff* systems, i.e., physical systems with highly disparate time scales (Hairer and Wanner 1996, p.2). For such systems, most explicit methods are unstable, unless the time step h is made exceptionally small, rendering these methods practically useless. For *nonstiff* systems, however, implicit methods behave similarly to their explicit analogues in terms of numerical accuracy and convergence properties.

Irrespective of which numerical integration method is employed, one obtains an approximation of the true solution of the system (2.1) *at* the discrete time levels, that is,

$$x_j \approx x(t_j), \tag{2.5}$$

where $x(t)$ is the exact solution at time t. The accuracy of the approximation, however, depends on both the numerical integration method and the time step length h used for the temporal discretization, given in equation (2.3). For nonlinear systems, analytical solutions usually do not exist. Thus, such systems are often analyzed by means of numerical methods.

2.1.1 *The Runge-Kutta family of numerical ODE solvers*

In numerical analysis, the Runge-Kutta family of methods is a popular collection of implicit and explicit iterative methods, used in temporal discretization in order to obtain numerical approximations of the *true* solutions of systems like (2.1). The German mathematicians C. Runge and M. W. Kutta developed the first of the family's methods at the turn of the twentieth century (Hairer, Nørsett, and Wanner 1993, p.134). The general scheme of what is now known as a Runge-Kutta method is as follows:

Definition 1. Let s be an integer and $a_{1,1}, a_{1,2}, \ldots, a_{1,s}, a_{2,1}, a_{2,2}, \ldots, a_{2,s}, \ldots, a_{s,1}, a_{s,2}, \ldots, a_{s,s}$, $b_1, b_2, \ldots, b_s$ and $c_1, c_2, \ldots, c_s$ be real coefficients. Let h be the numerical step length used in the temporal discretization. Then, the method

$$\begin{aligned} k_i &= f\left(t_n + c_i h, x_n + h \sum_{j=1}^{s} a_{i,j} k_j\right), \quad i = 1, \ldots, s \\ x_{n+1} &= x_n + h \sum_{i=1}^{s} b_i k_i \end{aligned} \tag{2.6}$$

is called an *s-stage Runge-Kutta method* for the system (2.1).

The main reason to include multiple stages in a Runge-Kutta method, is to improve the numerical accuracy of the computed solutions. The *order* of a Runge-Kutta method can be defined as follows:

Definition 2. A Runge-Kutta method, given by equation (2.6), is said to be of *order* p if, for sufficiently smooth systems (2.1), the local error e_n scales as h^{p+1}, that is:

$$e_n = \|x_n - u_{n-1}(t_n)\| \leq Kh^{p+1}, \tag{2.7}$$

where $u_{n-1}(t)$ is the exact solution of the ODE in system (2.1) at time t, subject to the initial condition $u_{n-1}(t_{n-1}) = x_{n-1}$, and K is a numerical constant. This is true, if the Taylor series for the exact solution $u_{n-1}(t_n)$ and the numerical solution x_n coincide up to (and including) the term h^p.

The *global* error

$$E_n = x_n - x(t_n), \tag{2.8}$$

where $x(t)$ is the exact solution of system (2.1) at time t, accumulated by n repeated applications of the numerical method, can be estimated by

$$|E_n| \leq C \sum_{l=1}^{n} |e_l|, \tag{2.9}$$

where C is a numerical constant, depending on both the right hand side of the ODE in system (2.1) and the difference $t_n - t_0$. Making use of definition 2, the global error can be estimated by

$$\begin{aligned} |E_n| &\leq C \sum_{l=1}^{n} |e_l| \leq C \sum_{l=1}^{n} |K_l|\, h^{p+1} \\ &\leq C \max_l \left\{|K_l|\right\} n\, h^{p+1} \leq C \max_l \left\{|K_l|\right\} \frac{t_n - t_0}{h}\, h^{p+1} \\ &\leq \widetilde{K} h^p, \end{aligned} \tag{2.10}$$

where $\widetilde{K}$ is a numerical constant. Equation (2.10) demonstrates that, for a p-th order Runge-Kutta method, the global error can be expected to scale as h^p.

In definition 1, the matrix $(a_{i,j})$ is commonly called the *Runge-Kutta matrix*, while b_i and c_i are known as the *weights* and *nodes*, respectively. Since the 1960s, it has been customary to represent Runge-Kutta methods, given by equation (2.6), symbolically, by means of mnemonic devices known as Butcher tableaus (Hairer, Nørsett, and Wanner 1993, p.134). The Butcher tableau for a general s-stage Runge-Kutta method, introduced in definition 1, is presented in table 2.1.

Table 2.1: Butcher tableau representation of a general s-stage Runge-Kutta method.

c_1	$a_{1,1}$	$a_{1,2}$	$\dots$	$a_{1,s}$
c_2	$a_{2,1}$	$a_{2,2}$	$\dots$	$a_{2,s}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	$a_{s,1}$	$a_{s,2}$	$\dots$	$a_{s,s}$
	b_1	b_2	$\dots$	b_s

For explicit Runge-Kutta methods, the Runge-Kutta matrix $(a_{i,j})$ is lower triangular. Similarly, for fully implicit Runge-Kutta methods, the Runge-Kutta matrix is upper triangular. The difference between explicit and implicit methods is outlined in equation (2.4).

During the first half of the twentieth century, a substantial amount of research was conducted in order to develop numerically robust, high-order, explicit Runge-Kutta methods. The idea was that using such methods would mean one could resort to larger time increments h without sacrificing precision in the computational solution. However, the number of stages s grows quicker than linearly as a function of the required order p. It has been proven that, for $p \geq 5$, no explicit Runge-Kutta method of order p with $s = p$ stages exists (Hairer, Nørsett, and Wanner 1993, p.173). This is one of the reasons for the attention shift from the latter half of the 1950s and onwards, towards so-called *embedded* Runge-Kutta methods.

The basic idea of embedded Runge-Kutta methods is that they, aside from the numerical approximation x_{n+1}, yield a second approximation $\widehat{x}_{n+1}$. The difference between the two approximations then yields an estimate of the local error of the less precise result, which can be used for automatic step size control (Hairer, Nørsett, and Wanner 1993, pp.167–168). The trick is to construct two independent, explicit Runge-Kutta methods which both use the *same* function evaluations. This results in practically obtaining the two solutions for the price of one, in terms of computational complexity. The Butcher tableau of an embedded, general, explicit Runge-Kutta method is illustrated in table 2.2.

For embedded methods, the coefficients are tuned such that

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i k_i \tag{2.11a}$$

is of order p, and

$$\widehat{x}_{n+1} = x_n + h \sum_{i=1}^{s} \widehat{b}_i k_i \tag{2.11b}$$

is of order $\widehat{p}$, typically with $\widehat{p} = p \pm 1$. Which of the solutions are used to continue the numerical integration, depends on the integration scheme in question. In the following, the solution which is *not* used to continue the integration, will be referred to as the *interpolant* solution.

Table 2.2: Butcher tableau representation of general, embedded, explicit Runge-Kutta methods.

0					
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
c_s	$a_{s,1}$	$a_{s,2}$	$\dots$	$a_{s,s-1}$	
	b_1	b_2	$\dots$	b_{s-1}	b_s
	$\widehat{b}_1$	$\widehat{b}_2$	$\dots$	$\widehat{b}_{s-1}$	$\widehat{b}_s$

2.1.2 *The Runge-Kutta methods under consideration*

There exists an abundance of Runge-Kutta methods. Many of them are fine-tuned for specific constraints, such as problems of varying degrees of stiffness. It is neither possible nor meaningful to investigate them all in the context of general flow dynamics. For this reason, we consider two classes of explicit Runge-Kutta methods, namely singlestep and adaptive stepsize methods. From both classes, we include four different general-purpose ODE solvers of varying order.

Singlestep methods

The singlestep methods under consideration are the classical, explicit Runge-Kutta methods of orders one through to four, i.e., the *Euler*, *Heun*, *Kutta* and *classical Runge-Kutta* methods. The Euler method is 1^{st}-order accurate, and requires a single function evaluation of the right hand side of the ODE of system (2.1) or (2.2) at each time step. Its Butcher tableau representation can be found in table 2.3. It is the simplest explicit method for numerical integration of ordinary differential equations. The Euler method is often used as a basis to construct more complex methods, such as the Heun method, which is also known as the *improved Euler method* or the *explicit trapezoidal rule*. The Heun method is 2^{nd}-order accurate, and requires two function evaluations at each time step. Its Butcher tableau representation can be found in table 2.4.

Table 2.3: Butcher tableau representation of the explicit Euler method.

0	
	1

Table 2.4: Butcher tableau representation of the explicit Heun method.

0		
1	1	
	$\frac{1}{2}$	$\frac{1}{2}$

The Kutta method is 3^{rd}-order accurate, and requires three function evaluations of the right hand side of the ordinary differential equation (2.1) or (2.2) at each time step. Its Butcher tableau representation can be found in table 2.5. The classical Runge-Kutta method is 4^{th}-order accurate, and perhaps the most well-known and frequently used of the four singlestep schemes discussed in this project. One reason for its popularity is that it is exceptionally stable numerically (of the aforementioned singlestep methods, the classical Runge-Kutta method has the largest numerical stability domain). Another is that, as mentioned previously, for $p \geq 5$, no explicit Runge-Kutta method of order p with $s = p$ stages exist (Hairer, Nørsett, and Wanner 1993, p.173) – in other words, the required number of function evaluations grows at a disproportional rate with the required accuracy order. For systems with right hand sides which are computationally costly to evaluate, this means that one frequently is able to obtain the desired numerical accuracy more effectively by using, for instance, the classical Runge-Kutta method with a finer step length. The Butcher tableau representation of the classical Runge-Kutta method can be found in table 2.6.

Table 2.5: Butcher tableau representation of the explicit Kutta method.

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	-1	2	
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

Table 2.6: Butcher tableau representation of the explicit, classical Runge-Kutta method.

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Adaptive stepsize methods

The adaptive stepsize methods under consideration are the Bogacki-Shampine 3(2) and 5(4) methods, and the Dormand-Prince 5(4) and 8(7) methods. The digit outside of the parentheses indicates the order of the solution which is used to continue the integration, while the digit within the parentheses indicates the order of the interpolant solution. Note that the concept of *order* does not translate directly from singlestep methods, as a direct consequence of the adaptive time step. Although the *local* errors of each integration step scale as per equation (2.7), the bound on the *global* (i.e., observable) error suggested in equation (2.10) is invalid, as the time step is, in principle, different for each integration step. Generally, lower order methods are more suitable than higher order methods for cases where crude approximations of the solution are sufficient. Bogacki and Shampine argue that their methods outperform other methods of the same order (Bogacki and Shampine 1989; Bogacki and Shampine 1996), a notion which, for the 5(4) method, is supported by Hairer, Nørsett, and Wanner (1993, p.194).

Butcher tableau representations of the aforementioned adaptive stepsize methods can be found in tables 2.7–2.10, the latter of which has been typeset in landscape orientation for the reader's convenience. Three of the methods, namely the Bogacki-Shampine 3(2) and 5(4) methods, in addition to the Dormand-Prince 5(4) method, possess the so-called *First Same As Last* property. This means that the last function evaluation of an accepted step is exactly the same as the first function evaluation of the next step. The notions of accepted and rejected integration steps will be elaborated upon in section 3.2.3. The *First Same As Last* property is readily apparent from their Butcher tableaus, where the b coefficients correspond exactly with the last row of the Runge-Kutta matrix. This property reduces the computational cost of a successive step. Moreover, the Bogacki-Shampine 5(4) method yields *two* interpolant solutions. The details on how these were used, will be presented in section 3.2.3.

Table 2.7: Butcher tableau representation of the Bogacki-Shampine 3(2) embedded Runge-Kutta method. The b coefficients correspond to a 3^{rd}-order accurate solution used to continue the integration. The $\widehat{b}$ coefficients correspond to a 2^{nd}-order accurate interpolant, which can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. The *First Same As Last* property is apparent from the fact that the b coefficients correspond exactly to the last row of coefficients in the Runge-Kutta matrix. For reference, see Bogacki and Shampine (1989).

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
	$\frac{7}{24}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$

Table 2.8: Butcher tableau representation of the Bogacki-Shampine 5(4) embedded Runge-Kutta method. The b coefficients correspond to a 5^{th}-order accurate solution used to continue the integration. The two rows of $\widehat{b}$ coefficients correspond to two independent 4^{th}-order accurate interpolants. They can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. The fact that two independent interpolants are included is a part of the reason for which the method nearly behaves like a 6^{th}-order method (Hairer, Nørsett, and Wanner 1993, p.194). The *First Same As Last* property is apparent from the fact that the b coefficients correspond exactly to the last row of coefficients in the Runge-Kutta matrix. For reference, see Bogacki and Shampine (1996).

0								
$\frac{1}{6}$	$\frac{1}{6}$							
$\frac{2}{9}$	$\frac{2}{27}$	$\frac{4}{27}$						
$\frac{3}{7}$	$\frac{183}{1372}$	$\frac{-162}{343}$	$\frac{1053}{1372}$					
$\frac{2}{3}$	$\frac{68}{297}$	$\frac{-4}{11}$	$\frac{42}{143}$	$\frac{1960}{3861}$				
$\frac{3}{4}$	$\frac{597}{22528}$	$\frac{81}{352}$	$\frac{63099}{585728}$	$\frac{58653}{366080}$	$\frac{4617}{20480}$			
1	$\frac{174197}{959244}$	$\frac{-30942}{79937}$	$\frac{8152137}{19744439}$	$\frac{666106}{1039181}$	$\frac{-29421}{29068}$	$\frac{482048}{414219}$		
1	$\frac{587}{8064}$	0	$\frac{4440339}{15491840}$	$\frac{24353}{124800}$	$\frac{387}{44800}$	$\frac{2152}{5985}$	$\frac{7267}{94080}$	
	$\frac{587}{8064}$	0	$\frac{4440339}{15491840}$	$\frac{24353}{124800}$	$\frac{387}{44800}$	$\frac{2152}{5985}$	$\frac{7267}{94080}$	
	$\frac{6059}{80640}$	0	$\frac{8559189}{30983680}$	$\frac{26411}{124800}$	$\frac{-927}{89600}$	$\frac{443}{1197}$	$\frac{7267}{94080}$	
	$\frac{2479}{34992}$	0	$\frac{123}{416}$	$\frac{612941}{3411720}$	$\frac{43}{1440}$	$\frac{2272}{6561}$	$\frac{79937}{1113912}$	$\frac{3293}{556956}$

Table 2.9: Butcher tableau representation of the Dormand-Prince 5(4) embedded Runge-Kutta method. The b coefficients correspond to a 5^{th}-order accurate solution used to continue the integration. The $\widehat{b}$ coefficients correspond to a 4^{th}-order interpolant, which can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. The *First Same As Last* property is apparent from the fact that the b coefficients correspond exactly to the last row of the Runge-Kutta matrix. For reference, see Dormand and Prince (1980).

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$\frac{-56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$\frac{-25360}{2187}$	$\frac{64448}{6561}$	$\frac{-212}{769}$			
1	$\frac{9017}{3168}$	$\frac{-355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$\frac{-5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$\frac{-2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$\frac{-2187}{6784}$	$\frac{11}{84}$	
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$\frac{-92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Table 2.10: Butcher tableau for the Dormand-Prince 8(7) embedded Runge-Kutta method. The b coefficients correspond to an 8^{th}-order accurate solution used to continue the integration. The $\widehat{b}$ coefficients correspond to a 7^{th}-order interpolant, which can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. Note that the provided coefficients are rational approximations, accurate to about 24 significant decimal digits – sufficiently well-resolved for most numerical applications. For reference, see Prince and Dormand (1981).

0												
$\frac{1}{18}$	$\frac{1}{18}$											
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$										
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$									
$\frac{5}{16}$	$\frac{5}{16}$	0	$\frac{-75}{64}$	$\frac{75}{64}$								
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$							
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736358}{692538347}$	$\frac{-28693883}{1125000000}$	$\frac{23124283}{1800000000}$						
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$\frac{-180193667}{1043307555}$					
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$\frac{-433636366}{683701615}$	$\frac{-421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$				
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$\frac{-37695042795}{15268766246}$	$\frac{-309121744}{1061227803}$	$\frac{-12992083}{490766935}$	$\frac{6005943493}{2108947689}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$			
$\frac{1201146811}{1299019798}$	$\frac{-1028468189}{846180014}$	0	0	$\frac{8478235783}{508512851}$	$\frac{1311729495}{1432422823}$	$\frac{-10304129995}{1701304382}$	$\frac{-48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$\frac{-45442868181}{339846796}$	$\frac{3065993473}{5917172653}$		
1	$\frac{185892177}{718116043}$	0	0	$\frac{-3185094517}{667107341}$	$\frac{-477755414}{1098053517}$	$\frac{-703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$\frac{-4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$	
1	$\frac{403863854}{491063109}$	0	0	$\frac{-5068492393}{434740067}$	$\frac{-411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$\frac{-13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$\frac{-160528059}{685178525}$	$\frac{248638103}{1413531060}$	0
	$\frac{1400451}{335480064}$	0	0	0	0	$\frac{-59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{56129285}{797845732}$	$\frac{-1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$\frac{-528747749}{2220607170}$
	$\frac{13451932}{455176623}$	0	0	0	0	$\frac{-808719486}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$\frac{-3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$

2.2 The type of flow systems considered

We consider flow in two-dimensional dynamical systems of the form

$$\dot{\mathbf{x}} = \mathbf{v}(t, \mathbf{x}), \quad \mathbf{x} \in \mathcal{U}, \quad t \in [t_0, t_1], \tag{2.12}$$

i.e., systems defined for the finite time interval $[t_0, t_1]$, on an open, bounded subset $\mathcal{U}$ of $\mathbb{R}^2$. In addition, the velocity field $\mathbf{v}$ is assumed to be smooth in its arguments. Depending on the exact nature of the velocity field $\mathbf{v}$, analytical particle trajectories, that is, analytical solutions of system (2.12), may or may not exist. The flow particles are assumed to be infinitesimal and massless, i.e., non-interacting *tracers* of the overall circulation.

Letting $\mathbf{x}(t; t_0, \mathbf{x}_0)$ denote the trajectory of a tracer in the system defined by equation (2.12), the flow map is defined as

$$\mathbf{F}_{t_0}^{t}(\mathbf{x}_0) = \mathbf{x}(t; t_0, \mathbf{x}_0), \tag{2.13}$$

i.e., the flow map describes the mathematical movement of the tracers from one point in time to another. Generally, the flow map is as smooth as the velocity field $\mathbf{v}$ in system (2.12) (Farazmand and Haller 2012). For sufficiently smooth velocity fields, the right Cauchy-Green strain tensor field is defined as

$$\boldsymbol{C}_{t_0}^{t}(\mathbf{x}_0) = \left(\nabla \mathbf{F}_{t_0}^{t}(\mathbf{x}_0)\right)^{*} \nabla \mathbf{F}_{t_0}^{t}(\mathbf{x}_0), \tag{2.14}$$

where $\nabla \mathbf{F}_{t_0}^{t}$ denotes the Jacobian matrix of the flow map $\mathbf{F}_{t_0}^{t}$, and the asterisk refers to the adjoint operation, which, because the Jacobian $\nabla \mathbf{F}_{t_0}^{t}$ is real-valued, equates to matrix transposition. Component-wise, the Jacobian matrix of a general vector-valued function $\mathbf{f}$ is defined as

$$(\nabla \mathbf{f})_{i,j} = \frac{\partial f_i}{\partial x_j}, \quad \mathbf{f} = \mathbf{f}(\mathbf{x}) = \left(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots\right), \tag{2.15}$$

which, for our two-dimensional flow, reduces to

$$\nabla \mathbf{f} = \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\[2ex] \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{pmatrix}. \tag{2.16}$$

Because the Jacobian of the flow map is invertible, the Cauchy-Green strain tensor $\boldsymbol{C}_{t_0}^{t}$ is symmetric and positive definite (Farazmand and Haller 2012). Thus, it has two real, positive eigenvalues and orthogonal, real eigenvectors. Its eigenvalues λ_i and corresponding unit eigenvectors ξ_i are defined by

$$\begin{gathered} \boldsymbol{C}_{t_0}^{t}(\mathbf{x}_0)\xi_i(\mathbf{x}_0) = \lambda_i \xi_i(\mathbf{x}_0), \quad \|\xi_i(\mathbf{x}_0)\| = 1, \quad i = 1, 2, \\ 0 < \lambda_1(\mathbf{x}_0) \leq \lambda_2(\mathbf{x}_0), \end{gathered} \tag{2.17}$$

where, for the sake of notational transparency, the dependence of λ_i and ξ_i on t_0 and t has been

suppressed. The geometric interpretation of equation (2.17) is that a fluid element undergoes the most stretching along the ξ_2 axis, and the least along the ξ_1 axis. This concept is shown in figure 2.1.

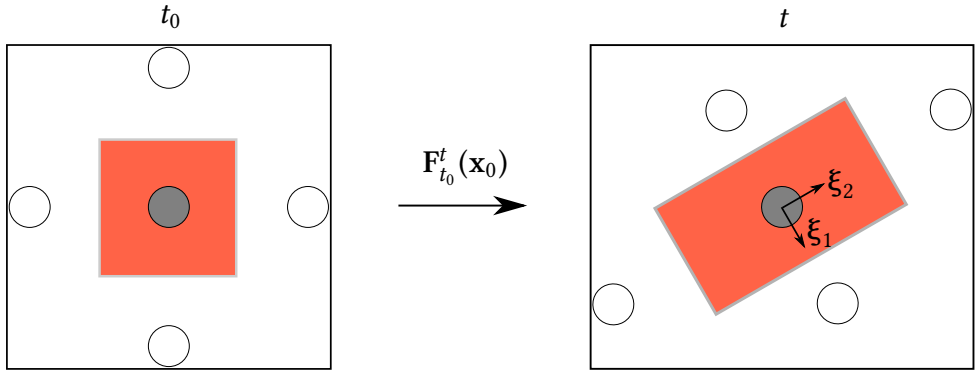


Figure 2.1: Geometric interpretation of the eigenvectors of the Cauchy-Green strain tensor. The central unit cell is stretched and deformed under the flow map $\mathbf{F}_{t_0}^{t}(\mathbf{x}_0)$. The local stretching is the largest in the direction of ξ_2, the eigenvector which corresponds to the largest eigenvalue λ_2 of the Cauchy-Green strain tensor, defined in equation (2.17). Along the ξ_2 and ξ_1 axes, the stretch factors are $\sqrt{\lambda_2}$ and $\sqrt{\lambda_1}$, respectively.

Because the stretch factors along the ξ_1 and ξ_2 axes are given by the square root of the corresponding eigenvalues, for incompressible flow, the eigenvalues satisfy

$$\lambda_1(\mathbf{x}_0)\lambda_2(\mathbf{x}_0) = 1 \quad \forall \ \mathbf{x}_0 \in \mathcal{U} \tag{2.18}$$

where, in the context of tracer advection, incompressibility is equivalent to the velocity field $\mathbf{v}$ being divergence-free (i.e., $\nabla \cdot \mathbf{v} \equiv 0$).

2.3 Definition of Lagrangian coherent structures for two-dimensional flows

Lagrangian coherent structures (henceforth abbreviated to LCSs) can be described as time-evolving surfaces which shape coherent trajectory patterns in dynamical systems, defined over a finite time interval (Haller 2010). There are three main types of LCSs, namely *elliptic*, *hyperbolic* and *parabolic*. Roughly speaking, parabolic structures outline cores of jet-like trajectories, elliptic structures describe vortex boundaries, whereas hyperbolic structures illustrate overall attractive or repelling manifolds. As such, hyperbolic LCSs practically act as organizing centers of observable tracer patterns (Onu, Huhn, and Haller 2015). Because hyperbolic LCSs provide the most readily applicable insight in terms of forecasting flow in e.g. oceanic currents, such structures have been the focus of this project.

2.3.1 *Hyperbolic LCSs*

The identifacion of LCSs for reliable forecasting requires sufficiency and necessity conditions, supported by mathematical theorems. Haller (2010) derived a variational LCS theory based on the Cauchy-Green strain tensor, defined by equation (2.14), from which the aforementioned conditions follow. The immediately relevant parts of Haller's theory are quoted in definitions 3–6 (Haller 2010).

Definition 3. A *normally repelling material line* over the time interval $[t_0, t_0 + T]$ is a compact material line segment $\mathcal{M}(t)$ which is overall repelling, and on which the normal repulsion rate is greater than the tangential repulsion rate.

A *material line* is a smooth curve $\mathcal{M}(t_0)$ at time t_0, which is advected by the flow map, given by equation (2.13), into the dynamic material line $\mathcal{M}(t) = \mathbf{F}_{t_0}^{t}\mathcal{M}(t_0)$. The required *compactness* of the material line segment signifies that, in some sense, it must be topologically well-behaved. That the material line is *overall repelling* means that nearby trajectories are repelled from, rather than attracted to, the material line. Lastly, requiring that the *normal repulsion rate* is greater than the *tangential repulsion rate* means that nearby trajectories are in fact driven away from the material line, rather than being stretched *along with* it due to shear stress.

Definition 4. A *repelling LCS* over the time interval $[t_0, t_0 + T]$ is a normally repelling material line $\mathcal{M}(t_0)$ whose normal repulsion admits a pointwise non-degenerate maximum relative to any nearby material line $\widehat{\mathcal{M}}(t_0)$.

Definition 5. An *attracting LCS* over the time interval $[t_0, t_0 + T]$ is defined as a repelling LCS over the *backward* time interval $[t_0 + T, t_0]$.

Definition 6. A *hyperbolic* LCS over the time interval $[t_0, t_0 + T]$ is a *repelling* or *attracting* LCS over the same time interval.

Note that the above definitions associate LCSs with the time interval I over which the dynamical system under consideration is known, or, at the very least, where information regarding the behaviour of tracers, is sought. Generally, LCSs obtained over a time interval I do not necessarily exist over different time intervals (Farazmand and Haller 2012).

For two-dimensional flow, the above definitions can be summarized as a set of mathematical existence criteria, based on the Cauchy-Green strain tensor, (Haller 2010; Farazmand and Haller 2011). These are given in theorem 1 (Farazmand and Haller 2012):

Theorem 1 (Sufficient and necessary conditions for LCSs in two-dimensional flows)**.** *Consider a compact material line* $\mathcal{M}(t) \subset \mathcal{U}$ *evolving over the time interval* $[t_0, t_0 + T]$*. Then* $\mathcal{M}(t)$ *is a repelling LCS over* $[t_0, t_0 + T]$ *if and only if all the following hold for all initial conditions* $\mathbf{x}_0 \in \mathcal{M}(t_0)$*:*

$$\lambda_1(\mathbf{x}_0) \neq \lambda_2(\mathbf{x}_0) > 1, \tag{2.19a}$$

$$\left\langle \xi_2(\mathbf{x}_0), \boldsymbol{H}_{\lambda_2}(\mathbf{x}_0)\xi_2(\mathbf{x}_0) \right\rangle < 0, \tag{2.19b}$$

$$\xi_2(\mathbf{x}_0) \perp \mathcal{M}(t_0), \tag{2.19c}$$

$$\left\langle \nabla\lambda_2(\mathbf{x}_0), \xi_2(\mathbf{x}_0) \right\rangle = 0. \tag{2.19d}$$

In theorem 1, $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product, and $\boldsymbol{H}_{\lambda_2}$ denotes the Hessian matrix of the set of largest eigenvalues of the Cauchy-Green strain tensor. Component-wise, the Hessian matrix of a general, smooth, scalar-valued function f is defined as

$$\left(\boldsymbol{H}_f\right)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}, \tag{2.20}$$

which, for our two-dimensional flow, reduces to

$$\boldsymbol{H}_f = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{pmatrix}. \tag{2.21}$$

Condition (2.19a) ensures that the normal repulsion rate is larger than the tangential stretch due to shear strain along the LCS, as per definition 3. Conditions (2.19c) and (2.19d) suffice to ensure that the normal repulsion rate attains a local extremum along the LCS, relative to all nearby material lines. Lastly, condition (2.19b) forces this to be a strict local apex.

2.4 FTLE fields as predictors for LCSs

Finite-time Lyapunov exponent (hereafter abbreviated to FTLE) fields provide a measure of the extent to which particles which start out close to each other, are separated in a given time interval. Mathematically, Lyapunov exponents quantify the asymptotic divergence or convergence of trajectories of tracers which start out infinitesimally close to each other (Strogatz 2014, pp.328–330). The FTLE field is intrinsically linked to the Cauchy-Green strain tensor. When a two-dimensional system is evolved from time t_0 to $t_0 + T$, the FTLE field is defined as

$$\sigma(\mathbf{x}_0) = \frac{\ln\left(\lambda_2(\mathbf{x}_0)\right)}{2|T|}, \tag{2.22}$$

where $\lambda_2(\mathbf{x}_0)$ is the largest eigenvalue of the Cauchy-Green strain tensor. The absolute value of the integration time T is taken because one generally can consider the evolution of a system in either temporal direction. Analogously to definition 5, the attraction of nearby trajectories can be estimated by FTLEs found by means of integration backwards in time.

Shadden, Lekien, and Marsden (2005) in fact *define* hyperbolic LCSs as ridges of the FTLE field. By *ridges*, they mean gradient lines which are orthogonal to the direction of minimum curvature. Haller (2010) showed, by means of explicit examples, that the sole use of the FTLE field for LCS detection is prone to both false positives and false negatives, even for conceptually simple flows. Furthermore, the ridges of the FTLE field are generally less well-resolved than the LCSs obtained from Haller's variational formalism. For these reasons, the FTLE field can generally be used as a first approximation in terms of where one can reasonably expect LCSs to be found, but it does not represent the blueprint for the actual LCSs of a system in general.

3 Method

In order to investigate how LCS identification by means of the variational approach presented in section 2.3.1 depends on the choice of numerical integration method, outlined in section 2.1, a system which has been studied extensively in the literature, was chosen. The system, an unsteady double gyre, has been used frequently as a test case for locating LCSs from different indicators (Farazmand and Haller 2012; Shadden, Lekien, and Marsden 2005). As a result, the LCSs the system exhibit are well documented.

3.1 The double gyre model

The double gyre model consists of a pair of counter-rotating vortices, with a time-periodic perturbation. The perturbation can be interpreted as a wall which oscillates periodically, causing the vortices to contract and expand. In terms of the Cartesian coordinate vector $\mathbf{x} = (x, y)$, the system can be expressed mathematically as

$$\dot{\mathbf{x}} = \mathbf{v}(t, \mathbf{x}) = \pi A \begin{pmatrix} -\sin\left(\pi f(t,x)\right)\cos(\pi y) \\ \cos\left(\pi f(t,x)\right)\sin(\pi y)\dfrac{\partial f(t,x)}{\partial x} \end{pmatrix} \tag{3.1}$$

where

$$\begin{aligned} f(t,x) &= a(t)x^2 + b(t)x \\ a(t) &= \epsilon \sin(\omega t) \\ b(t) &= 1 - 2\epsilon \sin(\omega t) \end{aligned} \tag{3.2}$$

and the parameters A, ϵ and ω dictate the nature of the flow pattern. The domain of interest, $\mathcal{U} = [0,\ 2] \times [0,\ 1]$, is characterized by the orthogonal component of the velocity field being zero at its boundaries. Thus, any trajectory which starts out within $\mathcal{U}$, remains there for all time. The parameter values

$$\begin{aligned} A &= 0.1 \\ \epsilon &= 0.1 \\ \omega &= \frac{2\pi}{10} \end{aligned} \tag{3.3}$$

were used, as has been common in literature, in particular in the articles by Farazmand and Haller (2012) and Shadden, Lekien, and Marsden (2005). The starting time $t_0 = 0$ and the integration time $T = 20$ forces the velocity field to undergo two periods of motion, as $\omega = 2\pi/10$, per equation (3.3).

Note that the velocity field $\mathbf{v}(t, \mathbf{x})$ in equation (3.1) can be expressed in terms of a scalar stream function:

$$\psi(t, \mathbf{x}) = A \sin\Big(\pi f(t, x)\Big) \sin(\pi y)$$
$$\mathbf{v}(t, \mathbf{x}) = \begin{pmatrix} -\dfrac{\partial \psi}{\partial y} \\ \\ \dfrac{\partial \psi}{\partial x} \end{pmatrix} \tag{3.4}$$

which means that the velocity field is divergence-free by construction:

$$\nabla \cdot \mathbf{v}(t, \mathbf{x}) = -\frac{\partial^2 \psi}{\partial x \partial y} + \frac{\partial^2 \psi}{\partial y \partial x} = 0, \tag{3.5}$$

where the latter equality follows from the smoothness of the stream function. In particular, since it is 'infinitely' continuously differentiable, partial differentiation of *any* order of the stream function is commutative (Adams and Essex 2010, p.689). Moreover, this means that we expect the property given in equation (2.18), that is, the product of the eigenvalues of the Cauchy-Green strain tensor to be unity, to hold for the double gyre flow.

3.2 Advecting a set of initial conditions

The variational model is based upon the advection of non-interacting tracers, which is described in section 2.2, by the velocity field defined in equation (3.1). To our knowledge, the system has no analytical solution for the tracer trajectories. Thus, it must be solved numerically, by means of some numerical integration method, e.g. a Runge-Kutta method, a family of numerical ODE methods which is outlined in section 2.1.1. With the main focus of this project being the dependence on LCSs on the chosen integration method, the advection was performed using all of the numerical integrators introduced in section 2.1.2.

3.2.1 *Generating a set of initial conditions*

The computational domain $\mathcal{U} = [0,\ 2] \times [0,\ 1]$ was discretized by a set of equidistant *principal* tracers, with 1000×500 grid points, effectively creating a nearly uniform grid of approximate spacing $\Delta x \simeq \Delta y \simeq 0.002$. Tracers were placed on and within the domain boundaries of $\mathcal{U}$. The grid was extended outside of $\mathcal{U}$, with an additional two rows or columns appended to all of the domain edges, with the same grid spacing as the *principal* grid. This was done in order to ensure that the dynamics at the domain boundaries were included in the analysis to follow. The velocity field outside of $\mathcal{U}$ was also defined by equation (3.1). The *extended* grid thus had a total of 1004×504 grid points. The construction of the grid is illustrated in figure 3.1.

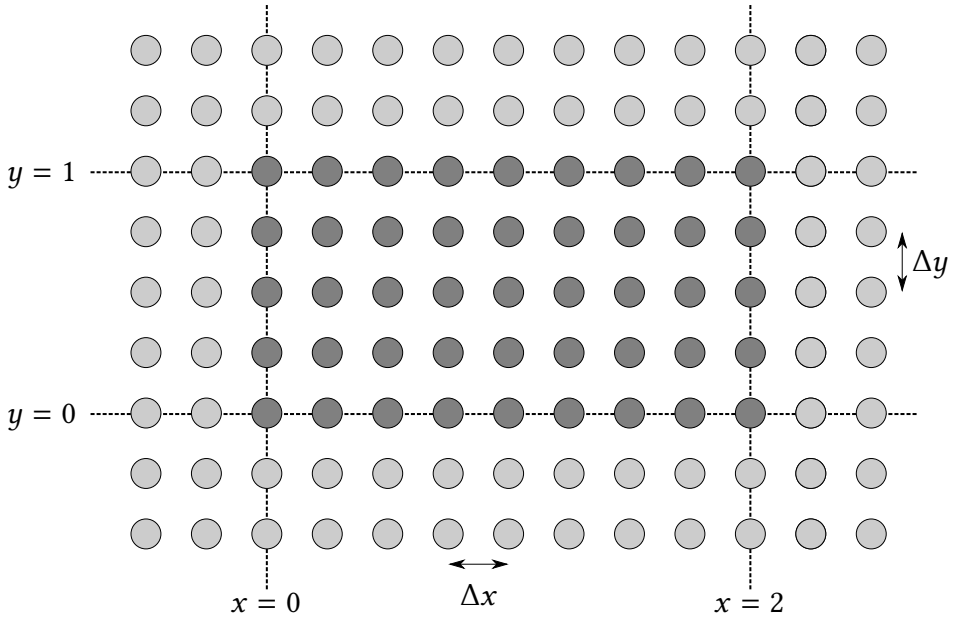


Figure 3.1: Illustration of the set of initial conditions. Dark grey blobs signify the principal tracers, i.e., the tracers which discretize the computational domain $[0\ \ 2] \times [0\ \ 1]$. These were equidistantly spaced in either direction, with twice as many points in the x- direction as the y-direction, in order to generate an approximately equidistant grid. Together with the principal grid, the light grey blobs, i.e., tracers originating outside of the computational domain, constitute the extended grid. This extension was made in order to properly encapsulate the dynamics at the domain boundaries, in the analysis to follow.

In order to increase the precision of the Cauchy-Green strain tensor, it is necessary to increase the accuracy with which one computes the Jacobian of the flow map, as their accuracies are intrinsically linked; which follows from equation (2.14). This was done by advecting a set of four auxiliary tracers surrounding each *main* tracer, that is, tracers originating at the principal or extended grid points. To each tracer point $\mathbf{x}_{i,j} = (x_i, y_j)$, neighboring points defined as

$$\begin{aligned} \mathbf{x}^r_{i,j} &= (x_i + \delta x, y_j), \quad & \mathbf{x}^l_{i,j} &= (x_i - \delta x, y_j) \\ \mathbf{x}^u_{i,j} &= (x_i, y_j + \delta y), \quad & \mathbf{x}^d_{i,j} &= (x_i, y_j - \delta y) \end{aligned} \tag{3.6}$$

were assigned, where δx and δy are increments smaller than the grid spacings $\Delta x \simeq \Delta y$. Even though this effectively means that five times as many tracers have to be advected, the resulting accuracy in computing the Jacobian of the flow map, by means of the auxiliary tracers, is determined by the independent variables δx and δy. This, in principle, allows for much higher precision than what would be obtained by simply advecting five times as many *equidistant* tracers. The concept of the auxiliary tracers is illustrated in figure 3.2.

Because of the limited number of decimal digits which can be accuractely represented by floating-point numbers, however, there is a strict lower limit to which it makes sense to lower the values of δx and δy. In particular, because spatial extent of the computational domain is of order 1, so too will all of the tracer coordinates be. Thus, the so-called machine epsilon of double-precision floating-point numbers — which, somewhat simplified, can be described as the

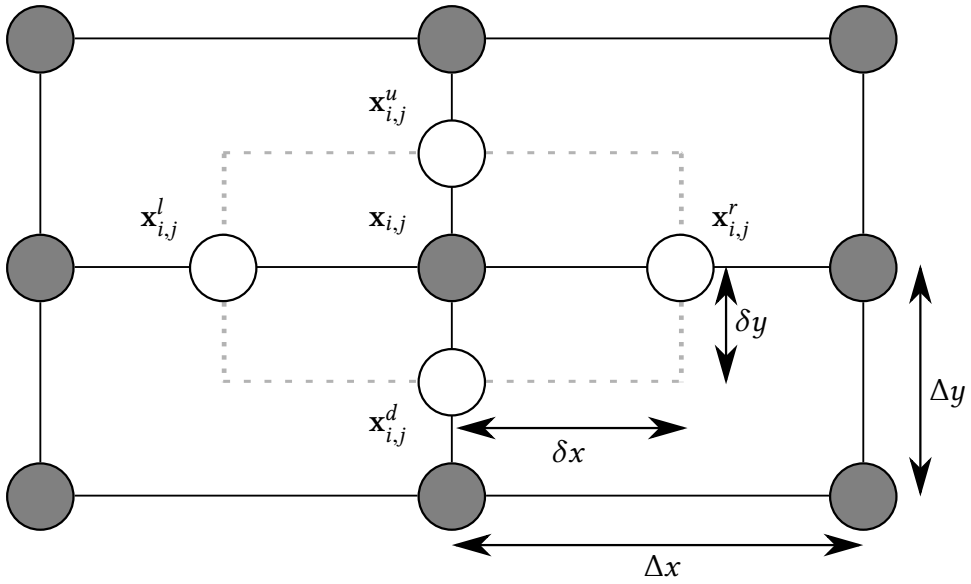


Figure 3.2: Illustration of the concept of auxiliary tracers,. These were introduced to increase the accuracy with which the Jacobian of the flow map, and by extension the Cauchy-Green strain tensor field (given by equation (2.14)), was computed. Grey blobs represent the original tracers, whereas white blobs represent the auxiliary ones.

smallest positive number ε for which $1 + \varepsilon$ is not rounded down to 1 — is the dominant source of floating-point errors with regards to the flow map, moreso than the inherent precision, that is, the smallest positive number which can be *represented* as a double-precision floating-point number. Per the IEEE standard for floating-point arithmetic, the double-precision machine epsilon is of order 10^{-16}, whereas the smallest positive number which can be represented in double precision is of order 10^{-308} (Institute of Electrical and Electronics Engineers 2008).

So, because of the scale of the numerical domain, we cannot reasonably expect to accurately represent the flow maps beyond 15 decimal digits. In order to approximate the Jacobian of the flow map, finite differencing was applied, which will be described in greater detail in section 3.3. Note, however, that taking the finite difference between the endpoint coordinates of neighboring tracers is generally expected to yield small decimal numbers, because the velocity field, given in equation (3.1), is reasonably well-behaved. Say, for instance, that the difference is of order 10^{-10}. Because, as previously mentioned, we can only really expect to accurately resolve the flow map to the 15^{th} decimal digit, this leaves only 5 significant digits. With the parameter choices specified in equation (3.3), the velocity field leads most tracers which are intially close to follow very similar trajectories, often ending up with a separation distance comparable to the initial offset. For this reason, the auxiliary grid spacing $\delta x = \delta y = 10^{-5}$ was chosen — two orders of magnitude less than the original spacing — ensuring that the derivatives in the Jacobian are far more well-resolved than for the main tracers alone, while also leaving up to 10 significant decimal digits for which the difference in the final positions of the auxiliary tracers can be resolved.

3.2.2 *On the choice of numerical step lengths and tolerance levels*

For the fixed stepsize integrators, step lengths of 10^{-1} through to 10^{-5} were used. The reason even smaller step lengths were not considered is that the accumulated numerical round-off errors, mentioned in section 3.2.1, for the advection of tracers by means of the highest order methods, that is, Kutta's method and the classical Runge-Kutta method (cf. tables 2.5 and 2.6), became large enough to counteract the inherent accuracies of the methods, for time steps smaller than 10^{-4} and 10^{-3}, respectively. Further details will be presented in chapter 4. We expect a similar development for both the Euler and Heun methods (cf. tables 2.3 and 2.4), albeit for even smaller time steps, due to these methods being of lower order accuracy. As previously mentioned, the concept of *order* does not translate directly from the singlestep integrators to the adaptive stepsize methods, by virtue of the dynamical step length. The step length adjustment procedure considered here, which will be outlined in detail in section 3.2.3, involves the use of numerical tolerances regarding the difference between the interpolant and locally extrapolated solutions. Empirical tests indicate that for both of the Bogacki-Shampine integrators, as well as for the Dormand-Prince 5(4) integrator, the accumulated floating-point errors caught up to the required tolerance level at some point between the levels 10^{-10} and 10^{-11}, while the Dormand-Prince 8(7) integrator held its ground until a tolerance level of about 10^{-13}. For this reason, tolerance levels of 10^{-1} through to 10^{-10} were used for the adaptive stepsize integrators. Furthermore, as to our knowledge, no analytical solution exists for the double gyre system, a numerical solution is needed as the reference. Following the discussion above, the solution obtained via the Dormand-Prince 8(7) integrator with a numerical tolerance level of 10^{-12} was used for this purpose.

With the addition of the aforementioned auxiliary tracers, the total number of tracers which were advected became of order 2.5 million. In order to accelerate the computational process, the advection was parallellized by means of MPI and run on NTNU's supercomputer, Vilje. The choice of MPI over alternative multiprocessing tools was mainly motivated by MPI facilitating access to multiple nodes within the Vilje cluster. Because the tracers are independent, the parallelization consisted of distributing an approximately even amount of tracers across all ranks, whereupon each rank advected its allocated tracers. In the end, all of the tracer end positions was collected by the designated main process (that is, rank = 0). The associated speedup was crucial for this project — for example, advection using the Euler method and a time step of 10^{-5} required in excess of 1000 CPU hours; an insurmountable feat for most computers, including the author's own personal laptop.

3.2.3 *On the implementation of embedded Runge-Kutta methods*

In order to implement automatic step size control, the procedure suggested by Hairer, Nørsett, and Wanner (1993, pp.167–168) was followed closely. A starting step size of $h = 0.1$ was used throughout. For the first solution step, the embedded integration method, as described in section 2.1.1 yields the two approximations x_1 and $\widehat{x}_1$, from which the difference $x_1 - \widehat{x}_1$ can be used as an estimate of the error of the less precise result. The idea is to enforce the error of the numerical solution to satisfy componentwise:

$$|x_{1,i} - \widehat{x}_{1,i}| \leq \mathrm{sc}_i, \qquad \mathrm{sc}_i = \mathrm{Atol}_i + \max\left(|x_{0,i}|, |x_{1,i}|\right) \cdot \mathrm{Rtol}_i \tag{3.7}$$

where Atol_i and Rtol_i are the desired absolute and relative numerical tolerances, prescribed by the user. For this project, Atol_i was always set equal to Rtol_i.

As a measure of the numerical error,

$$\mathrm{err} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_{1,i} - \widehat{x}_{1,i}}{\mathrm{sc}_i}\right)^2} \tag{3.8}$$

is used. Then, err is compared to 1 in order to find an optimal step size. From definition 2, it follows that $\mathrm{err} \approx Kh^{q+1}$, where $q = \min(p, \widehat{p})$. With $1 \approx Kh_{\mathrm{opt}}^{q+1}$, one finds the optimal step size

$$h_{\mathrm{opt}} = h \cdot \left(\frac{1}{\mathrm{err}}\right)^{\frac{1}{q+1}}. \tag{3.9}$$

If $\mathrm{err} \leq 1$, the solution step is accepted, the time variable t is increased by h, and the step length is modified according to equations (3.9) and (3.10). Which of the two approximations x_{n+1} or $\widehat{x}_{n+1}$ are used to continue the integration varies depending on the embedded method in question. Continuing the integration with the higher order result is commonly referred to as *local extrapolation*. If $\mathrm{err} > 1$, the solution step is rejected, the time variable t remains unaltered, and the step length is decreased before another attempted step. The procedure for updating the step length can be summarized as follows:

$$h_{\mathrm{new}} = \begin{cases} \min(\mathrm{fac}_{\max} \cdot h, \mathrm{fac} \cdot h_{\mathrm{opt}}), & \text{if the solution step is accepted} \\ \mathrm{fac} \cdot h_{\mathrm{opt}}, & \text{if the solution step is rejected} \end{cases} \tag{3.10}$$

where fac and $\mathrm{fac}_{\max}$ are numerical safety factors, intended to prevent increasing the step size *too* much. For this project, $\mathrm{fac} = 0.8$ and $\mathrm{fac}_{\max} = 2.0$ were used throughout.

All of the embedded methods considered here are tuned in order to minimize the error of the higher order result; accordingly, local extrapolation was used throughout. Regarding the Bogacki-Shampine 5(4) method, which yields *two* independent 4^{th}-order interpolant solutions, the procedure was slightly altered. First, the error of the 5^{th}-order solution with regards to the interpolant corresponding to the first row of $\widehat{b}$-coefficients in table 2.8 was computed. If this error was larger than unity, the attempted step was rejected, and the same error was used in

order to update the time step, per equations (3.9) and (3.10). In the opposite case, a new error estimate of the 5^{th}-order solution was computed relative to the interpolant corresponding to the second row of $\widehat{b}$-coefficients in table 2.8. If this was larger than unity, the attempted step was rejected, whereas the step was accepted if the error was smaller than or equal to unity. In either case, this *second* error estimate was used in order to update the time step.

3.3 Calculating the Cauchy-Green strain tensor

Making use of the auxiliary tracer points, as outlined in section 3.2.1, the Jacobian of the flow map was approximated by means of centered differences as

$$\nabla \mathbf{F}_{t_0}^{t}(\mathbf{x}_{i,j}) \approx \left(\frac{\mathbf{F}_{t_0}^{t}(\mathbf{x}_{i,j}^{r}) - \mathbf{F}_{t_0}^{t}(\mathbf{x}_{i,j}^{l})}{2\delta x}, \quad \frac{\mathbf{F}_{t_0}^{t}(\mathbf{x}_{i,j}^{u}) - \mathbf{F}_{t_0}^{t}(\mathbf{x}_{i,j}^{d})}{2\delta y} \right). \tag{3.11}$$

The Cauchy-Green strain tensor was then calculated as per equation (2.14). Farazmand and Haller (2012) bring up a point originally raised by Lekien and Ross (2010), that using only the auxiliary tracers in the calculation of the eigenvalues and -vectors of the Cauchy-Green strain tensor is not sufficient for measuring the amount of local repulsion or attraction of the material lines. Their justification is that the set of auxiliary tracers around a main tracer $\mathbf{x}_{i,j}$ tends to stay on the same side of an LCS. This is due to the small initial separations δx and δy, resulting in the auxiliary tracers undergoing less stretching than the main tracers which lay on opposite sides of repelling LCSs. Thus, the magnitudes of eigen*values* stemming from finite differencing applied to the auxiliary tracers are generally underestimated. On the other hand, the eigen*vectors* are computed more accurately than they would be by use of the main tracers alone.

Because of the tendency of finite differencing applied to the auxiliary tracers to underestimate the magnitudes of the eigenvalues, an analogous approximation of the Jacobian of the flow map was also made by means of the main tracers, as

$$\widetilde{\nabla \mathbf{F}}_{t_0}^{t}(\mathbf{x}_{i,j}) \approx \left(\frac{\mathbf{F}_{t_0}^{t}(\mathbf{x}_{i+1,j}) - \mathbf{F}_{t_0}^{t}(\mathbf{x}_{i-1,j})}{2\Delta x}, \quad \frac{\mathbf{F}_{t_0}^{t}(\mathbf{x}_{i,j+1}) - \mathbf{F}_{t_0}^{t}(\mathbf{x}_{i,j-1})}{2\Delta y} \right). \tag{3.12}$$

The extended grid, as outlined in section 3.2.1, allowed for a consistent centered differencing approximation for all the main tracers. This also includes the first set of extended rows and columns, where the latter play an important role in the analysis to follow — more on that in section 3.4. The numerical approximation of the eigen*values* of the Cauchy-Green strain tensor were thus calculated by means of the main tracers, using equation (3.12), while the approximation of the corresponding eigen*vectors* were calculated by means of the auxiliary tracers, that is, using equation (3.11).

3.4 Identifying LCS candidates numerically

Farazmand and Haller (2012) found some of the conditions of theorem 1 to be numerically sensitive. Therefore, they suggest a reformulated set of conditions which make for a more robust numerical implementation:

$$\lambda_1(\mathbf{x}_0) \neq \lambda_2(\mathbf{x}_0) > 1, \tag{3.13a}$$

$$\left\langle \xi_2(\mathbf{x}_0), \boldsymbol{H}_{\lambda_2}(\mathbf{x}_0)\xi_2(\mathbf{x}_0) \right\rangle \leq 0, \tag{3.13b}$$

$$\xi_1(\mathbf{x}_0) \parallel \mathcal{M}(t_0), \tag{3.13c}$$

$$\begin{aligned} &\overline{\lambda}_2, \text{ the average of } \lambda_2 \text{ over a curve } \gamma, \text{ is maximal on } \mathcal{M}(t_0) \\ &\quad \text{among all nearby curves } \gamma \text{ satisfying } \gamma \parallel \xi_1(\mathbf{x}_0). \end{aligned} \tag{3.13d}$$

The relaxation of condition (2.19b) in theorem 1 from strict inequality to condition (3.13b), also allowing equality, means that LCSs may have finite thickness. However, the set of criteria given in equation (3.13) enforce that all LCSs have uniquely defined local orientations. Conditions (2.19c) and (3.13c) are equivalent, due to the orthogonality of the eigenvectors $\xi_1(\mathbf{x}_0)$ and $\xi_2(\mathbf{x}_0)$, although the form (3.13c) turns out to be more advantageous for use in computations. For details regarding the numerical implementation of conditions (3.13c) and (3.13d), see section 3.4.1.

Because of the extension of the main computational grid, as outlined in section 3.2.1, the Cauchy-Green strain tensor could be calculated for the innermost of the padded rows and columns by means of the same centered difference methods as described in section 3.3. Thus, a similar centered differencing approach was used in order to approximate the Hessian matrices of the set of eigenvalues $\lambda_2(\mathbf{x}_0)$ for the main tracers in the principal grid, that is, the main tracers which originate from within (or on the borders of) the computational domain $\mathcal{U} = [0,\ 2]\times[0,\ 1]$.

3.4.1 *A framework for computing smooth strainlines*

Per condition (3.13c), hyperbolic LCSs are composed of material curves tangent to the $\xi_1(\mathbf{x}_0)$ vector field, i.e., the eigenvector field associated with the smaller eigenvalue field $\lambda_1(\mathbf{x}_0)$ of the Cauchy-Green strain tensor field $\boldsymbol{C}_{t_0}^{t}(\mathbf{x}_0)$. Traditionally, curves which are everywhere tangent to the eigenvector field corresponding to the largest eigenvalues of a two-dimensional tensor field have been called tensor lines (Farazmand and Haller 2012). Thus, in order to distinguish between the tensor lines tangent to ξ_1 and those tangent to ξ_2, the tensor lines tangent to the ξ_1-field will be referred to as *strainlines* in the following; a term coined by Farazmand and Haller. Aside from points within $\mathcal{U}$ which exhibit orientational discontinuities in both eigenvector fields, strainlines can be computed as smooth trajectories of the ordinary differential equation

$$\mathbf{r}' = \xi_1(\mathbf{r}), \quad \mathbf{r} \in \mathcal{U}, \quad \|\xi_1(\mathbf{r})\| = 1, \tag{3.14}$$

where, because the eigenvectors ξ_1 are stationary, the numerical integration is spatial, in principle. However, in order to conform with the established conventions with regards to the numerical integration of ODEs, we consider pseudotime — that is, a ficticious time-like coordinate s is introduced, where advancing s by a numerical integration step Δ corresponds to

a spatial translation by the same length, because the eigenvectors ξ_1 are normalized to unit length.

As pointed out by Onu, Huhn, and Haller (2015), the orientational discontinuities of the ξ_1-field are removable, through careful monitoring and local reorientation. This process is illustrated in figure 3.3, and can be described in terms of three steps. First, the four nearest neighboring grid points to $\mathbf{r}$ are identified. Then, orientational discontinuities inbetween the grid elements are found by inspecting the inner product of the ξ_1 vectors of adjacent grid points. Rotations exceeding 90° between pairs of neighboring vectors are labelled as orientational discontinuities. These are corrected prior to linear interpolation by flipping the corresponding vectors by 180°. In the end, linear interpolation is used within the grid element whose corners are defined by the four nearest neighbors to $\mathbf{r}$.

Furthermore, should the ξ_1-vector obtained from the local special-purpose linear interpolation outlined above prove to be rotated by more than 90° relative to the interpolated ξ_1-vector at the previous point of the strainline, it would be flipped 180°, to facilitate smooth trajectories through oriental discontinuities. The entire process of this special-purpose local linear interpolation method is outlined in figure 3.3.

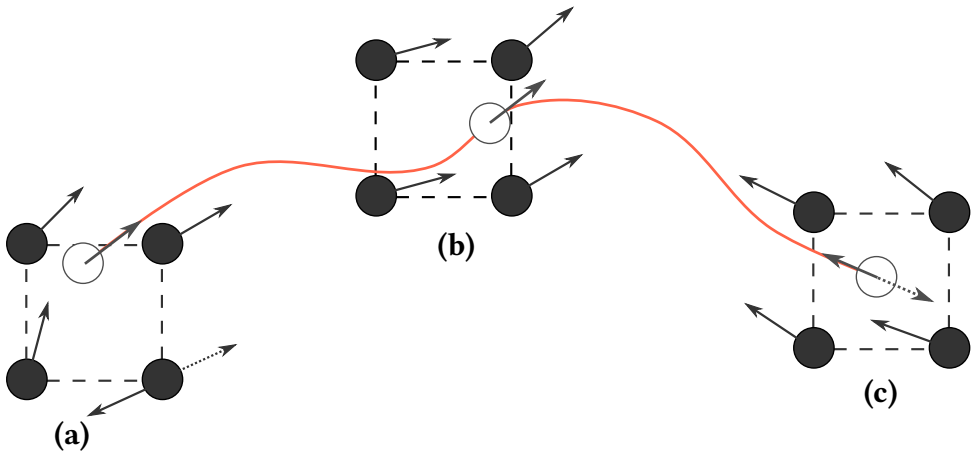


Figure 3.3: Illustration of the special-purpose linear interpolation used to compute trajectories of the ξ_1 eigenvector field. At point **(a)**, there is an orientational discontinuity at the lower right grid point, which is corrected by rotating the corresponding vector by 180° prior to linear interpolation. At point **(b)**, there is no orientational discontinuity. Lastly, at point **(c)**, the interpolated vector must be flipped due to the overall orientation of the trajectory.

So, in order to compute globally smooth strainlines, equation (3.14) is altered in the following way:

$$\mathbf{r}'(s) = \mathbf{f}\Big(\mathbf{r}(s)\Big), \tag{3.15}$$

where $\mathbf{f}$ denotes the special-purpose local linear interpolation of the ξ_1 field, as outlined above and in figure 3.3.

3.4.2 *Extracting hyperbolic LCSs from strainlines*

If a material line $\mathcal{M}(t_0)$ lies within a strainline, it automatically fulfills condition (3.13c), as a trajectory of the ξ_1-field is, per definition, always parallel to it. The segments of the strainlines on which the remaining conditions (3.13a), (3.13b) and (3.13d) are satisfied comprises the set of hyperbolic LCSs in the flow over time interval $[t_0, t_0 + T]$. Farazmand and Haller (2012) suggest that in order to identify this set of LCSs, one should start by identifying the subdomain $\mathcal{U}_0 \subset \mathcal{U}$ on which the conditions (3.13a) and (3.13b) are satisfied, and then integrate the system given by equation (3.15) from initial conditions within $\mathcal{U}_0$ to construct strainlines. Generally, the integration proceeds until each strainline reaches the domain boundaries of $\mathcal{U}$, or reaches a degenerate point (see below) of the original ξ_1 vector field.

Using all of the points in the $\mathcal{U}_0$ domain would inevitably involve computing a lot of strainlines several times over. In order to reduce the number of redundant calculations, the set of considered strain initial conditions were reduced, with the approach suggested by Farazmand and Haller (2012). In particular, the set $\mathcal{U}_0$ was reduced to its intersections with four horizontal and four vertical lines. The set $\mathcal{U}_0$ for the double gyre system, as well as the aforementioned intersections, are illustrated in figure 3.4. The reduced set of strain initial conditions consists of 1470 grid points, which is two orders of magnitude less than the total number of points in $\mathcal{U}_0$.

The degenerate points of the ξ_1 vector field is, as the name implies, the set of points in $\mathcal{U}$ for which the eigenvalues $\lambda_1(\mathbf{x}_0)$ and $\lambda_2(\mathbf{x}_0)$ are equal, leaving the strain eigenvector field ξ_1 undefined. As a computational measure of this degeneracy, the scalar field defined as

$$\alpha(\mathbf{x}_0) = \left(\frac{\lambda_2(\mathbf{x}_0) - \lambda_1(\mathbf{x}_0)}{\lambda_2(\mathbf{x}_0) + \lambda_1(\mathbf{x}_0)} \right)^2 \tag{3.16}$$

was used, inspired by Farazmand and Haller (2012). For points $\mathbf{x}$ which did not coincide with the grid points $\mathbf{x}_{i,j}$, the values $\lambda_1(\mathbf{x})$ and $\lambda_2(\mathbf{x})$ were found by means of regular linear interpolation. Wherever the value of $\alpha(\mathbf{x})$ decreased below a predefined threshold of 10^{-6}, the point $\mathbf{x}$ was flagged as degenerate, thus stopping the strainline integration.

Generally, one expects the selected strain initial conditions, shown in figure 3.4, to be located somewhere in the middle of a strainline. Hence, two strainline segments were computed and merged for each initial condition; one moving forwards in pseudotime, and one moving backwards, i.e., using the 180° rotated ξ_1 vector field in the system given by equation (3.15). This ensured that no strainline was ended prematurely, purely as a consequence of the reduced number of strain initial conditions. Furthermore, the classical Runge-Kutta method with an integration step length $\Delta = 10^{-3}$ was used in order to compute all of the strainlines, regardless of the integration scheme used in the advection of the tracers, described in section 3.2. The classical Runge-Kutta method was applied universally for consistency, and chosen for its simplicity and relative accuracy. Moreover, this project is centered around the dependence of the choice of integration method in the tracer advection. The step length was made to be half of the main grid spacing, to ensure that the micro-scale strain behaviour would be encapsulated in the resulting strainlines, while maintaining reasonable computation times.

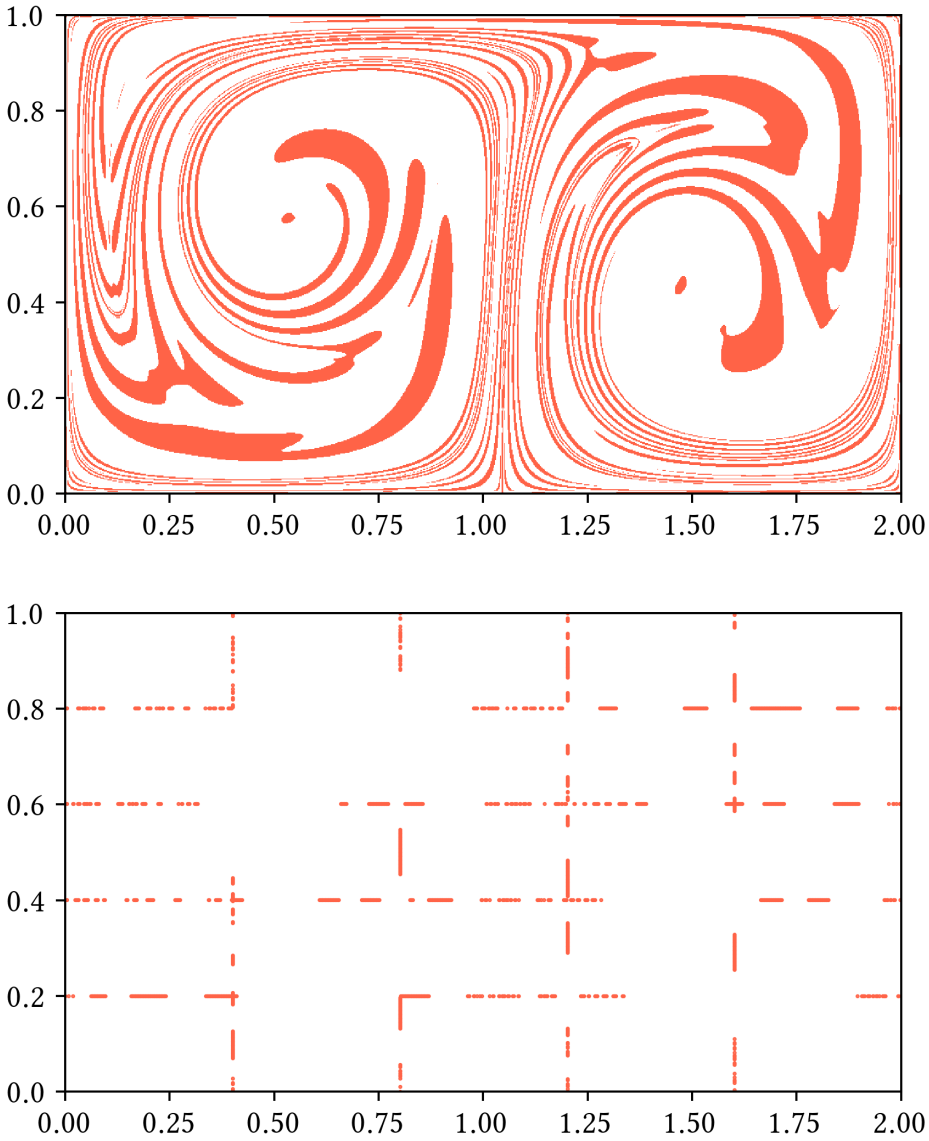


Figure 3.4: The set $\mathcal{U}_0$ for the double gyre system, given by equations (3.1)–(3.3), is shown at the top. This is the set of initial conditions where the LCS existence conditions given in equations (3.13a) and (3.13b) are satisfied. Its itersection with a set of four horizontal and four vertical, equidistant lines is shown at the bottom. The latter was used as the set of strain initial conditions, in order to eliminate redundant computations of strainlines within $\mathcal{U}_0$.

Frequently, only a segment of any given strainline will qualify as a hyperbolic LCS. Hence, the integration of any strainline can be stopped when it reaches a point at which one of the conditions (3.13a) or (3.13b) fails. Doing so uncritically, however, opens up the possibility of stopping a strainline which only exited the $\mathcal{U}_0$ domain due to numerical noise. In order to avoid such unwanted failures, the approach of Farazmand and Haller (2012) was followed. Therefore, strainline integration was only stopped if one of the aforementioned LCS conditions fail repeatedly over a pre-set length $l_f = 0.2$ of the strainline. Furthermore, the strainlines which were stopped because of continuous failure of LCS conditions were cut such that their *new* endpoints in either direction, as seen from the strain initial condition, were the last point on which the *original* strainline satisfied the aforementioned conditions. This concept is illustrated in figure 3.5.

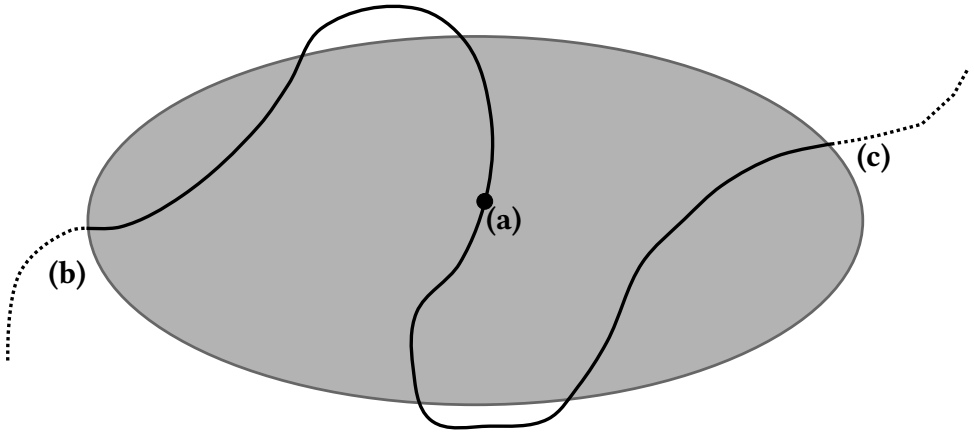


Figure 3.5: Illustration of the concept of strainline tail end cutting. The filled, gray ellipsis constitutes a fictitious set of points $\mathcal{U}_0$, for which the LCS existence conditions expressed in equations (3.13a) and (3.13b) hold. A strainline, starting at (a), is integrated in both directions in pseudotime. In both directions, the strainline integration is eventually stopped due to repeated failure of at least one of the two aforementioned LCS existence conditions, over a pre-set length $l_f = 0.2$. The tails at either end, i.e., the parts of the strainline which exited the $\mathcal{U}_0$ and did not return, are indicated by dashed curves at (b) and (c). These were cut, leaving the solid curve as the part of the strainline which was considered as an LCS candidate. Although parts of *this* curve may reside outside of the $\mathcal{U}_0$ domain, these segments are all *shorter* than l_f.

Identifying strainlines which are local strain maximizers

Now, having located the strainline pieces which satisfy conditions (3.13a) and (3.13b), the next step is imposing condition (3.13d), i.e., identifying the strainline segments that are local maxima of the averaged maximum strain. In the following, we let $\{\gamma_j\}$ denote the set of parametrized strainline curves, computed as solutions of the strain ODE system given in equation (3.15). The suggested approach of Farazmand and Haller (2012) is to define a set $\mathcal{L}$ of uniformly spaced horizontal and vertical lines within the domain $\mathcal{U}_0$. Then, the values of $\overline{\lambda}_2(\gamma_0)$ — the average of λ_2 on the curve γ_0 — are compared at the neighboring intersections of all sufficiently close strainline segments along each of the lines modeline $\mathcal{L}$. The process is illustrated in figure 3.6.

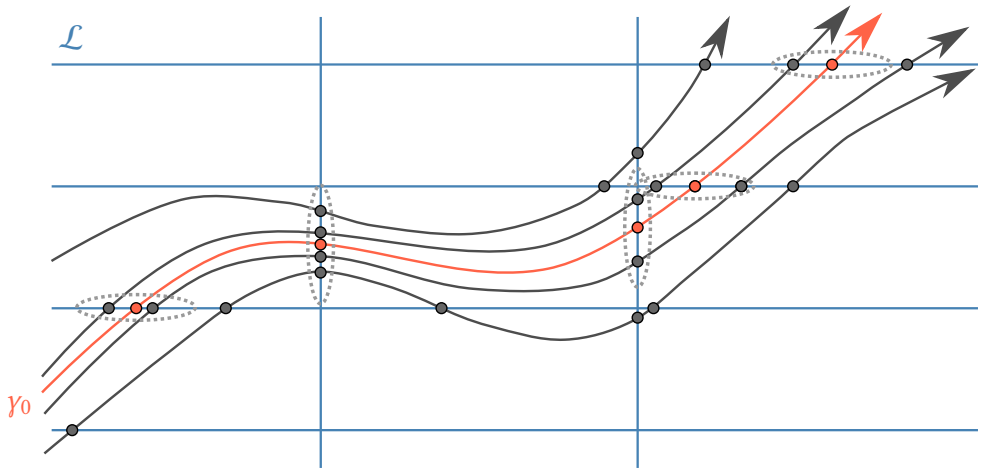


Figure 3.6: Illustration of the identification process of strainlines which are local strain maximizers. Local maxima of $\overline{\lambda}_2(\gamma)$, the average of λ_2 over a curve γ, are found along a set $\mathcal{L}$ of horizontal and vertical lines. The strainline segment γ_0 is identified as a local maximizer if it contains a locally maximal $\overline{\lambda}_2$ value along at least one of the lines in $\mathcal{L}$. The dashed ellipses indicate the regions outside which adjacent intersections are excluded from the local maximization process. The half-width of the ellipses is given as a pre-select threshold $l_{\text{nb}} = 0.2$. Intersections of sufficiently close strainlines, that is, strainlines whose intersection(s) fall fall within the dashed ellipses, are included in the local comparison process for a given intersection between a line in $\mathcal{L}$ and the strainline segment γ_0.

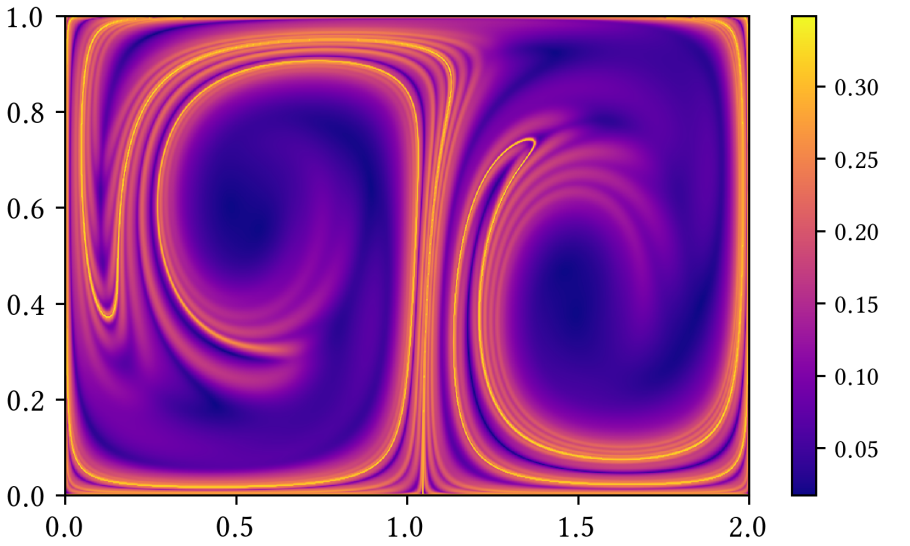
Intersections between strainlines and the lines in $\mathcal{L}$ are found by means of linear interpolation. Should a strainline segment prove to be a local maximizer along at least one of the lines in $\mathcal{L}$, the strainline segment is labelled as an LCS. Adjacent intersections that are separated by a distance larger than a preselected threshold $l_{\text{nb}} = 0.2$ were excluded from the local maximization process, as indicated by the dashed ellipses in figure 3.6. Furthermore, short LCSs are expected to have negligible impact on the overall flow pattern. In order to filter out such miniscule structures, any strainline segment which was shorter than a pre-set small length $l_{\text{min}} = 1$ were discarded as LCS candidates, as suggested in Farazmand and Haller (2012).

One shortcoming in the work of Farazmand and Haller (2012), is that they do not provide a way of selecting the lines in $\mathcal{L}$. Neither has a general, robust way of identifying strainlines which

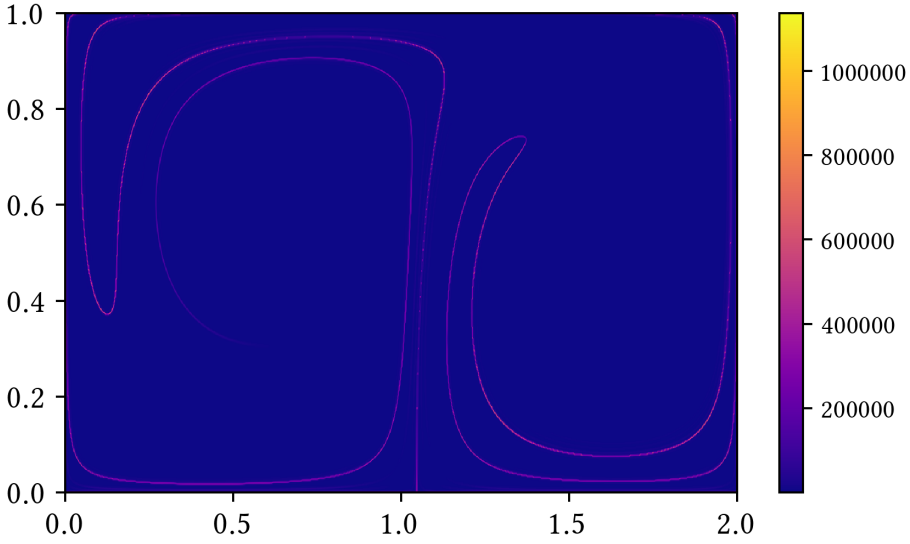
are local strain maximizers yet been described literature, to our knowledge. For this reason, Farazmand and Haller's approach was used, with a set of lines determined by studying the domain $\mathcal{U}_0$ (as shown in figure 3.4) in relation to the $\lambda_2(\mathbf{x}_0)$ distribution, as well as the FTLE field (described in section 2.4), where the latter two are shown in figure 3.7.

By inspection of figure 3.4, we will for example expect there to exist strainline segments in the upper right part of the computational domain. So, if any of the lines in $\mathcal{L}$ pass through this region, we must expect to find one or more strainline segments which are labelled as LCSs. From figure 3.7, one may infer that the local strain in the region is relatively small, seeing as there is practically no discernible structure there at all in the $\lambda_2(\mathbf{x}_0)$ distribution, whereas the FTLE values there are also very small. From this, it follows that any hypothetical LCSs located in this region will have a relatively small impact on the overall flow in the system, at least compared with other hypothetical LCSs which lie on, or near, the recognizable ridges. The same logic is practically applicable on the entirety of the computational domain.

In order to eliminate the weakest among the LCSs, it is thus clear that the lines in $\mathcal{L}$ must be selected carefully. Here, these lines were chosen in order to maximize the number of intersections with the computed strainlines in the regions within $\mathcal{U}$ where the local strain is expected to be largest, using as few lines as possible. As previously mentioned, this was done by studying figures 3.4 and 3.7. Using two vertical and two horizontal lines, given by $x = 0.15$, $x = 1.05$, $y = 0.05$ and $y = 0.95$, respectively, proved sufficient in order to accurately reproduce the same LCS as found by Farazmand and Haller (2012).



(a) FTLE field



(b) $\lambda_2(\mathbf{x}_0)$ distribution

Figure 3.7: Plots of the FTLE field (a) and $\lambda_2(\mathbf{x}_0)$ distribution (b) of the double gyre system, given by equation (3.1). Due to the different scalings, different levels of detail are resolved. Most notably, the $\lambda_2(\mathbf{x}_0)$ contains a thin, patchy ridge of extremal values, whereas a similar, albeit continuous structure is recognizable in the FTLE field. Moreover, the FTLE field exhibits a greater amount of detail away from the most prominent ridge than the $\lambda_2(\mathbf{x}_0)$ distribution. Both plots were used, together with the domain $\mathcal{U}_0$ shown in figure 3.4, in order to select the lines in $\mathcal{L}$, illustrated in figure 3.6.

The reference LCS, that is, the LCS obtained with the reference numerical integrator, as mentioned in section 3.2.2, is presented in figure 3.8. It consists of seven segments, and agrees visually with the LCS found in the literature, for example by Farazmand and Haller (2012). Note, however, that Farazmand and Haller claim that their LCS consists of a single strainline segment. This will be elaborated upon in chapter 5. The curve shown in figure 3.8 was used to benchmark how the different numerical integrators, mentioned in section 2.1.2, for the parameters mentioned in section 3.2.2, performed.

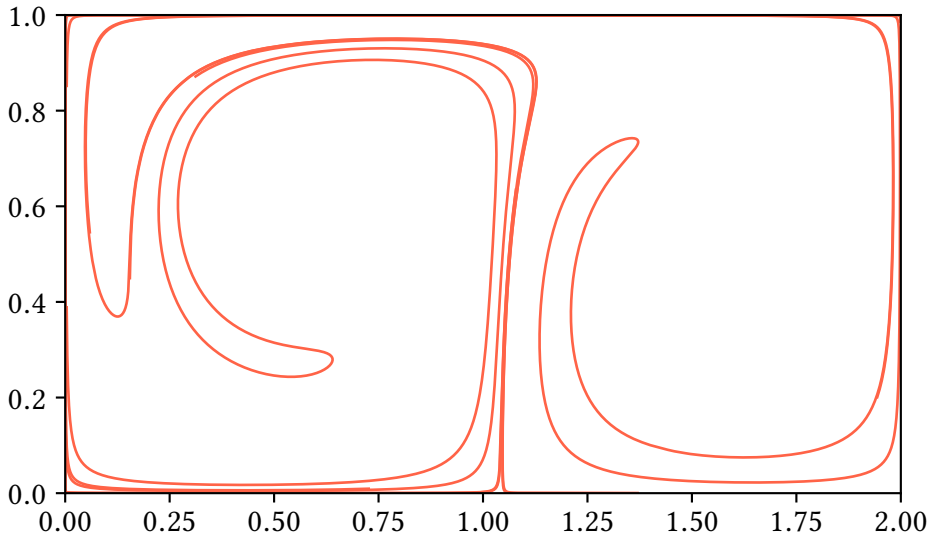


Figure 3.8: The repelling reference LCS of the double gyre system, i.e., the LCS obtained by means of the reference numerical integrator, as outlined in section 3.2.2. This curve agrees visually with the LCS found in the literature, for instance in Farazmand and Haller (2012). It was used as the benchmark for comparing the accuracy of the considered numerical integrators, outlined in section 2.1.2, for varying numerical step lengths and tolerance levels.

3.5 Estimation of errors

Aside from the qualitiative visual comparison between the obtained LCSs and the reference LCS, a set of numerical errors was also computed, with a view to use them in order to explain any visual discrepancies. To this end, the error in the flow map, given by equation (2.13), was computed as the root mean square deviation (hereafter abbreviated to RMSD) with regards to the reference, as follows:

$$\mathrm{RMSD}_{\text{flow map}} = \sqrt{\frac{1}{\widetilde{N}} \sum_{\mathbf{x}_0 \in \widetilde{\mathcal{U}}} \left(\widehat{\mathbf{F}}_{t_0}^{t}(\mathbf{x}_0) - \mathbf{F}_{t_0}^{t}(\mathbf{x}_0) \right)^2}, \tag{3.17}$$

where the summation is over all of the tracers in the extended computational domain $\widetilde{\mathcal{U}}$, that is, both the main and auxiliary tracers which originate from points in the extended grid, as outlined in section 3.2.1. $\widetilde{N}$ is the total number of advected tracers, and $\widehat{\mathbf{F}}_{t_0}^{t}(\mathbf{x}_0)$ is the flow map approximated by the numerical integrator in question.

The RMSD in both sets of eigen*values*, $\lambda_1(\mathbf{x}_0)$ and $\lambda_2(\mathbf{x}_0)$ of the Cauchy-Green strain tensor field, as given by equation (2.17) was also computed analogously:

$$\mathrm{RMSD}_{\text{eigenvalue}} = \sqrt{\frac{1}{N} \sum_{\mathbf{x}_0 \in \mathcal{U}'} \left(\widehat{\lambda}(\mathbf{x}_0) - \lambda(\mathbf{x}_0) \right)^2}, \tag{3.18}$$

where the summation is over all main tracers located in (or at the boundary of) the domain $\mathcal{U}'$, the borders of which are governed by the first set of rows and columns in the extended grid. N is the corresponding total number of main tracers in $\mathcal{U}'$, while $\widehat{\lambda}(\mathbf{x}_0)$ is the approximation of the true eigenvalues $\lambda(\mathbf{x}_0)$ located at $\mathbf{x}_0$, obtained by means of the numerical integrator in question.

Because the eigenvectors of the Cauchy-Green strain tensor field are normalized to unit length, a reasonable way of estimating the error in the computed eigen*vectors* is by considering their orientation. Thus, the eigenvector directions were computed as azimuthal angles. The RMSD for the direction of the eigenvector fields was then calculated as follows:

$$\begin{aligned} \boldsymbol{\xi}(\mathbf{x}_0) &= \left(\xi_x(\mathbf{x}_0), \quad \xi_y(\mathbf{x}_0) \right), \quad \phi(\mathbf{x}_0) = \arctan\left(\frac{\xi_y(\mathbf{x}_0)}{\xi_x(\mathbf{x}_0)} \right), \\ \mathrm{RMSD}_{\text{eigenvector direction}} &= \sqrt{\frac{1}{N} \sum_{\mathbf{x}_0 \in \mathcal{U}} \left(\widehat{\phi}(\mathbf{x}_0) - \phi(\mathbf{x}_0) \right)^2}, \end{aligned} \tag{3.19}$$

where the conventions for $\mathcal{U}$ and N are the same as for the RMSD of the eigenvalues, given in equation (3.18). $\widehat{\phi}(\mathbf{x}_0)$ denotes the azimuthal angle of the eigenvector located at $\mathbf{x}_0$, found by means of the numerical integrator in question.

Regarding the error in the computed LCS curves, false positives, that is, any parts which of the computed LCS curves which are not present in the reference curve, and false negatives,

i.e., any parts of the reference curve which is not present in the computed LCS curves, should be treated separately. This is because as they would influence the predictions of overall flow patterns in the system in different ways. Numerical noise could result in LCS curve segments being erroneously identified as either a false positive or a false negative. In order to reduce this sort of error, a lower numerical threshold $l_{\text{noise}} = 0.01$, that is, ten times the numerical integration step $\Delta = 10^{-3}$ used in order to compute the constituent strainlines, was used in identifying false positives and negatives; any point on an LCS curve which was farther away from all points on the reference LCS than l_{noise} was identified as a false positive. Similarly, any point on the reference LCS curve which was farther away from all points on the LCS curve under consideration than l_{noise} was flagged as a false negative.

Let $\widehat{\gamma}$ represent the parametrization of the computed LCS approximation under consideration, and γ the parametrization of the reference LCS. In order to estimate the offset of the false LCS segments, the area between these segments and the reference LCS curve was approximated by the midpoint rule for numerical integration;

$$\text{Offset}_{\text{false LCSs}} = \sum_{\substack{\mathbf{x} \in \gamma \\ \mathbf{x} \notin \widehat{\gamma}}} \min_{\widehat{\mathbf{x}} \in \widehat{\gamma}} \|\widehat{\mathbf{x}} - \mathbf{x}\| \cdot \Delta, \tag{3.20}$$

where the sum is over all points with a greater offset than the aforementioned l_{noise}. Regarding false negatives, the roles of γ and $\widehat{\gamma}$ are reversed. Per equation (3.15), Δ, the previously mentioned numerical integration step length used in order to compute strainlines, equals the distance separating consecutive points in the parametrization of the LCS curves. This is due to the eigenvectors ξ_1 being normalized to unit length. The calculation of this offset is illustrated in figure 3.9.

Lastly, in order to obtain a quantitative measure of the offset in the LCS curve segments which comply with the reference LCS (that is, to the numerical threshold l_{noise}), the RMS distance separating each point on the LCS curve with the nearest point on the reference, was calculated:

$$\text{RMSD}_{\text{LCS}} = \sqrt{\frac{1}{\breve{N}} \sum_{\mathbf{x} \in \gamma} \min_{\widehat{\mathbf{x}} \in \widehat{\gamma}} \|\widehat{\mathbf{x}} - \mathbf{x}\|^2}, \tag{3.21}$$

where the term RMSD is used somewhat loosely, in order to conform with notation used for the other measures of error introduced in this section. $\breve{N}$ is the number of points which constitute $\widehat{\gamma}$, the LCS curve found by means of the numerical integrator in question, which are within the distance l_{noise} of any point on γ, the reference LCS curve. The idea is that LCS curves need not necessarily start from nor end at the same points in space. Thus, a robust way of estimating the error of a given LCS curve is the summation of the smallest distances separating each point on it from the reference.

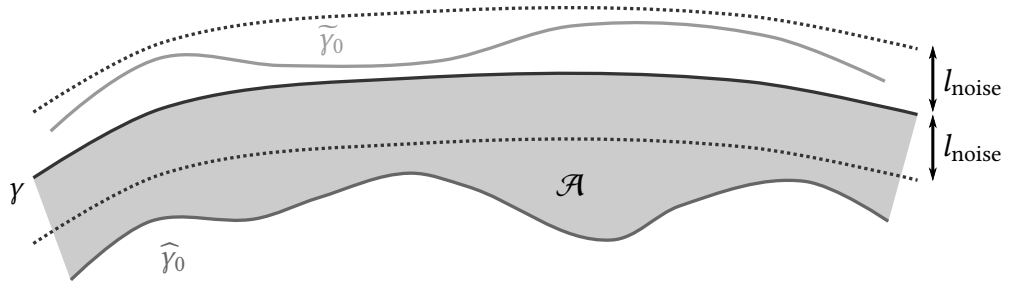


Figure 3.9: Illustration of how the offset of false LCS segments was computed. A segment of the reference LCS is denoted γ. Two segments of an LCS approximation are denoted $\widetilde{\gamma}_0$ and $\widehat{\gamma}_0$, respectively. When one or more such segments are within a pre-set length $l_{\text{noise}} = 0.01$ of γ — where the borders of this region are indicated by dashed lines — any other segments which are further away are labelled as false positives. Accordingly, $\widehat{\gamma}_0$ is recognized as a false positive in the above figure. Then, the shaded area $\mathcal{A}$ between the reference LCS and the curve segments identified as false positives is estimated by means of the midpoint rule. That is, for each point on a false positive segment of the approximated LCS, the smallest distance separating it and the reference is weighted with Δ, the step length used in the numerical integration of strainlines. Because the ξ_1-field is normalized to unit length, Δ equals the distance between consecutive points in the parametrizations of both the approximated and the reference LCS curves, per equation (3.15). In the case of false negatives, no LCS curve segments fall within l_{noise}, in which case $\mathcal{A}$ denotes the minimal area between the reference LCS curve and the approximated LCS curve, for each point of the reference LCS segment which is not found in the approximation. In the figure, this would correspond to the absence of $\widetilde{\gamma}_0$, if $\widehat{\gamma}_0$ was then the part of the LCS approximation closest to γ.

4 Results

4.1 The LCS curves obtained using the different schemes

Here, the repelling LCS curves found by means of the variety of numerical integration schemes under consideration, are presented. All of the LCS curves obtained for a given integrator, for all numerical time step lengths or tolerance levels are included in one figure, where the reference LCS, shown in figure 3.8, is also included in order to facilitate visual comparison. The idea is that any LCS curve that deviates from the reference, will reveal itself by not conforming perfectly. The LCS curves resulting from the singlestep methods are presented in figures 4.1–4.4, whereas the ones found by virtue of the embedded, that is, adaptive stepsize, methods, are shown in figures 4.6–4.9.

4.1.1 *LCS curves stemming from singlestep methods*

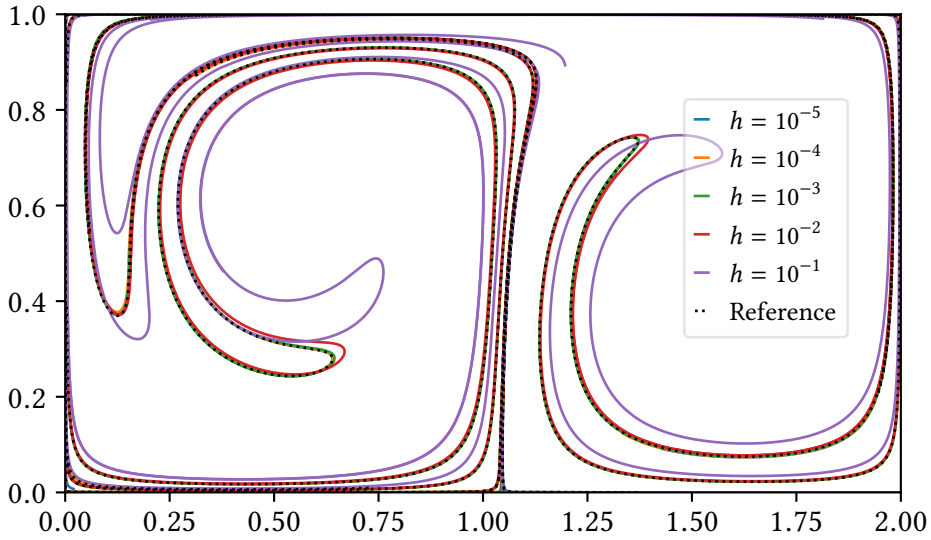


Figure 4.1: LCS curves found by means of the Euler integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. There is a clearly visible offset compared to the reference, for all but the two smallest numerical step lengths considered. The two latter LCS curves appear to conform well.

The LCS curves obtained from the Euler scheme using relatively large step lengths are the most obviously incorrect ones, in comparison to the reference. There appears to be very little separating the performance of the other singlestep methods. Interestingly, the Heun scheme appears to yield more accurate results than the Kutta scheme for all the considered step lengths. This is somewhat unexpected, because, as mentioned in section 2.1.2, the Heun scheme is 2^{nd}-order accurate in the time step, whereas the Kutta scheme is 3^{rd}-order accurate. On a less surprising note, seeing as it is 4^{th}order accurate, the classical Runge-Kutta scheme produces very accurate curves for all step lengths.

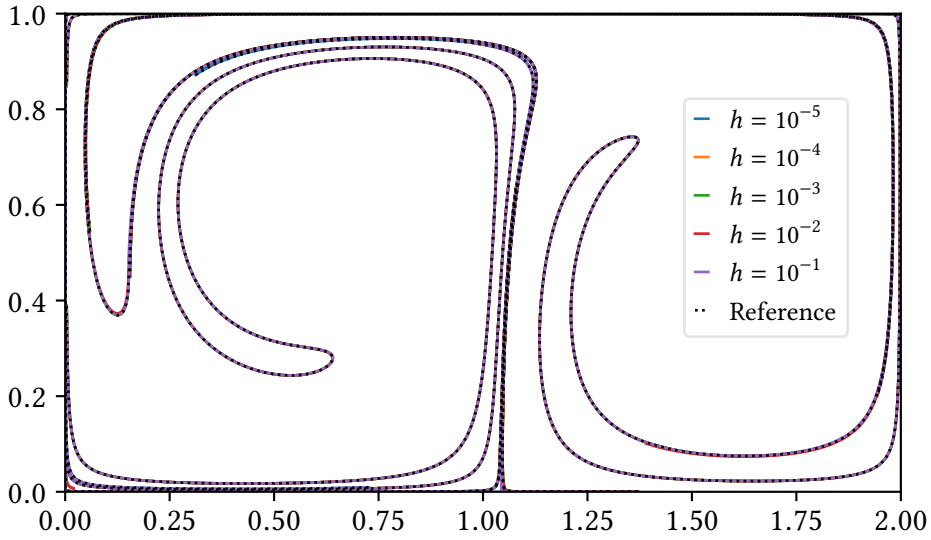


Figure 4.2: LCS curves found by means of the Heun integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. There exists discrepancies with regards to the reference for the two largest numerical time step lengths considered. These are most prominent in the lower left corner, and near $x = 1$.

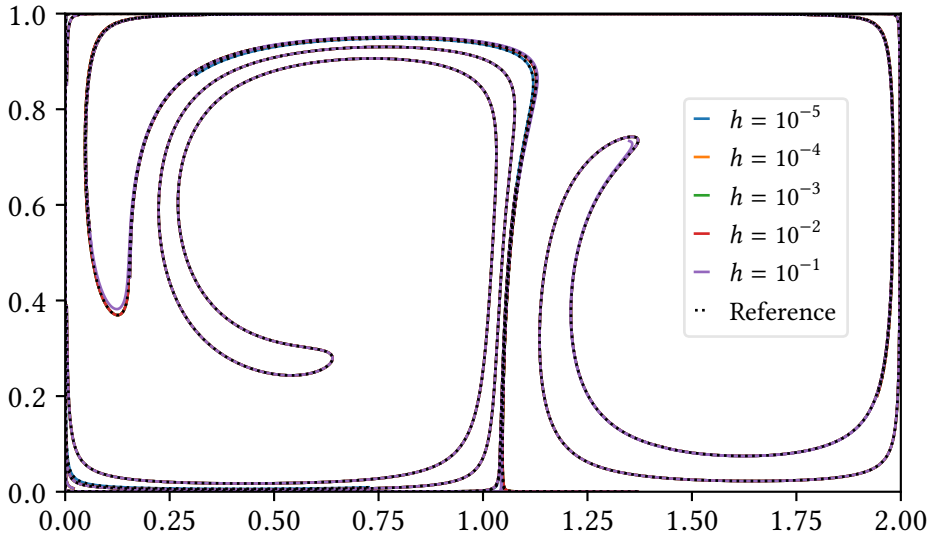


Figure 4.3: LCS curves found by means of the Kutta integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. There are some disparities with regards to the reference both for the two largest numerical time step lengths considered. These are most pronounced in the lower left corner, near the leftmost '∪' shape by $y = 0.4$ and the '∩' shape near $x = 1.25$.

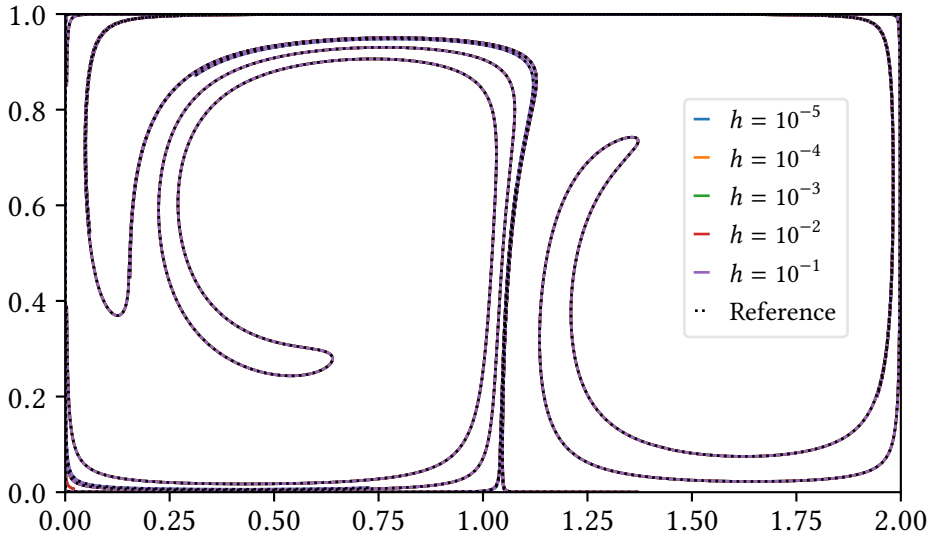
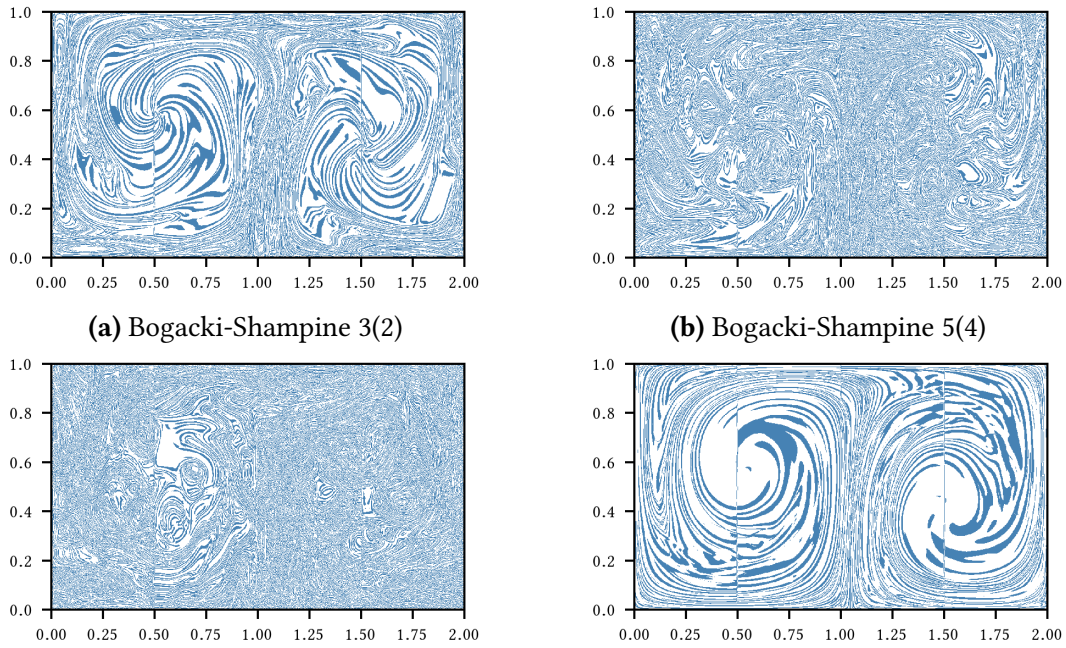


Figure 4.4: LCS curves found by means of the classical Runge-Kutta integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. One of the very few visible discrepancies belongs to the second largest numerical time step length considered, and is located in the lower left corner.

4.1.2 *LCS curves stemming from adaptive stepsize methods*

Note that, although numerical tolerance levels of 10^{-1} through to 10^{-10} were investigated, the figures presented here do not show LCS curves for tol = 10^{-1}. This is due to the fact that, for all embedded methods, none of the resulting domains $\mathcal{U}_0$ bore a great amount of resemblance to the reference. This resulted in very disparate behaviour of strainlines, to the extent that the resulting strain systems were too dissimilar to the reference to warrant any meaningful comparisons. The erroneous $\mathcal{U}_0$ domains are shown in figure 4.5.

Notably, all of the embedded integration methods result in LCSs which are dissimilar from the reference for the tolerance level tol = 10^{-2}, where only the curve obtained by the Dormand-Prince 8(7) scheme is anything alike the reference. More curiously, the Bogacki-Shampine 3(2) method appears to outperform its theoretically more accurate sibling — the Bogacki-Shampine 5(4) method —, even for relatively large tolerance levels. This is visible in the vicinity of the 'U' shape near $y = 0.4$ in figures 4.6 and 4.7, for instance. Furthermore, the Dormand-Prince 5(4) method appears to yield more accurate LCS curves in general, than the Bogacki-Shampine 5(4) method. This is somewhat surprising, given that the latter scheme is supposedly more accurate than other methods of similar order, as mentioned in section 2.1.2. Lastly, the Dormand-Prince 8(7) method appears to produce very accurate results for all tolerance levels smaller than 10^{-2}, and, as previously mentioned, gives the most correct LCS curve for the tolerance level 10^{-2} — which is to be expected, as it is the highest order method among the ones considered, after all.



(a) Bogacki-Shampine 3(2)

(b) Bogacki-Shampine 5(4)

(c) Dormand-Prince 5(4)

(d) Dormand-Prince 8(7)

Figure 4.5: The $\mathcal{U}_0$ domains obtained with the adaptive stepsize integration schemes, with numerical tolerance level tol $= 10^{-1}$. Out of the four schemes, only the Dormand-Prince 8(7) method is remotely close to reproducing the reference domain, shown in 3.4. In any case, the differences are sufficiently large to alter the resulting strain system beyond recognition, rendering LCS comparisons with the reference impractical.

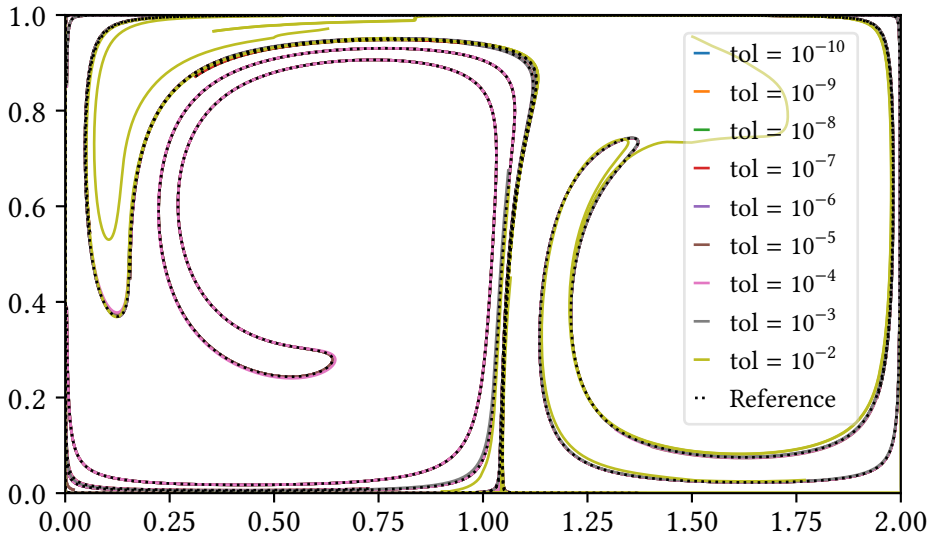


Figure 4.6: LCS curves found by means of the Bogacki-Shampine 3(2) integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. Note that the LCS for the lowest tolerance level considered, that is, tol $= 0.1$, is not included. This is because the corresponding $\mathcal{U}_0$ domain, shown in figure 4.5a, and the reference $\mathcal{U}_0$, shown in figure 3.4 are very dissimilar. The second lowest tolerance level, i.e., tol $= 10^{-2}$, is the main culprit among the remaining, as far as discrepancies are concerned.

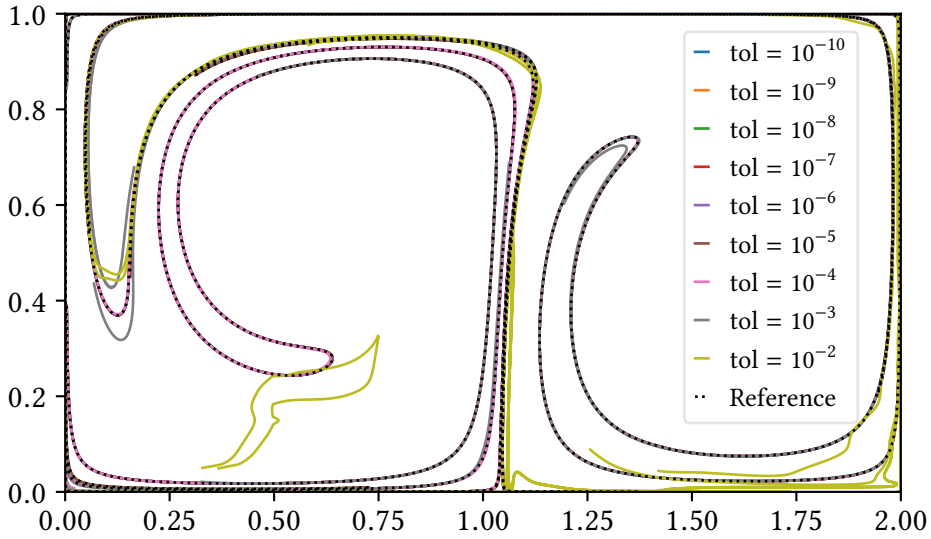


Figure 4.7: LCS curves found by means of the Bogacki-Shampine 5(4) integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. Note that the LCS for the lowest tolerance level considered, that is, tol = 0.1, is not included. This is because the corresponding $\mathcal{U}_0$ domain, shown in figure 4.5b, and the reference $\mathcal{U}_0$, shown in figure 3.4 are very dissimilar. Here, there are visible discrepancies for all tolerance levels tol $> 10^{-6}$. These are prominent all over the domain.

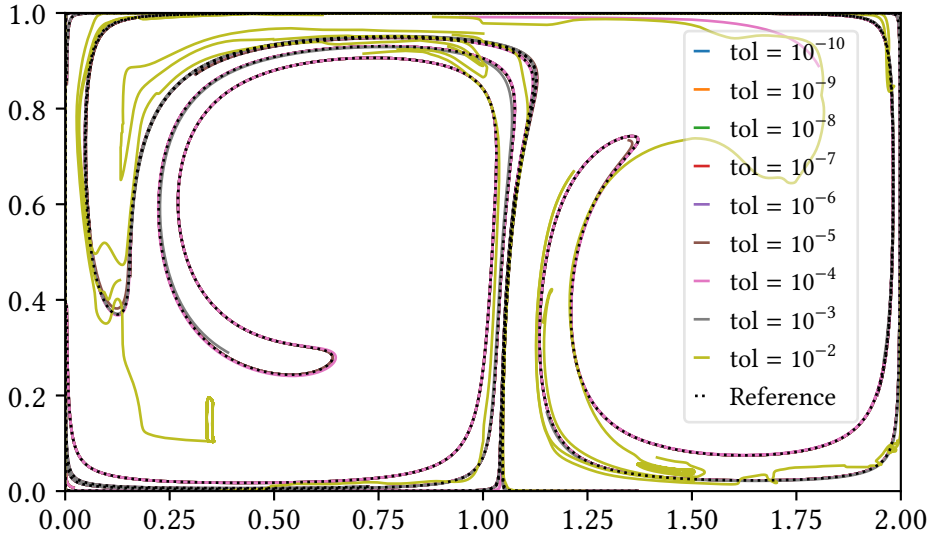


Figure 4.8: LCS curves found by means of the Dormand-Prince 5(4) integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. Note that the LCS for the lowest tolerance level considered, that is, tol = 10^{-1}, is not included. This is because the corresponding $\mathcal{U}_0$ domain, shown in figure 4.5c, and the reference $\mathcal{U}_0$, shown in figure 3.4 are very dissimilar. Here, there are visible disparities for for all tolerance levels tol $> 10^{-6}$. These are noticable throughout the domain.

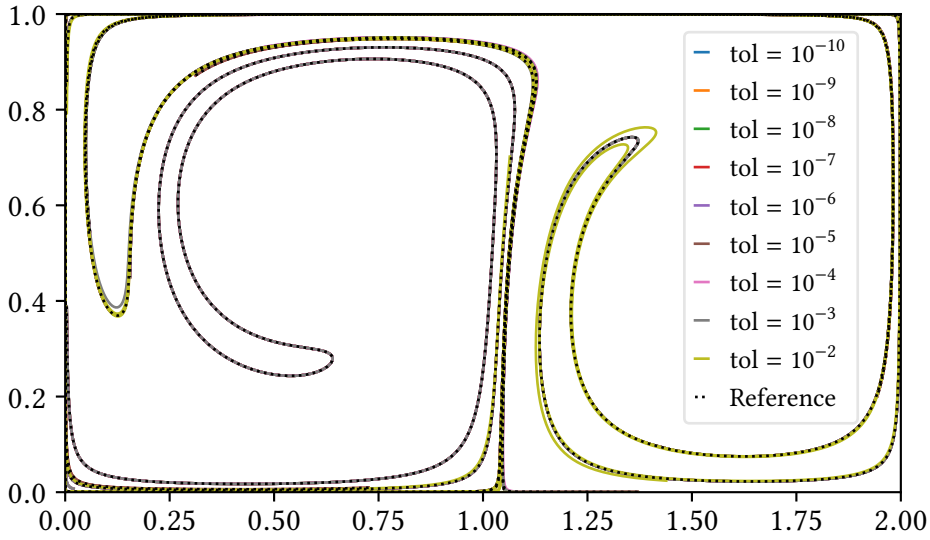


Figure 4.9: LCS curves found by means of the Dormand-Prince 8(7) integration scheme. The reference LCS, as shown in figure 3.8, is dashed on the top layer. Note that the LCS for the lowest tolerance level considered, that is, tol = 10^{-1}, is not included. This is because the corresponding $\mathcal{U}_0$ domain, shown in figure 4.5d, and the reference $\mathcal{U}_0$, shown in figure 3.4 are quite unlike one another. Here, the most immediately discernible dissimilarities correspond to the tolerance level tol = 10^{-2}.

4.2 Measures of error

Here, the measures of error introduced in section 3.5 for the different numerical integration schemes considered, are presented. For each different type of error, two figures are included. One in which the error of the singlestep methods is shown as a function of the integration step length, and one in which the error of *all* the integration methods is shown as a function of the number of times right hand side of the ordinary differential equation system, that is, the velocity field given by equation (3.1), was evaluated for each step length or tolerance level. The reason for which only the singlestep errors are included in the first set of figures, is that they are the only methods where the error is expected to scale as a power of h, per definition 2. This is naturally not the case for the adaptive stepsize methods, where the step length varies.

More pertinent information can be found in the second set of figures, where a well-suited integration method is characterized by generating small errors at a small cost in terms of the required number of function calls, in this case, the number of times the velocity field had to be evaluated. Unlike the singlestep integrators, the advection of each tracer by means of an adaptive stepsize method in principle requires a different number of integration steps, and thus function evaluations. Thus, the presented number of function evaluations is the *average* across all of the advected tracers, including the number of evaluations associated with rejected trial integration steps. The number of function evaluations for each step of the considered Runge-Kutta methods can be found as the number of rows in the Runge-Kutta matrices of tables 2.3–2.10.

4.2.1 *Computed deviations in the flow maps*

Figure 4.10 indicates that the RMSD of the flow maps, as defined in equation (3.17) follow the expected power laws of the numerical errors for the various singlestep integrators. This is as expected, because the flow maps are found by direct application of the numerical integrators. In addition, the flow map RMSDs seem to be limited from below by the counteracting accumulated floating-point arithmetic error, as hypothesized in section 3.2.2. This effect is most prominent in the error curves of the Kutta and classical Runge-Kutta schemes, which in fact *decrease* as a function of the step length until $h = 10^{-4}$ and $h = 10^{-3}$, respectively. Based on this evidence, the chosen step lengths appear to be appropriate.

Figure 4.11 indicates that a peak in numerical accuracy did not materialize for the adaptive stepsize methods, for the considered numerical tolerance levels. The figure also reveals that the Bogacki-Shampine 3(2) scheme is 'cheap' when only a very crude approximation of the flow map is needed; in particular, the aforementioned scheme requires less function evaluations than even the Dormand-Prince 8(7) method for RMSDs upwards of 10^{-1}. However, for smaller RMSD values, the Bogacki-Shampine 3(2) scheme falls inbetween the Heun and Kutta schemes, which is to be expected, seeing as the latter methods are of 2$^{\text{nd}}$ and 3$^{\text{rd}}$ order, respectively. Notably, the Bogacki-Shampine 5(4) scheme is able to keep up with the Dormand-Prince 8(7) method regarding efficiency, until approximately 600 function evaluations. This complies with the notion that the Bogacki-Shampine 5(4) method practically behaves as an even higher order method, as mentioned in section 2.1.2. There does not seem to be a direct correspondence between the RMSD of the flow maps, and that of the computed LCS curves, which will be presented in section 4.2.3 — which seems reasonable, given that a lot of computations separate the flow maps from the resulting LCS curves, as outlined in chapter 3.

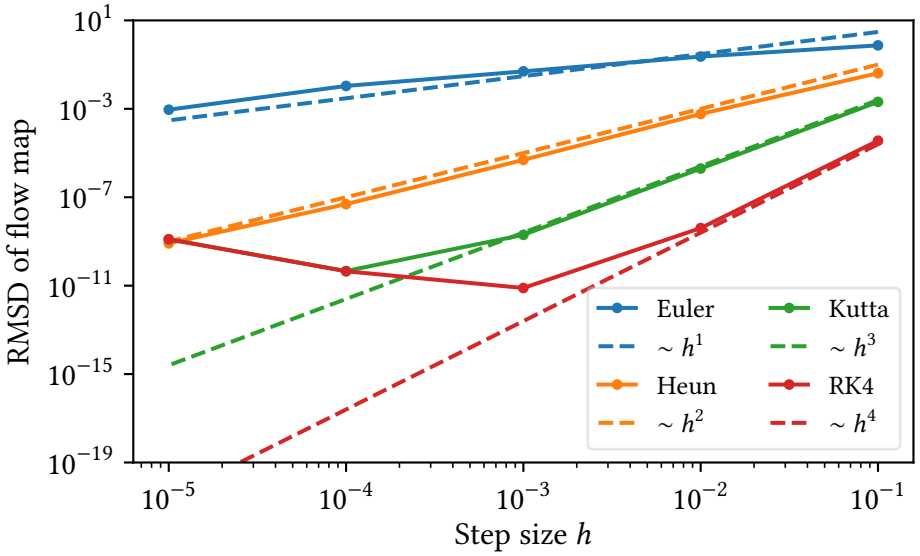


Figure 4.10: RMSD of the flow maps as a function of numerical step length, for the singlestep methods considered. The dashed curves have slopes corresponding to the expected scalings with the step length for each method.

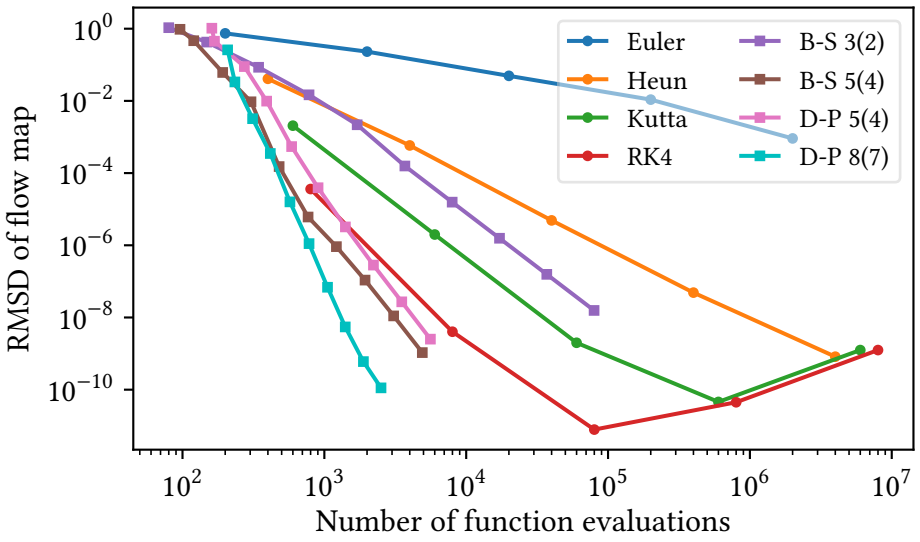


Figure 4.11: RMSD of all flow maps as a function of the required number of function evaluations per advected tracer. For the adaptive stepsize methods, the *average* number of function evaluations across all advected tracers is used as the abscissa.

4.2.2 *Computed deviations in the strain eigenvalues and -vectors*

For brevity, only the RMSD of λ_2 and the direction of ξ_2, are included here. The latter is justified because of the orthogonality of the eigenvectors of the Cauchy-Green strain tensor, which means that the RMSD of the directions of both eigenvectors *must* be identical. As it turns out, the RMSD of λ_1 scales similarly to that of λ_2, both as a function of numerical step length (for the singlestep methods) and the number of function evaluations. Additionally, the numerical reformulation of the existence theorem for hyperbolic LCSs, given in equation (3.13), exhibits a more sensitive dependence to λ_2 than λ_1. This is explicitly observable from conditions (3.13b) and (3.13d), and favors λ_2 as the most crucial eigenvalue.

Figure 4.12 indicates that the RMSD of λ_2, as defined in equation (3.18), follows the anticipated power law of the numerical errors for the various singlestep integrators. Interestingly, the RMSD dependence on the numerical steplength is completely analogous to that of the RMSD of the flow maps, as presented in section 4.2.1. Furthermore, inspection of figure 4.13 reveals that the RMSD of the λ_2 scales similarly to the RMSD of the flow maps for all of the integration schemes, differing only by a numerical prefactor. This makes sense, seeing as the calculation of the Cauchy-Green strain tensor field is based upon finite differencing applied to the flow map, as described in section 3.3.

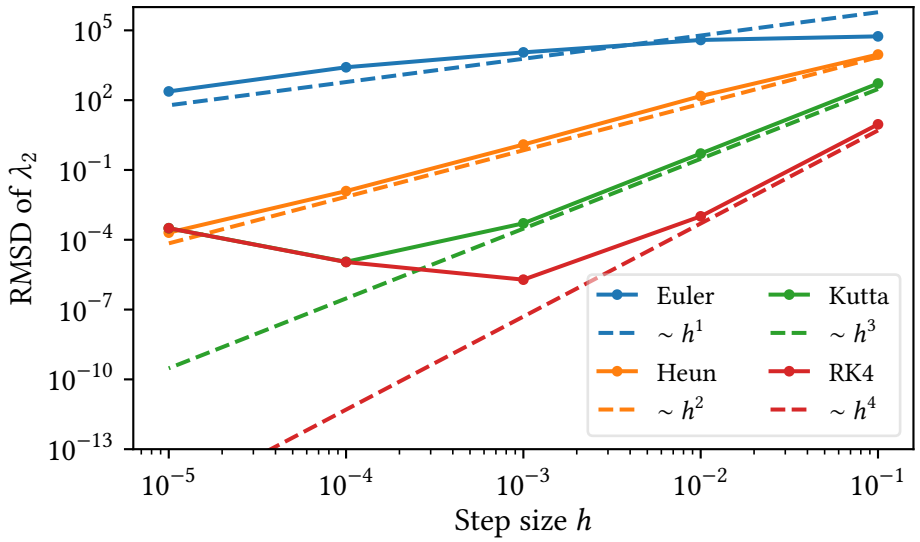


Figure 4.12: RMSD of the λ_2 field as a function of numerical step length, for the singlestep methods considered. The dashed curves have slopes corresponding to the expected scalings with the step length for each method.

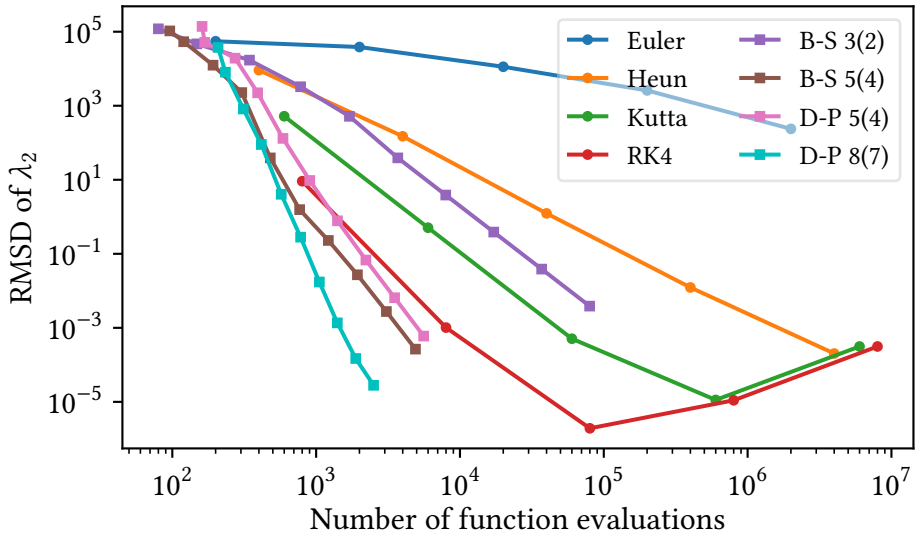


Figure 4.13: RMSD of the λ_2 field as a function of the required number of function evaluations per advected tracer. For the adaptive stepsize methods, the *average* number of function evaluations across all advected tracers is used as the abscissa.

Figure 4.14 reveals that the RMSD of the eigenvector direction, as defined in equation (3.19), is negligible for all numerical step lengths. In fact, the numerical errors are comparable to, or even smaller than, the machine epsilon of double-precision floating-point numbers, which, as mentioned in section 3.2.1, is of order 10^{-16} (Institute of Electrical and Electronics Engineers 2008). Because these errors are so small, the agreement with the anticipated scalings with the stepsize should not necessarily be taken as conclusive evidence of general behaviour. Figure 4.15, shows that this is in fact the case for the adaptive stepsize methods too, aside from the tolerance level tol = 10^{-1}, which corresponds to the fewest function evaluations for each scheme. As has already been established in figure 4.5, for that tolerance level, none of the integrators yield $\mathcal{U}_0$ domains with reasonable degrees of resemblance to the reference, which is shown in figure 3.4.

Aside from the least accurate tolerance level for the embedded integrators, though, the error in eigenvector direction is comparable to, or smaller than, double-precision machine epsilon. Thus, we may conclude that the eigenvector directions are computed correctly for just about any tolerance level and numerical time step length. This can be understood as a consequence of the use of the auxiliary tracers in order to compute the strain eigenvectors, as described in section 3.3, which, by construction, is more accurate than using the main tracers. Moreover, these results imply that the error in the resulting LCS curves are strongly driven by the error in the computed eigenvalues. The sharp turns in the computed LCS curves for the tolerance level 10^{-2}, which are particularly visible in figures 4.7 and 4.8, are likely a consequence of the corresponding $\mathcal{U}_0$ domains extending to regions in which there are very strong local orientational discontinuities in the ξ_1-field, for which the special-purpose linear interpolation introduced in section 3.4.1 is evidently insufficient.

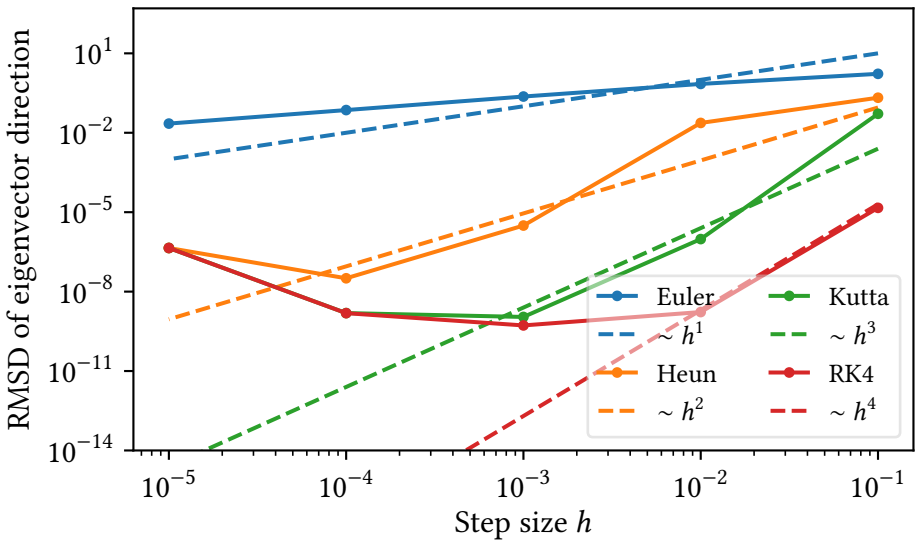


Figure 4.14: RMSD of the strain eigendirections as a function of numerical step length, for the singlestep methods considered. The dashed curves have slopes corresponding to the expected scalings with the step length for each method.

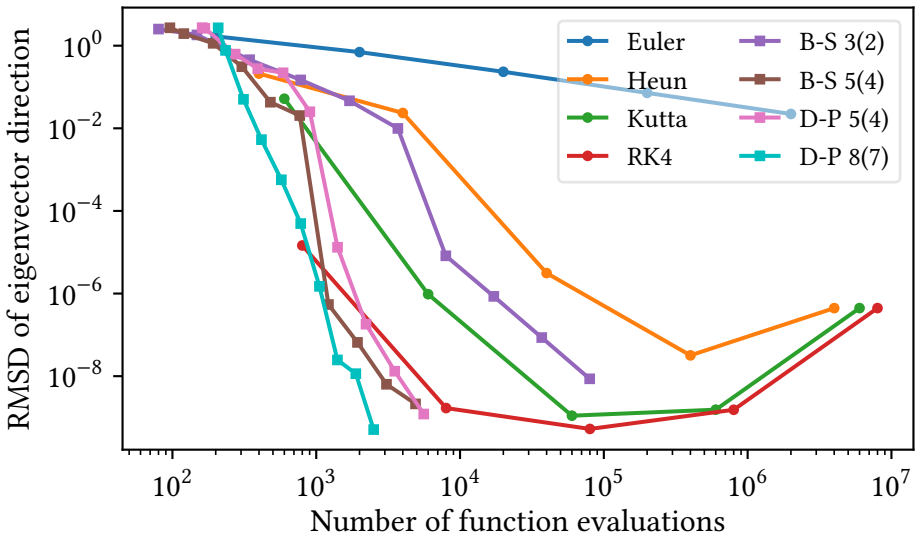


Figure 4.15: RMSD of the strain eigendirections as a function of the required number of function evaluations per advected tracer. For the adaptive stepsize methods, the *average* number of function evaluations across all advected tracers is used as the abscissa.

4.2.3 *Computed deviations in the LCS curves*

Figures 4.16 and 4.17 indicate that the integrated offset of false positives and negatives, as defined in equation (3.20), is a decreasing function of the number of function evaluations for all integration schemes. Moreover, the issue of false positives and negatives vanishes entirely for sufficiently many function evaluations, that is, sufficiently small numerical step lengths or tolerance levels. This conforms well with the visual representations of the various approximations of the LCS curve, as shown in section 4.1.

Note that the offset of false positives and negatives has not been plotted as a function of numerical time step for the singlestep methods. As can be seen from figures 4.16 and 4.17, only the Euler method results in false positives or negatives for more than one of the considered integration step lengths, meaning that there are simply too few data points to identify correlations between these numerical errors and the time step, cf. definition 2. Moreover, there is no clear reason why these offsets should scale similarly as, for instance, the errors in the computed flow maps (see section 4.2.1). Furthermore, offsets for the adaptive stepsize methods are not included for the tolerance levels 10^{-1}, because the LCS identification proved to be exceedingly demanding in terms of computations for those cases, as described in section 4.1.2. However, based on the associated $\mathcal{U}_0$ domains, shown in figure 4.5, one may infer an overwhelming probability of a large number of false positives and negatives to be present, seeing as the underlying strain systems are quite clearly different to the reference, shown in figure 3.4.

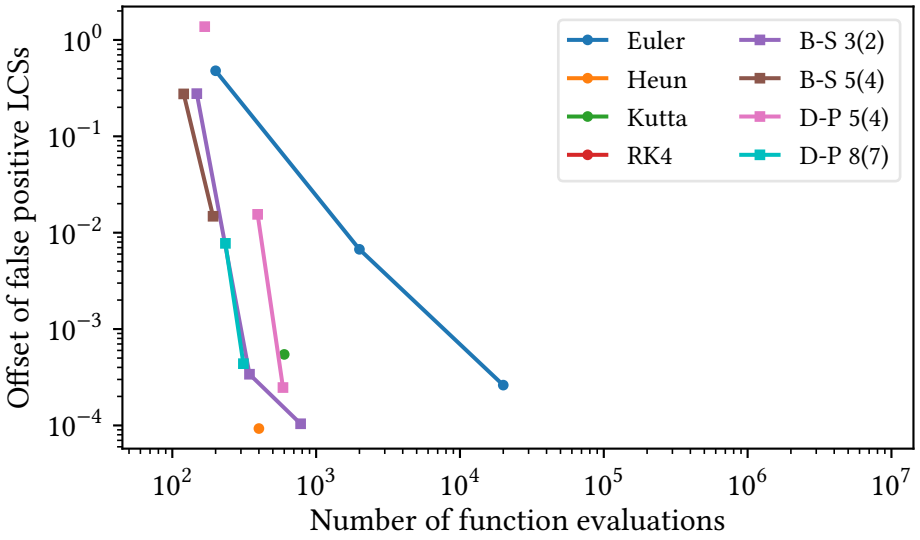


Figure 4.16: Offset of the LCS curve segments identified as false positives as a function of the required number of function evaluations per advected tracer. The integration schemes for which no points are visible, yielded no false positives. For the adaptive stepsize methods, the *average* number of function evaluations across all of the advected tracers is used as the abscissa.

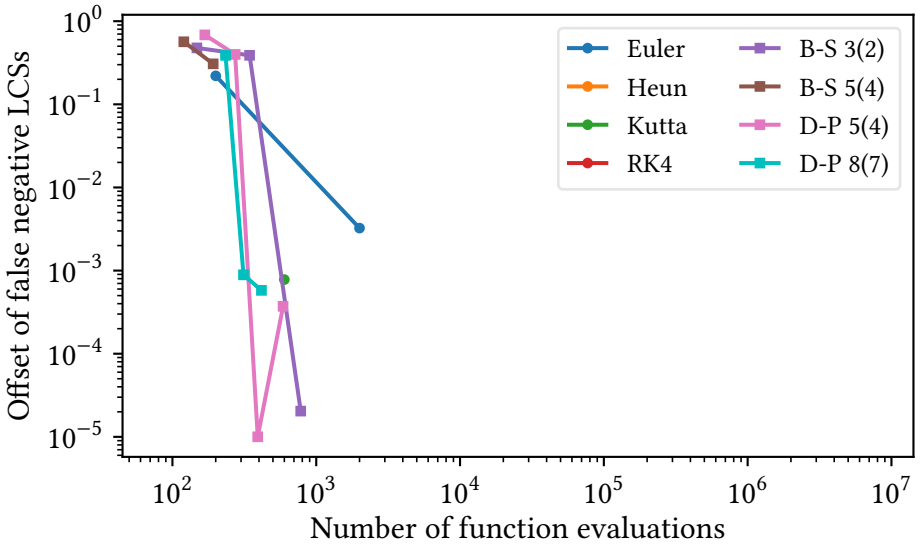


Figure 4.17: Offset of the LCS curve segments identified as false negatives as a function of the required number of function evaluations per advected tracer. The integration schemes for which no points are visible, yielded no false negatives. For the adaptive stepsize methods, the *average* number of function evaluations across all of the advected tracers is used as the abscissa.

Figures 4.18 and 4.19 indicate that the RMSD of the LCS curves, as defined in equation (3.21), does not follow the expected scalings of the numerical errors with the stepsize for the various singlestep integrators. Moreover, the two measures of the RMSD show the same quantitative behaviour, which is to be expected. The error quickly flattens for all integrators except the Euler scheme, which agrees well with the LCS curves presented in figures 4.2–4.4, where there are no visible discrepancies with regards to the reference LCS curve for step lengths smaller than 10^{-2}. The fact that the RMSD of the curves obtained by means of the Euler method appears to have flattened at a higher level than the rest of the integrators implies that the Euler method will never result in LCS curves with similar degrees of accuracy as for the other schemes. Lastly, the sudden drop in RMSD for the other methods for the transition $h = 10^{-4} \rightarrow h = 10^{-5}$ is unexpected. There is no clear reason for why it occurs. A similar effect is not present in the RMSDs of the flow map, eigenvalues nor eigenvectors, shown in figures figures 4.10, 4.12 and 4.14.

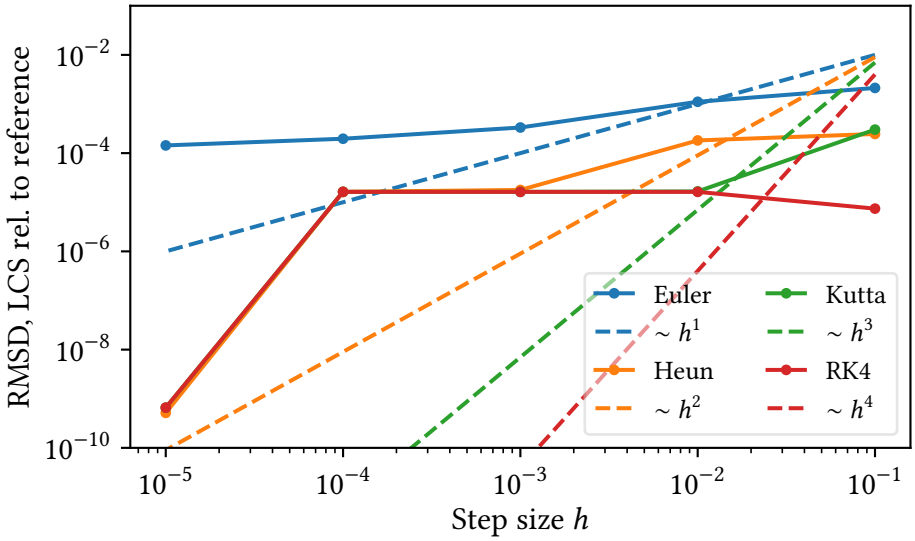


Figure 4.18: RMSD of the computed LCS curves relative to the reference as a function of numerical step length, for the singlestep methods considered. The dashed curves have slopes corresponding to the expected scalings with the step length for each method.

Figures 4.20 and 4.21 show the RMSD of the LCS curves for all numerical integration methods as a function of the required number of function evaluations per advected tracer. For the adaptive stepsize methods, the error decreases steadily with the number of function evaluations until a certain point, after which the error suddenly jumps to the same steady level as the (high order) singlestep methods. Like the sudden drop in the error for the singlestep methods with a large number of function evaluations (i.e., for very small time steps), there is no obvious reason why this jump occurs. Neither does a similar trend manifest itself in the RMSDs of the flow map, eigenvalues or -vectors, as shown in figures 4.11, 4.13 and 4.15. Finally, figures 4.20 and 4.21 suggest that the Bogacki-Shampine 5(4) and Dormand-Prince 8(7) schemes provide the overall most efficient means of obtaining accurate LCS curves for the considered double gyre system, in terms of the number of function evaluations needed to reach a given level of precision.

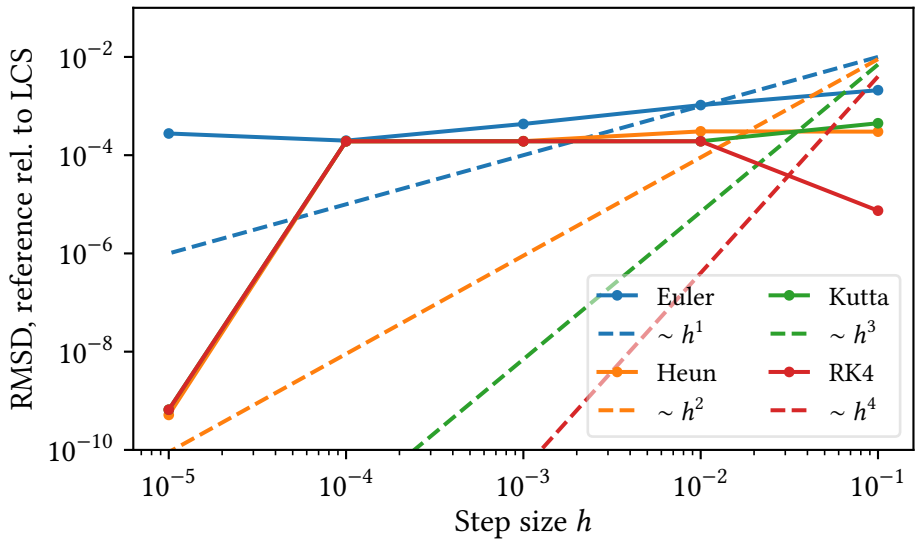


Figure 4.19: RMSD of the reference LCS relative to the computed LCS curves as a function of numerical step length, for the singlestep methods considered. The dashed curves have slopes corresponding to the expected scalings with the step length for each method.

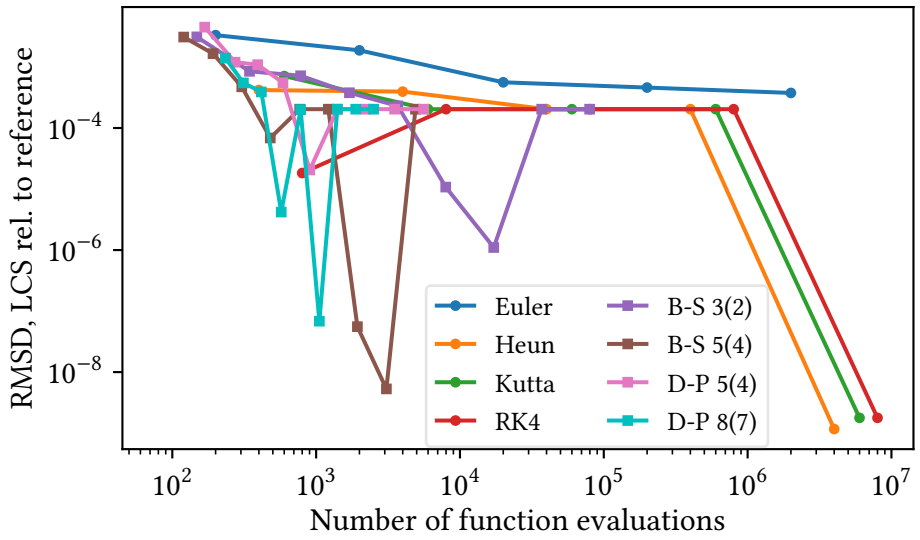


Figure 4.20: RMSD of the computed LCS curves relative to the reference as a function of of the required number of function evaluations per advected tracer. For the adaptive stepsize methods, the *average* number of function evaluations across all of the advected tracers is used as the abscissa.

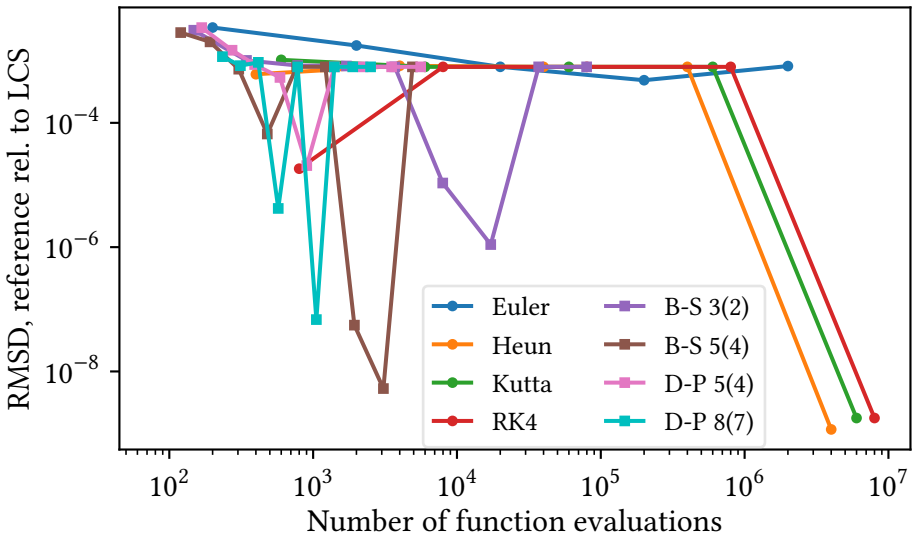


Figure 4.21: RMSD of the reference LCS relative to the computed LCS curves as a function of of the required number of function evaluations per advected tracer. For the adaptive stepsize methods, the *average* number of function evaluations across all of the advected tracers is used as the abscissa.

5 Discussion

5.1 General remarks

First off, the computed reference LCS, shown in figure 3.8, is made up of *seven* different strainline segments. The LCS presented in the article by Farazmand and Haller (2012) is claimed to consist of a *single* strainline segment. Comparing the two curves visually, however, indicates that the resulting LCSs are similar. Likewise, the domain $\mathcal{U}_0$, shown in figure 3.4, strongly resembles the one found by Farazmand and Haller. Nevertheless, the total number of points in the domain computed here is approximately two percent larger than what Farazmand and Haller found. These discrepancies could originate from different conventions in terms of generating the grid of tracers. Notably, Farazmand and Haller fail to provide a description of their approach.

Overall, the use of a variational framework for computing LCSs appears to produce robust and consistent LCS curves for all of the numerical integration schemes considered here, subject to usage of a sufficiently small numerical step length or tolerance level. This indicates that calculations of this kind are not particularly sensitive to integration method. Note that the $\mathcal{U}_0$ domains obtained by means of the adaptive stepsize methods for tol $= 10^{-1}$, shown in figure 4.5, differ greatly from the *correct* domain, which correlates well with the computed RMSD of the flow maps, shown in figure 4.11. In particular, the error in the flow map for the aforementioned tolerance level is of order 1 for all of the embedded methods. This error is comparable to the extent of the numerical domain, that is, $[0,\ 2] \times [0,\ 1]$. Naturally, this leads to a drastically different system.

The above observation implies that the most crucial part of the computation is advecting the tracers accurately. As mentioned in section 4.2.2, the error in the computed strain eigenvalues scales like the error in the computed flow map. This is to be expected, as the eigenvalues are essentially found by applying finite differences to the flow map. The eigenvalues are crucial in terms of identifying LCSs, as can be seen in the numerical reformulation of the LCS existence theorem, which is given in equation (3.13). The error in the computed strain eigendirections, however, is consistently negligible. This is most likely due to them being computed based on the *auxiliary* set of tracers, which per construction results in more accurate finite differences.

For the considered double gyre system, there appears to be a lower threshold in terms of the required advection accuracy, beneath which the computed LCS curves do not become more precise. This effect is apparent from inspection of figures 4.18–4.21, where the error of strainline components identified as LCS constituents flattens abruptly. Notably, this occurs for larger numerical step lengths or tolerance levels, respectively, than the corresponding turning points for the error in the flow maps. For the double gyre system considered here, it appears that this advection accuracy threshold is of the order 10^{-6}–10^{-7}, which follows from comparing figures 4.18–4.21 to figures 4.10 and 4.11. In particular, for flow maps with RMSD of 10^{-7} or lower, the RMSD of the LCS curves appears to not decrease further as the flow map precision increases.

However, because a similar flattening of the RMSD for the strain eigenvalues and eigenvectors is not apparent in figures 4.12–4.15, one may infer that this threshold is likely only valid for the LCS curves of this particular velocity field — that is, the system given by equations (3.1)–(3.3)— which appear quite robust. The same flow map accuracy threshold probably does not suffice for other, more volatile flow systems. Investigating this further for a wider range of systems could result in valuable insight. Should such a threshold be valid in general, it would naturally be of great significance when investigating generic transport systems by means of a similar variational LCS approach. Admittedly, there is no apparent reason why this should be the case.

The double gyre model considered in this project is obviously not representative of generic systems, in terms of the exact numerical step lengths or tolerance levels necessary in order to obtain correct LCSs with a certain degree of confidence. It does, however, indicate that these quantities should be chosen based on the considered system. For a fixed stepsize integration scheme, any single integration time step should not be so large that *too* much detail in the local and instantaneous velocity field is glossed over. Similar logic applies when adaptive stepsize methods are used, although it may be more difficult to enforce, depending on how the step length update is implemented. One possibility in terms of choosing the time step, is to find a characteristic velocity for the system, and choose the time step small enough so that a tracer moving with the characteristic velocity never traverses a distance greater than the grid spacing, when moving from one time level to the next.

When computing transport based on discrete data sets, such as snapshots of the instantaneous velocity fields in oceanic currents, spatial and temporal interpolation becomes necessary. Together with the inherent precision of the model data, the choice of interpolation scheme(s) sets a lower bound in terms of the accucacy with which tracers can be advected. For such cases, the interaction between the integration and interpolation schemes could be critical — both in terms of computation time and memory requirements, aside from the numerical precision. Independently of the scales at which well-resolved LCS information is sought in this kind of system, the aforementioned effects warrant further investigation.

5.2 On the incompressibility of the velocity field

As explained in section 3.1, the double gyre velocity field is incompressible per construction. Thus, per equation (2.18), the product of the strain eigenvalues should equal one. Regardless of the integration method used, however, this property was lost after approximately five units of time. This is not particularly surprising, because this property only really holds for infinitesimal fluid elements. Seeing as there is no assurance that neighboring tracers will remain nearby under the flow map, the finite difference approximation to the local stretch and strain could practically be rendered invalid. This issue is expected to be most prominent in regions of high local repulsion, which are precisely where accuracy is most imperative.

Thus, one should not really expect the property $\lambda_1\lambda_2 = 1$ to hold numerically, especially for

an indefinite time. However, sample tests indicate that the incompressibility property was preserved for longer when using a denser grid of tracers. In fact, the strain eigenvalue product was found to be unity beyond 20 units of time, that is, the integration time used in this project, for an initial grid spacing of 10^{-12}. However, due to the machine epsilon of double-precision floating point numbers being of order 10^{-16}, this would leave very few significant digits with which one could perform finite differencing, as mentioned in section 3.2.2. In the end, the grid spacings $\Delta x \simeq \Delta y \simeq 0.02$ and $\delta x = \delta y = 10^{-5}$ were chosen, because these were the resolutions at which repelling LCS curves for the double gyre system have previously been described in the literature, cf. Farazmand and Haller (2012). For further analysis, auxiliary grid spacings δx and δy of approximate order 10^{-7}–10^{-8} could be considered, when working with double-precision floating-point numbers. This would probably result in the incompressibility property being preserved for longer, while also leaving up to 7 or 8 significant digits for the finite difference approximation of the flow map, which is outlined in section 3.3.

An entirely different approach, suggested by Onu, Huhn, and Haller (2015), is to simply *define* the smaller strain eigenvalue as the reciprocal of the larger one, that is, $\lambda_1 \equiv \lambda_2^{-1}$. Computing the smaller eigenvalue in this way would influence the computation of LCSs through the condition (3.13a). This was not done here, primarily because Onu, Huhn, and Haller did not provide an illustration of their computed $\mathcal{U}_0$ domain, that is, a figure similar to figure 3.4; unlike Farazmand and Haller (2012), who did not follow the same approach as Onu, Huhn, and Haller. Thus, to our knowledge, following the convention $\lambda_1 = \lambda_2^{-1}$ would result in principally unverifiable LCSs. Moreover, as is evident from inspection of the numerical LCS existence conditions, given in equation (3.13), the computations are considerably more sensitive to λ_2 than λ_1, which suggests that accurate computations of LCSs for the system under consideration is not particularly susceptible to the numerical conservation of its inherent incompressibility.

5.3 Concerning the numerical representation of tracers

One way of increasing the numerical accuracy in the flow map, would be to use higher precision floating-point numbers to represent the tracer coordinates. The main drawback of making such a change, and using e.g. quadruple-precision floating-point numbers, is that these are, to this date, not fully implemented in conventional hardware — even that of the NTNU supercomputer, Vilje. Accordingly, quadruple-precision floating-point arithmetic is several orders of magnitude slower than that of double-precision floating-point numbers. Coupled with the increase in required memory (quadruple-precision numbers are stored using twice as many bits as double-precision numbers), one could be forced to perform several advections of smaller sets of tracers, calculating a piecewise representation of the overall flow map. However, should this prove impossible, using higher precision floating-point numbers would lead to a coarser representation of the flow map. Such an approach is perhaps most sensible for systems with velocity fields that are well-behaved, or even largely spatially invariant. Then again, in such cases, calculating LCSs have little practical relevance, seeing as the overall behaviour of the flow system could be estimated purely by advecting a smaller set of tracers, or even by simple inspection.

Rather than generating and advecting a fixed, large amount of tracer particles, another possible approach would be to use a set of fewer 'base' tracers, and making use of an *adaptive multigrid method*. A practical implementation could, for instance, involve the dynamic introduction of increasingly finer grids in regions where the local velocities would have the largest Euclidean norm. Such grids, however, would have to move *with* the flow, as the benefit over simply increasing the initial tracer density would diminish otherwise. The main reason why this was not implemented for this project, aside from it not being used in the literature, is that it in all likelyhood leads to inconsistencies when applying finite differences. That is, unless some sort of interpolation scheme was applied to the flow map. Seeing as this project is centered around integration methods, this idea was discarded. Regardless, this technique seems promising — at least on paper — which warrants further investigation.

5.4 Regarding the computation of strainlines

As outlined in section 3.4.1, a special sort of rectifying linear interpolation routine was implemented in order to eliminate local orientational discontinuities in the ξ_1 strain eigenvector field. The logical next step would be to consider a larger local subset of grid points, for instance, the 3×3 or 4×4 square of the 9 or 16 nearest neighbors, respectively, systematically reorientating the eigenvectors if necessary, and then using a higher order interpolation scheme in order to approximate the local strain eigenvector. This sort of generalization could make the strainline computation process more robust — although the linear approximation approach proved sufficient for the velocity field considered here, that may not be the case for more complex or volatile flow systems.

The classical Runge-Kutta method, with a numerical step length $\Delta = 10^{-3}$, was used for all computations of strainlines, regardless of advection integration scheme. This was a conscious choice, based on the idea of using the available information in the flow maps with the same degree of precision for all integrator configurations. If, however, the entire process of numerical integration, including both the advection of tracers and the calculation of strainlines, were to be performed with the same integrator configuration, comparisons between the resulting LCSs could have been misleading. That in, the strain information present in the flow maps would necessarily not have been utilized to the full extent, if the strainline integration was performed by means of an imprecise integration scheme, i.e., for large time steps or tolerance levels. For instance, the LCSs curves obtained by means of the Euler method, shown in figure 4.1, could have exhibited even more false positives and negatives if the same method was used in order to obtain strainlines as well, due to it only being 1^{st}-order accurate.

An alternative approach to the numerical integration of strainlines, would be to make use of a high order adaptive steplength method. This could, in principle, reduce the required computational time, in addition to reducing the impact of floating-point arithmetic error. However, in order to encapsulate the local strain dynamics accurately, the embedded automatic step size control should probably be more elaborate than the implementation outlined in section 3.2.3. In particular, it would be advisable to incorporate the local strainline curvature

somehow, in order to minimize deviations from the *true* trajectories. Furthermore, the use of a higher order strainline integration scheme might prove a fruitless exercise, unless a higher order strain eigenvector interpolation routine, as mentioned above, was implemented in tandem. This is because the effective accuracy of the strainline integration method depends strongly on the interpolation method, as evidenced by the reformulation of the basic strainline ODE given in equation (3.14) to the ODE which was used here, given in equation (3.15). As for the transport of tracers based on discrete velocity data, mentioned in section 5.1, the interaction between strain eigenvector interpolation method, and strainline integration method, seems like an interesting research topic for potential future endeavors.

5.5 The identification of strain maximizing strainlines

As mentioned in section 3.4.2, the lines in the set $\mathcal{L}$, used in order to find strainline segments satisfying the numerical LCS existence condition given in equation (3.13d) — regarding the identification of strainlines which serve as local maxima for $\overline{\lambda}_2$ — had to be selected with great care in order to accurately reproduce the LCS curve found in the literature. Although the LCS identification with the particular set of horizontal and vertical lines proved robust across all integration methods, the fact that only certain sets $\mathcal{L}$ were found conducive to appropriate strainline selection, is a damning indicament that the procedure as such is not particularly robust. Although not investigated here, an alternative approach could be to identify all strainlines which follow very similar trajectories, either manually or by means of some numerical clustering algorithm, then extracting the most strongly repellent strainline segments of each bunch as LCSs for the system.

The strainline tail end cutting procedure which was employed, briefly brought up in section 3.4.2 and illustrated in figure 3.5, is most certainly debatable. The reason it was considered in the first place, is that the specific wording used by Farazmand and Haller (2012) on the subject of comparing strainlines is somewhat ambiguous, in that they also describe that part of the process as comparison of *curve segments*. Moreover, they also state that a *part* of a strainline may qualify as an LCS. In addition, cutting the tail ends of strainlines which are stopped due to continious failures of one or more of the LCS conditions given in equation (3.13) could also, quite logically, be extended so that parts of the remaining strainline curve which, for example due to numerical noise, do not satisfy all of the aforementioned conditions, are excluded from the ensuing LCS identification. That is, those parts would not be considered when computing the $\overline{\lambda}_2$, the averaged λ_2 of the strainline segment, on the strainline as a whole, nor the strainline length. Effectively, this could result in a strainline being chopped into several shorter segments, further resulting in disjointed LCSs. As mentioned previously, the tail end cutting proved necessary in order to reproduce the LCS curve found by Farazmand and Haller (2012), meaning that this concept can, and most likely should, be investigated further.

Lastly, the rationale of filtering out LCS candidate curves which are shorter than the preselect length $l_f = 1$, described in section 3.4.2 and inspired by Farazmand and Haller (2012), was that excessively short LCSs are expected to have a negligible impact on the overall flow in the

system. However, Farazmand and Haller do not provide a justification for why they chose that particular filtering length. Another way to perform this sifting, would be to consider $\overline{\lambda}_2$ together with the length of the strainline segment. This could be done in such a way that, when selecting an LCS candidate from two strainlines, if one is somewhat longer but has a slightly smaller $\overline{\lambda}_2$, that is, is slightly less repelling than, the other, then the longer strainline is selected. This sort of routine should naturally be based upon sound mathematical logic. In short, there is a lot of room for research in terms of how to enforce the LCS condition given by equation (3.13d). The conditions given by equations (3.13a)–(3.13c) are quite unambigious, in comparison.

5.6 About the measures of error

RMS averaging was utilized for all the measures of error introduced in section 3.5, except the one concerning false positives and negatives with regards to LCS identification. Regarding the flow maps, this choice was made based on the fact that it encapsulates the effect of outliers, that is, any tracers for which the advection error becomes relatively large. These are expected to have the most severe impact on the ensuing LCS identification procedure. As shown in figures 4.10 and 4.11, the errors in the flow map resulting of the Euler method and all of the adaptive stepsize methods, for the largest numerical step length and tolerance level, respectively, are of order 1, similar to the dimensions of the computational domain, that is, $[0,\ 2] \times [0,\ 1]$. However, because there is no general a priori way of knowing the regions of the domain in which the flow maps will be resolved the most poorly, the maximum error is not of particular interest in this kind of analysis.

Because the normal component of the velocity field is zero at the domain boundaries, one expects the flow map error to, in some sense, be limited from above by the domain extent. Thus, one could argue that using the median error as the measure in such cases would be more appropriate. This is, however, clearly an artifact of the particular velocity field studied here. For a generic, compressible flow system, this need not be the case. In order to conform to the general case, and include the effect of outlier errors, the RMSD was chosen as the measure of error for all flow maps. Moreover, given that the error in the flow maps follows some statistical distribution — the details of which was not investigated in this project — the RMSD can be viewed as a first approximation of the standard deviation.

Regarding the errors in the strain eigenvalues, one could argue that using *relative* errors would be sensible. However, the property given in equation (2.18), that is, $\lambda_1\lambda_2 = 1$, was not preserved for more than a few units of time for the flow system considered here, as mentioned in section 5.2. The same property does not hold for general, compressible flows. This could result in some very small eigenvalues, which would yield deceptively large relative errors. This would, again, be undesirable, because one would expect the eigenvalue errors to scale similarly to that of the flow map, due to the computation of the Cauchy-Green strain tensor field being based on finite differencing applied to the flow map, as described in section 3.3. Thus, the RMSD became the method of choice in computing the error of the strain eigenvalues. Moreover, seeing as the azimuthal angles of the strain eigenvectors are calculated relative to

what is really an arbitrary axis, as far as the overall flow patterns are concerned, the *absolute* errors in the strain eigendirections is the only sensible measure. Accordingly, the RMSD was chosen as the measure of error for the directions of all strain eigenvectors, too.

Lastly, regarding the estimation of the offset of false positive and false negative LCS curve segments, there certainly exists other relevant measures. An important aspect of any such principle, however, is the elimination of subjective assessment, for instance, one should not have to identify what LCS curve segment a false positive most closely resembles by means of inspection. Furthermore, the offset of false positives and negatives is not an independent error measure — it depends intrinsically on the errors in the strain eigenvalue and eigenvector fields. As such, an interesting approach could be to consider the entire set of computed strainlines, employing some means of identifying the curve segments which resemble the reference LCS curves the most, and examining the reason for which any false negatives are eliminated, or why any false positives are included, in the final picture. This sort of approach could provide useful insight into the nature of LCSs, but was ultimately not pursued in this project, as it simply fell outside the overall scope of this project, that is, the dependence of LCSs on the underlying numerical integration scheme.

6 Conclusions

For the double gyre system considered here, the calculation of its LCSs does not exhibit particularly sensitive dependence to the choice of numerical integration method. However, this could be a consequence of the LCSs present within the system being robust under the chosen parameter values. This view is supported by the fact that the errors in the computed LCS curves quickly flattened for sufficiently small integration time steps or tolerance levels, while a similar effect did not manifest itself for the computed flow maps or strain eigenvalues. Based upon the errors in the computed flow maps, however, high order integration methods are generally advisable for more generic flow systems — they result in more efficient calculations, which are less susceptible to numerical round-off error. Moreover, the fact that the same strainline segments were identified as LCSs even for quite large errors in the strain eigenvalues, suggests that the numerical implementation of the variational principles in order to find LCSs is, in itself, robust.

There is, however, room for further research with regards to the numerical implementation of one of the LCS existence conditions, derived from their variational theory. In particular, to our knowledge, a general, robust numerical implementation is yet to be described in the literature. The approach used in this project involves a set of parameters independent of the overall flow. These parameter values were chosen based upon careful inspection of the system under consideration, in order for the LCS curves obtained here to conform with those found in the literature. A suggested alternative approach, which was not investigated as part of this project, would be to utilize a sort of numerical clustering algorithm rather than resorting to similar (to some extent) subjective considerations to the ones which were employed here.

Although no numerical integration scheme stood out as superior in general, using higher order methods invariably resulted in more efficient calculations, which are less prone to numerical round-off errors. Accordingly, the use of higher order methods is generaly advisable. Equally important as the choice of integration scheme, however, is the choice of numerical step length or tolerance level. Ideally, this should be selected based upon physical considerations of the system at hand. For instance, one could identify a characteristic velocity, then tune the step length or tolerance level such that no tracers moving with said velocity ever moves further than a characteristic grid spacing, when moving from one time step to the next. This, in order to ensure that the local instantaneous dynamics are resolved properly. To what scale the microscopic behaviour should be resolved depends on what scale to which detailed information regarding the flow is sought. Regarding most real-life applications, this is also dictated by the the sampling frequency — or, more likely, model output — of the discrete data samples (spatial and temporal alike).

On that note, when investigating transport systems for which the available data sets are discrete, the choice of numerical *interpolation* scheme will generally also impact the calculations of LCSs. This depencence has not been investigated as part of this project. However, similar reservations as for the integration time step or tolerance levels are also applicable to the spatial

and temporal sampling frequencies involved in the interpolation. Furthermore, the choice of integration scheme should be made in relation to the interpolation method. For instance, it makes little sense to use a 5^{th}-order accurate integration scheme in tandem with a 3^{rd}-order accurate interpolator. Put simply, the interaction between interpolation and integration schemes in the analysis of discrete data sets is of great interest for practical applications of the LCS theory, and warrants further investigation.

Suggestions for further work

A plethora of numerical integation methods were not tested as a part of this project. An entirely different class of ODE solvers to the ones considered here, namely linear multistep methods, could have various desirable properties. In particular, this type of methods have memory, thus, they function quite differently to the memoryless solvers considered here. Interestingly, both fixed and adaptive stepsize linear multistep methods exist, and, by means of known recurrence relations, one may in principle design multistep methods of arbitrary order (Hairer, Nørsett, and Wanner 1993, chapter III). Research with regards to the usefulness of such methods in the context of LCS detection — in addition to investigations on robust, numerical implementation of the as yet somewhat ambiguous LCS existence condition — could reasonably be conducted based upon the double gyre system examined in this project.

For my own future work, I think that it would be natural to continue the study of LCSs. In particular, I find the prospect of analyzing LCSs in three-dimensional flow systems alluring. To my knowledge, the three-dimensional formulation of the variational LCS approach has not yet been explored in the literature. Thus, this could potentially accelerate the recommendability of the use of LCS theory to describe systems which can not reasonably be regarded as two-dimensional. Moreover, examining a broader range of systems seems like a logical extension of the work I have conducted as part of this project. Lastly, as suggested previously, investigations with regards to the interaction between numerical integration and interpolation schemes in the context of LCS detection are appealing, due to their innate relation to real-world applications.

References

Adams, R. and Essex, C. (2010). *Calculus: A Complete Course*. 7th ed. Pearson, Toronto. ISBN: 978-0-321-54928-0.

Ali, S. and Shah, M. (2007). "A Lagrangian Particle Dynamics Approach for Crowd Flow Segmentation and Stability Analysis". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–6. ISSN: 1063-6919.

Bogacki, P. and Shampine, L. (1989). "A 3(2) Pair of Runge-Kutta Formulas". In: *Applied Mathematics Letters* 2.4, pp. 321–325. ISSN: 0893-9659.

Bogacki, P. and Shampine, L. (1996). "An Efficient Runge-Kutta (4, 5) Pair". In: *Computers & Mathematics with Applications* 32.6, pp. 15–28. ISSN: 0898-1221.

Dormand, J. and Prince, P. (1980). "A family of embedded Runge-Kutta formulae". In: *Journal of Computational and Applied Mathematics* 6.1, pp. 19–26. ISSN: 0377-0427.

Farazmand, M. and Haller, G. (2011). "Erratum and addendum to 'A variational theory of hyperbolic Lagrangian coherent structures' [Physica D 240 (2011) 547–598]". In: *Physica D: Nonlinear Phenomena* 241.4, pp. 439–441. ISSN: 0167-2789.

Farazmand, M. and Haller, G. (2012). "Computing Lagrangian coherent structures from their variational theory". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.1, p. 013128. ISSN: 1054-1500.

Hairer, E., Nørsett, S. P., and Wanner, G. (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems*. 2nd ed. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-56670-0. Corrected 3rd printing, 2008.

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. 2nd ed. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-05221-7. Corrected 2nd printing, 2002.

Haller, G. and Yuan, G. (2000). "Lagrangian coherent structures and mixing in two-dimensional turbulence". In: *Physica D: Nonlinear Phenomena* 147.3, pp. 352–370. ISSN: 0167-2789.

Haller, G. (2010). "A variational theory of hyperbolic Lagrangian Coherent Structures". In: *Physica D: Nonlinear Phenomena* 240.7, pp. 547–598. ISSN: 0167-2789.

Institute of Electrical and Electronics Engineers (2008). "IEEE Standard for Floating-Point Arithmetic". In: *IEEE Std 754-2008*, pp. 1–70. ISBN: 978-0-7381-5752-8.

Lekien, F. and Ross, S. D. (2010). "The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20.1. ISSN: 1054-1500.

Olascoaga, M. et al. (2008). "Tracing the early development of harmful algal blooms on the West Florida Shelf with the aid of Lagrangian coherent structures". In: *Journal of Geophysical Research: Oceans* 113.C12. ISSN: 0148-0227.

Onu, K., Huhn, F., and Haller, G. (2015). "LCS Tool: A computational platform for Lagrangian coherent structures". In: *Journal of Computational Science* 7, pp. 26–36. ISSN: 1877-7503.

Prince, P. and Dormand, J. (1981). "High order embedded Runge-Kutta formulae". In: *Journal of Computational and Applied Mathematics* 7.1, pp. 67–75. ISSN: 0377-0427.

Shadden, S. C., Lekien, F., and Marsden, J. E. (2005). "Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows". In: *Physica D: Nonlinear Phenomena* 212.3, pp. 271–304. ISSN: 0167-2789.

Strogatz, S. H. (2014). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview press, Colorado. ISBN: 978-0-813-34910-7.