

# Sensitivity to Numerical Integration Scheme in Calculation of Transport Barriers

Arne Magnus Tveita Løken

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

November 30, 2017



# ***Abstract***

---



# Sammendrag

---



## Preface

---





# Table of Contents

---

<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xii</b>
<b>Notation</b>	<b>xiv</b>
<b>1 Introduction and Motivation</b>	<b>1</b>
<b>2 Theory</b>	<b>2</b>
2.1 Solving systems of ordinary differential equations . . . . .	2
2.1.1 The Runge-Kutta family of numerical methods . . . . .	2
2.1.2 The Runge-Kutta methods under consideration . . . . .	5
2.2 The type of flow systems considered . . . . .	11
2.3 Definition of Lagrangian Coherent Structures for two-dimensional flows	12
2.3.1 Hyperbolic LCSs . . . . .	13
2.4 FTLE fields as predictors for LCSs . . . . .	14
<b>3 Method</b>	<b>16</b>
3.1 The double gyre model . . . . .	16
3.2 Advecting a set of initial conditions . . . . .	17
3.2.1 Generating a set of initial conditions . . . . .	17
3.2.2 On the choice of numerical step lengths and tolerance levels . .	19
3.2.3 On the implementation of embedded Runge-Kutta methods . .	20
3.3 Calculating the Cauchy-Green strain tensor . . . . .	21
3.4 Identifying LCS candidates numerically . . . . .	22
3.4.1 A framework for computing smooth strainlines . . . . .	23
3.4.2 Extracting hyperbolic LCSs from strainlines . . . . .	24
3.5 Estimation of errors . . . . .	30
<b>4 Results and Discussion</b>	<b>32</b>
4.1 The LCS curves obtained using the different schemes . . . . .	32
4.1.1 LCS curves stemming from singlestep methods . . . . .	32
4.1.2 LCS curves stemming from adaptive stepsize methods . . . . .	34
4.2 Measures of error . . . . .	38
4.2.1 Computed deviations in the LCS curves . . . . .	38
4.2.2 Computed deviations in the flow maps . . . . .	42
4.2.3 Computed deviations in the strain eigenvalues and -vectors . .	44
4.3 General remarks . . . . .	47
4.4 On the incompressibility of the velocity field . . . . .	48
4.5 Concerning the numerical representation of tracers . . . . .	49
4.6 About the computation of strainlines . . . . .	49

4.6.1	The special linear interpolation of strain eigendirections . . . .	49
4.6.2	The use of a single numerical integrator . . . . .	49
4.6.3	The identification process of local strain maximizing strainlines	49
<b>5</b>	<b>Conclusions</b>	<b>53</b>
	<b>References</b>	<b>54</b>
<b>APPENDIX A</b>	<b>Haller er en dust</b>	<b>55</b>



## List of Figures

---

2.1	Geometric interpretation of the eigenvectors of the Cauchy-Green strain tensor	12
3.1	Illustration of the set of initial conditions . . . . .	18
3.2	Illustration of the concept of auxiliary tracers . . . . .	19
3.3	Illustration of the special linear interpolation used for the $\xi_1$ eigenvector field	23
3.4	The set $\mathcal{U}_0$ for the double gyre system . . . . .	25
3.5	The identification process of strainlines which are local strain maximizers . .	27
3.6	The repelling reference LCS of the double gyre system . . . . .	28
3.7	Plots of the FTLE field and $\lambda_2(\mathbf{x}_0)$ distribution of the double gyre system . . .	29
4.1	LCS curves found by means of the Euler integration scheme . . . . .	32
4.2	LCS curves found by means of the Heun integration scheme . . . . .	33
4.3	LCS curves found by means of the Kutta integration scheme . . . . .	33
4.4	LCS curves found by means of the classical Runge-Kutta integration scheme .	34
4.5	The $\mathcal{U}_0$ domains obtained with the adaptive stepsize integration schemes, with numerical tolerance level $\text{tol} = 10^{-1}$ . . . . .	35
4.6	LCS curves found by means of the Bogacki-Shampine 3(2) integration scheme	36
4.7	LCS curves found by means of the Bogacki-Shampine 5(4) integration scheme	36
4.8	LCS curves found by means of the Dormand-Prince 5(4) integration scheme .	37
4.9	LCS curves found by means of the Dormand-Prince 8(7) integration scheme .	37
4.10	Offset of the LCS curve segments identified as false positives, as a function of the required number of function evaluations . . . . .	39
4.11	Offset of the LCS curve segments identified as false negatives, as a function of the required number of function evaluations . . . . .	39
4.12	RMSD of the computed LCS curves relative to the reference as a function of numerical step length, for the singlestep methods considered . . . . .	40
4.13	RMSD of the reference LCS relative to the computed LCS curves as a function of numerical step length, for the singlestep methods considered . . . . .	41
4.14	RMSD of the computed LCS curves relative to the reference as a function of the required number of function evaluations . . . . .	41
4.15	RMSD of the reference LCS relative to the computed LCS curves relative a function of the required number of function evaluations . . . . .	42
4.16	RMSD of the flow maps as a function of numerical step length, for the singlestep methods considered . . . . .	43
4.17	RMSD of all flow maps as a function of the required number of function evaluations	43
4.18	RMSD of the $\lambda_2$ field as a function of numerical step length, for the singlestep methods considered . . . . .	44
4.19	RMSD of the $\lambda_2$ field as a function of the required number of function evaluations	45
4.20	RMSD of the strain eigendirections as a function of numerical step length, for the singlestep methods considered . . . . .	46
4.21	RMSD of the strain eigendirections as a function of the required number of function evaluations . . . . .	46



## List of Tables

---

2.1	Butcher tableau representation of a general $s$ -stage Runge-Kutta method . . .	3
2.2	Butcher tableau representation of general, embedded, explicit Runge-Kutta methods . . . . .	4
2.3	Butcher tableau representation of the explicit Euler method . . . . .	5
2.4	Butcher tableau representation of the explicit Heun method . . . . .	5
2.5	Butcher tableau representation of the explicit Kutta method . . . . .	6
2.6	Butcher tableau representation of the explicit, classical Runge-Kutta method .	6
2.7	Butcher tableau representation of the Bogacki-Shampine 3(2) embedded Runge-Kutta method . . . . .	7
2.8	Butcher tableau representation of the Bogacki-Shampine 5(4) embedded Runge-Kutta method . . . . .	8
2.9	Butcher tableau representation of the Dormand-Prince 5(4) embedded Runge-Kutta method . . . . .	9
2.10	Butcher tableau representation of the Dormand-Prince 8(7) embedded Runge-Kutta method . . . . .	10



## Notation

---

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

$$\dot{f}(t) \equiv \frac{df(t)}{dt}$$

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

$$\boldsymbol{\xi} = (\xi_1, \xi_2, \dots, \xi_n)$$

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

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

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

$$\boldsymbol{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}$$





# 1 Introduction and Motivation

- Lagrangian coherent structure: A structure that can separate dynamically distinct invariant regions. Invariant regions: All trajectories starting out within it, remain within the region, although the region itself may move and deform with time.
- Lagrangian coherent structure: Landscapes in multidimensional landscapes, shaping the flow patterns in dynamical systems.
- Motivation: Complex systems, i.e., many-particle nonlinear systems: Need computational shortcuts
- LCS: Principally robust structures which enable us to make predictions regarding the future states of a flow system.
- Examples of real-world application:
  - Population growth
  - Predicting the advection of particles by means of oceanic currents or wind
    - Recent examples: The volcanic eruptions at Eyafjallajökul, and at Bali (2017)
  - Predicting where the remnants of an oil spill will end up → Where to focus the rescue operation in the short and medium term

## 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 a differential equation of the form

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

where  $x$  is an unknown function (scalar or vector) of time  $t$ . The function  $f$  is defined on an open subset  $\Omega$  of  $\mathbb{R} \times \mathbb{R}^n$ . 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 (i.e.,  $n > 1$ ) the differential equation (2.1) is replaced by a family of equations

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

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.

#### 2.1.1 The Runge-Kutta family of numerical methods

For nonlinear systems, analytical solutions frequently do not exist. Thus, such systems are often analyzed by means of numerical methods. In numerical analysis, the Runge-Kutta family of methods are 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  $a_{1,1}, a_{1,2}, \dots, a_{1,s}, a_{2,1}, a_{2,2}, \dots, a_{2,s}, \dots, a_{s,1}, a_{s,2}, \dots, a_{s,s}, b_1, b_2, \dots, b_s$  and  $c_1, c_2, \dots, c_s$  be real coefficients. Let  $h$  be the numerical step length used in the temporal discretization. Then, the method

$$\begin{aligned} k_i &= f\left(t_n + c_i h, x_n + h \sum_{j=1}^s a_{i,j} k_j\right), \quad i = 1, \dots, s \\ x_{n+1} &= x_n + h \sum_{i=1}^s b_i k_i \end{aligned} \quad (2.3)$$

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

The main reason to include multiple stages  $s$  in a Runge-Kutta method, is to improve the numerical accuracy of the computed solutions. Hairer, Nørsett, and Wanner (1993, p.134) define the *order* of a Runge-Kutta method as follows:

**Definition 2.** A Runge-Kutta method (2.3) is said to be of *order*  $p$  if, for sufficiently smooth systems (2.1),

$$\|x_{n+1} - x(t_{n+1})\| \leq Kh^{p+1}, \quad (2.4)$$

where  $K$  is a numerical constant, i.e., if the Taylor series for the exact solution  $x(t_{n+1})$  and the numerical solution  $x_{n+1}$  coincide up to (and including) the term  $h^p$ .

It is easy to show that if the local error of a Runge-Kutta method is of order  $p + 1$ , the global error, i.e., the total accumulated error resulting of applying the algorithm a number of times, is expected to scale as  $h^p$ . Showing this is left as an exercise for the interested reader.

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 (2.3) 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. Unlike explicit methods, implicit methods require the solution of a linear system at every time level, making them more computationally demanding than their explicit siblings. 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 highly numerically unstable, unless the numerical step size is made exceptionally small, rendering most explicit methods practically useless. For *nonstiff* systems,

however, implicit methods behave similarly to their explicit analogues in terms of numerical accuracy and convergence properties.

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  $\hat{x}_{n+1}$ . 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. 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.

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

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

For embedded methods, the coefficients are tuned such that

$$x_{n+1} = x_n + h \sum_{i=1}^s b_i k_i \quad (2.5a)$$

is of order  $p$ , and

$$\widehat{x}_{n+1} = x_n + h \sum_{i=1}^s \widehat{b}_i k_i \quad (2.5b)$$

is of order  $\widehat{p}$ , typically with  $\widehat{p} = p \pm 1$ .

### 2.1.2 The Runge-Kutta methods under consideration

Naturally, an abundance of Runge-Kutta methods exist. 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, I consider two classes of explicit Runge-Kutta methods, namely singlestep and adaptive stepsize methods. From both classes, I 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<sup>st</sup> order accurate, and requires a single function evaluation 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 2.3. 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<sup>nd</sup> 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<sup>rd</sup> 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<sup>th</sup> 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 (i.e., 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. 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. This is readily apparent from their Butcher tableaus, where the  $b$  coefficients correspond exactly with the last row the Runge-Kutta matrix. The *First Same As Last* property reduces the computational cost of a successive step.

**Table 2.7:** Butcher tableau representation of the Bogacki- Shampine 3(2) embedded Runge-Kutta method. The  $b$  coefficients correspond to a 3<sup>rd</sup> order accurate solution used to continue the integration. The  $\hat{b}$  coefficients correspond to a 2<sup>nd</sup> 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<sup>th</sup> order accurate solution used to continue the integration. The two rows of  $\hat{b}$  coefficients correspond to two independent 4<sup>th</sup> 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<sup>th</sup> order method (Hairer, Nørsett, and Wanner 1993, p.194 in the 2008 printing). 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<sup>th</sup> order accurate solution used to continue the integration. The  $\hat{b}$  coefficients correspond to a 4<sup>th</sup> 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 Hairer, Nørsett, and Wanner (1993, p.178 in the 2008 printing).

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	$\frac{7571}{16695}$	$\frac{393}{640}$	$\frac{-92097}{339200}$	$\frac{187}{2100}$
						$\frac{1}{40}$



## 2.2 THE TYPE OF FLOW SYSTEMS CONSIDERED

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

$$\dot{\mathbf{x}} = \mathbf{v}(t, \mathbf{x}), \quad \mathbf{x} \in \mathcal{U}, \quad t \in [t_0, t_1], \quad (2.6)$$

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 (2.6), may or may not be computed. 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 (2.6), the flow map is defined as

$$\mathbf{F}_{t_0}^t(\mathbf{x}_0) = \mathbf{x}(t; t_0, \mathbf{x}_0), \quad (2.7)$$

i.e., the flow map describes the mathematical movement of the tracers from one point in time (e.g. the initial condition) to another. Generally, the flow map is as smooth as the velocity field  $\mathbf{v}$  in system (2.6) (Farazmand and Haller 2012). For sufficiently smooth velocity fields, the right Cauchy-Green strain tensor field is defined as

$$\mathbf{C}_{t_0}^t(\mathbf{x}_0) = \left( \nabla \mathbf{F}_{t_0}^t(\mathbf{x}_0) \right)^* \nabla \mathbf{F}_{t_0}^t(\mathbf{x}_0), \quad (2.8)$$

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

$$(\nabla \mathbf{f})_{i,j} = \frac{\partial f_i}{\partial x_j} \quad (2.9)$$

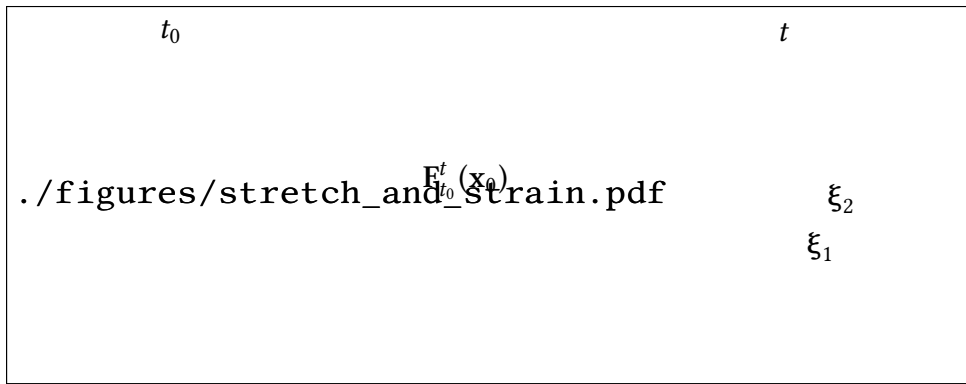
which, for our two-dimensional flow, reduces to

$$\nabla \mathbf{f} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix}. \quad (2.10)$$

By construction, the Cauchy-Green strain tensor  $C_{t_0}^t$  is symmetric and positive definite. Thus, it has two real, positive eigenvalues and orthogonal, real eigenvectors. Its eigenvalues  $\lambda_i$  and corresponding unit eigenvectors  $\xi_i$  are defined by

$$\begin{aligned} C_{t_0}^t(\mathbf{x}_0)\xi_i(\mathbf{x}_0) &= \lambda_i\xi_i(\mathbf{x}_0), \quad \|\xi_i(\mathbf{x}_0)\| = 1, \quad i = 1, 2, \\ 0 &< \lambda_1(\mathbf{x}_0) \leq \lambda_2(\mathbf{x}_0), \end{aligned} \quad (2.11)$$

where, for the sake of notational transparency, the dependence of  $\lambda_i$  and  $\xi_i$  on  $t_0$  and  $t$  has been suppressed. The geometric interpretation of equation (2.11) is that a fluid element undergoes the most stretching along the  $\xi_2$  axis, and the least along the  $\xi_1$  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  $F_{t_0}^t(\mathbf{x}_0)$ . The local stretching is the largest in the direction of  $\xi_2$ , the eigenvector which corresponds to the largest eigenvalue  $\lambda_2$  of the Cauchy-Green strain tensor, defined in equation (2.11). Along the  $\xi_2$  and  $\xi_1$  axes, the stretch factors are  $\sqrt{\lambda_2}$  and  $\sqrt{\lambda_1}$ , respectively.

Because the stretch factors along the  $\xi_1$  and  $\xi_2$  axes are given by the square root of the corresponding eigenvalues, for incompressible flow, the eigenvalues satisfy

$$\lambda_1(\mathbf{x}_0)\lambda_2(\mathbf{x}_0) = 1 \quad \forall \mathbf{x}_0 \in \mathcal{U} \quad (2.12)$$

where, for our case, incompressibility is equivalent to the velocity field  $\mathbf{v}$  being divergence-free (i.e.,  $\nabla \cdot \mathbf{v} \equiv 0$ ). This is due to the tracer particles being massless, per definition.

## 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 use 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 (2.8), from which the aforementioned conditions follow. The immediately relevant parts of Haller’s theory are summarized in definitions 3–6.

**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 (2.7), 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 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:

**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  $[t_0, t_0 + T]$ .  $\mathcal{M}(t)$  is a repelling LCS over  $[t_0, t_0 + T]$  if and only if all the following hold for all initial conditions  $\mathbf{x}_0 \in \mathcal{M}(t_0)$ :*

$$\lambda_1(\mathbf{x}_0) \neq \lambda_2(\mathbf{x}_0) > 1 \quad (2.13a)$$

$$\langle \xi_2(\mathbf{x}_0), \mathbf{H}_{\lambda_2}(\mathbf{x}_0) \xi_2(\mathbf{x}_0) \rangle < 0 \quad (2.13b)$$

$$\xi_2(\mathbf{x}_0) \perp \mathcal{M}(t_0) \quad (2.13c)$$

$$\langle \nabla \lambda_2(\mathbf{x}_0), \xi_2(\mathbf{x}_0) \rangle = 0 \quad (2.13d)$$

In theorem 1,  $\langle \cdot, \cdot \rangle$  denotes the Euclidean inner product, and  $\mathbf{H}_{\lambda_2}$  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

$$(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}, \quad (2.14)$$

which, for our two-dimensional flow, reduces to

$$\mathbf{H}_f = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} \quad (2.15)$$

Condition (2.13a) ensures that the normal repulsion rate is larger than the tangential stretch due to shear strain along the LCS, as per definition 3. Conditions (2.13c) and (2.13d) suffice to ensure that the normal repulsion rate attains a local extremum along the LCS, relative to all nearby material lines. Lastly, condition (2.13b) 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

$$\sigma(\mathbf{x}_0) = \frac{\ln(\lambda_2(\mathbf{x}_0))}{2|T|}, \quad (2.16)$$

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. As per definition 5, attracting LCSs are identified as repelling LCSs in backwards 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 FTLE field is 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-order 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 the dependence of LCS identification by means of the variational approach as presented in section 2.3.1 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 solid, as in impenetrable, wall which oscillates periodically, causing the vortices to contract and expand periodically. In terms of the cartesian coordinate vector  $\mathbf{x} = (x, y)$ , the system can be expressed mathematically as

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

where

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

and the parameters  $A$ ,  $\epsilon$  and  $\omega$  dictate the nature of the flow pattern. As in the literature, the parameter values

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

were used (Farazmand and Haller 2012; Shadden, Lekien, and Marsden 2005). Moreover, the starting time was  $t_0 = 0$ , and the integration time was  $T = 20$ , i.e., forcing two periods of motion, per (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:

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

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

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

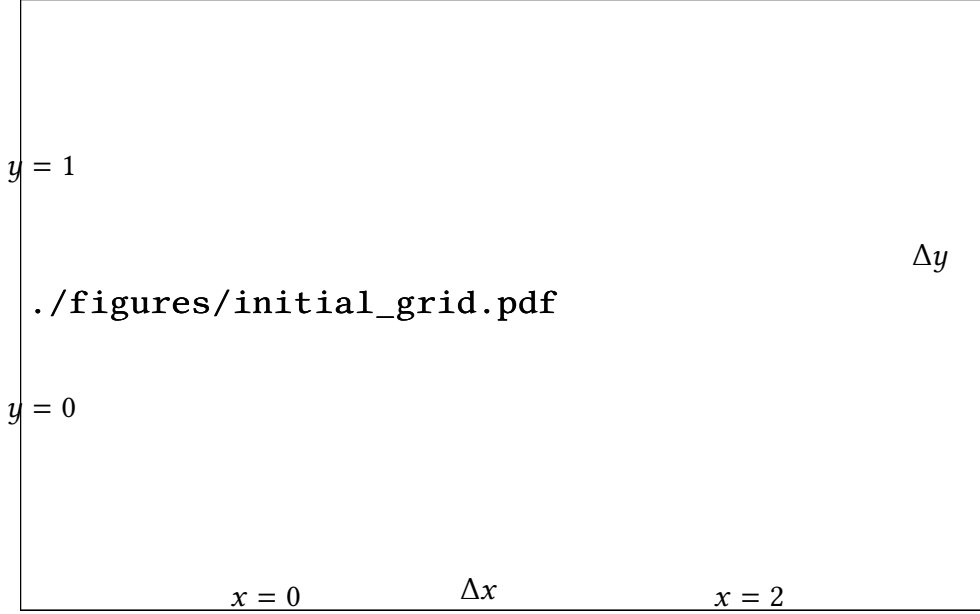
where the latter equality follows from Schwartz' theorem of mixed partial derivatives, as the stream function is smooth. This means that we expect the property given in equation (2.12) to hold for the double gyre flow.

## 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). The system has no known 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, some of which are 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 linearly spaced tracers, with  $1000 \times 500$  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 artificially, with an additional two rows or columns appended to all of the domain edges, with the same grid spacing as the *main* grid. This was done in order to ensure that the dynamics at the domain boundaries were included in the analysis to follow. The extended grid thus had a total of  $1004 \times 504$  grid points. The construction of the grid is illustrated in figure 3.1.



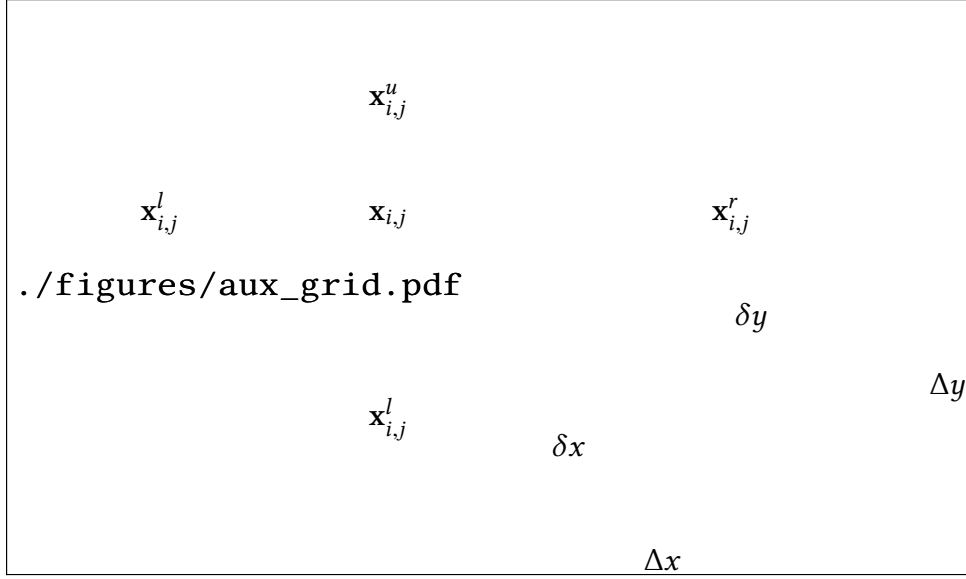
**Figure 3.1:** Illustration of the set of initial conditions. Dark grey blobs signify the main tracers, i.e., the tracers which discretize the computational domain  $[0, 2] \times [0, 1]$ . These were linearly spaced in either direction, with twice as many points in the  $x$ -direction as the  $y$ -direction, in order to generate an approximately equidistant grid. Light grey blobs signify the artificially extended grid, i.e., tracers starting originating outside of the computational domain. These were used in order to properly encapsulate the dynamics at the domain boundaries, in the analysis to follow.

In order to increase the resolution of the Cauchy-Green strain tensor, it is necessary to increase the accuracy with which one computes the Jacobian of the flow map, as can be seen from equation (2.8). This was done by advecting a set of auxiliary tracer points surrounding each main point. To each tracer point  $\mathbf{x}_{i,j} = (x_i, y_j)$ , neighboring points defined as

$$\begin{aligned} \mathbf{x}_{i,j}^r &= (x_i + \delta x, y_j), & \mathbf{x}_{i,j}^l &= (x_i - \delta x, y_j) \\ \mathbf{x}_{i,j}^u &= (x_i, y_j + \delta y), & \mathbf{x}_{i,j}^d &= (x_i, y_j - \delta y) \end{aligned} \quad (3.6)$$

were assigned, where  $\delta x$  and  $\delta y$  are increments smaller than the grid spacings  $\Delta x \approx \Delta y$ . 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  $\delta x$  and  $\delta y$ . 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 accurately represented by floating-point numbers, however, there is a strict lower limit to which it makes sense to lower the values of  $\delta x$  and  $\delta y$ . In particular, the smallest number which can be resolved as a double-precision floating-point number is of the order  $10^{-16}$ . When decreasing the auxiliary grid spacing, the increase in precision is quickly offset by the fact that one automatically gets allocated



**Figure 3.2:** Illustration of the concept of auxiliary tracers, used in order to compute the Jacobian of the flow map, and by extension, the Cauchy-Green strain tensor field, given by equation (2.8), more accurately. Grey blobs represent the original tracers, whereas white blobs represent the auxiliary ones.

a smaller number of decimal digits with which one calculate the discrete approximation of the derivatives involved in the Jacobian. This is due to the double gyre velocity field, being reasonably well-behaved, leading most tracers which are initially 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 — three orders of magnitude smaller than the original grid spacing, ensuring that the derivatives in the Jacobian are far more well-resolved than for the main tracers, while also leaving approximately 10 decimal digits for which there can be a difference in the final positions of the auxiliary tracers.

### 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 the following: For a step length of  $10^{-5}$ , the total number of integration steps required in order to take the system from  $t = 0$  to  $t = 20$  is of order  $10^6$ . As previously mentioned, the inherent accuracy of double-precision floating point numbers is of order  $10^{-16}$ . Thus, the total floating point error expected to arise when integrating with a step length of  $10^{-5}$  is of order  $10^{-10}$ .

The least accurate of the fixed stepsize integrators under consideration, the Euler method, is presented in table 2.3. It is 1<sup>st</sup> order accurate globally, meaning that its local error is of 2<sup>nd</sup> order, per definition 2. Thus, we expect that the local error of the Euler method, for a step length of  $10^{-5}$ , is of order  $10^{-10}$ , that is, the same order as the accumulated floating-point errors. Reducing the time step further necessarily leads to an increase in the accumulated floating-point errors, meaning that we cannot reasonably expect to resolve the positions from one step to the next more accurately, for the Euler method. At the very least, a time step of

$10^{-5}$  appears to represent a point after which there is little to be gained in terms of increased numerical accuracy for the Euler method. For the other fixed stepsize integrators, which are of higher order, we expect this breaking point to occur for a somewhat larger time step.

While the above logic does not translate directly for the adaptive stepsize integrators, 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 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 ran on NTNU's supercomputer, Vilje. 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 computational walltime 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  $\hat{x}_1$ , from which the difference  $x_1 - \hat{x}_1$  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:

$$|x_{1,i} - \hat{x}_{1,i}| \leq sc_i, \quad sc_i = Atol_i + \max(|x_{0,i}|, |x_{1,i}|) \cdot Rtol_i \quad (3.7)$$

where  $Atol_i$  and  $Rtol_i$  are the desired 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,

$$err = \sqrt{\frac{1}{n} \sum_{i=1}^n \left( \frac{x_{1,i} - \hat{x}_{1,i}}{sc_i} \right)^2} \quad (3.8)$$

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

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

If  $\text{err} \leq 1$ , the solution step is accepted, the time level is increased by  $h$ , and the step length is increased. 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  $\text{err} > 1$ , the solution step is rejected, the time level remains unaltered, and the step length is decreased. The procedure for updating the step length can be summarized as follows:

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

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

### 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

$$\nabla \mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}) \approx \begin{pmatrix} \frac{\mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^r) - \mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^l)}{2\delta x} & \frac{\mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^u) - \mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^l)}{2\delta y} \end{pmatrix}. \quad (3.11)$$

The Cauchy-Green strain tensor was then calculated as per equation (2.8). Farazmand and Haller (2012) bring up the point 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 a LCS due to the small initial separations  $\delta x$  and  $\delta y$ , resulting in the auxiliary tracers undergoing less stretching than the main tracers which lay on opposite sides of repelling LCSs.

For this reason, an approximation of the Jacobian of the flow map was also made by means of

applying centered differencing to the main tracers, as

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

The artificial grid extension, as outlined in section 3.2.1, allowed for a consistent centered differencing approximation for all the main tracers, including 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 *eigenvalues* of the Cauchy-Green strain tensor were thus calculated by means of the main tracers, equation (3.12), while the approximation of the corresponding *eigenvectors* were calculated by means of the auxiliary tracers, equation (3.11).

### 3.4 IDENTIFYING LCS CANDIDATES NUMERICALLY

Farazmand and Haller (2012) found the identification of the zeros of the inner product in equation (2.13d) of theorem 1 to be numerically sensitive. Therefore, they suggest a reformulated set of conditions which make for more robust numerical implementation, as follows:

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

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

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

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

The relaxation of condition (2.13b) in theorem 1 from strict inequality to condition (3.13b), which also allows equality, means that LCSs are allowed to have finite thickness. However, the set of criteria given in equation (3.13) enforce that all LCSs have uniquely defined local orientations. Conditions (2.13c) 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.

Because of the artificial 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 tracers in the entire computational domain.

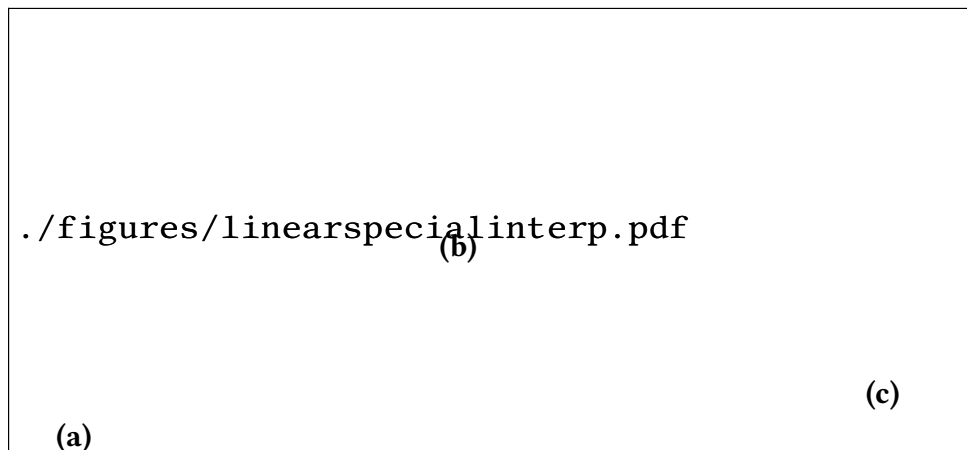
### 3.4.1 A framework for computing smooth strainlines

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  $C_{t_0}^t(\mathbf{x}_0)$ . The tensor lines tangent to the  $\xi_1$ -field will be referred to as *strainlines* in the following, a term coined by Farazmand and Haller (2012). Aside from points within  $\mathcal{U}$  which exhibit repeated eigenvalues and thus oriental discontinuities in both eigenvector fields, strainlines can be computed as smooth trajectories of the ordinary differential equation

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

As pointed out by Onu, Huhn, and Haller (2015), the orientational discontinuities of the  $\xi_1$ -field are removable, through careful monitoring and local reorientation. This process can be described in terms of three steps. First, the nearest neighboring grid points are identified. Then, oriental discontinuities inbetween the grid elements are identified by inspecting the inner product of the  $\xi_1$  vectors of adjacent grid points. Rotations exceeding  $90^\circ$  between pairs of neighboring vectors are labelled as oriental discontinuities, which are corrected prior to the linear interpolation by flipping the corresponding vectors by  $180^\circ$ . In the end, linear interpolation is used within the grid element.

Furthermore, should the  $\xi_1$ -vector obtained from the local special-purpose linear interpolation outlined above prove to be rotated by more than  $90^\circ$  relative to the  $\xi_1$ -vector from the previous level of pseudotime used in the numerical integration of system (3.14), it would be flipped  $180^\circ$ , by the same logic as in the special linear interpolation. The entire process of the special-purpose local linear interpolation method is outlined in figure 3.3.



**Figure 3.3:** Illustration of the special linear interpolation used for the  $\xi_1$  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  $180^\circ$  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:

$$\mathbf{r}'(s) = \text{sgn} \left( \left\langle \mathbf{f}(\mathbf{r}(s)), \mathbf{r}'(s - \Delta) \right\rangle \right) \mathbf{f}(\mathbf{r}'(s)), \quad (3.15)$$

where  $\mathbf{f}$  denotes the special-purpose local linear interpolation of the  $\xi_1$  field, as outlined above and in figure 3.3,  $\Delta$  is the numerical step length used in the numerical integration, while the signum function is defined as

$$\text{sgn}(x) = \begin{cases} 1, & \text{for } x > 0 \\ 0, & \text{for } x = 0 \\ -1, & \text{for } x < 0 \end{cases} \quad (3.16)$$

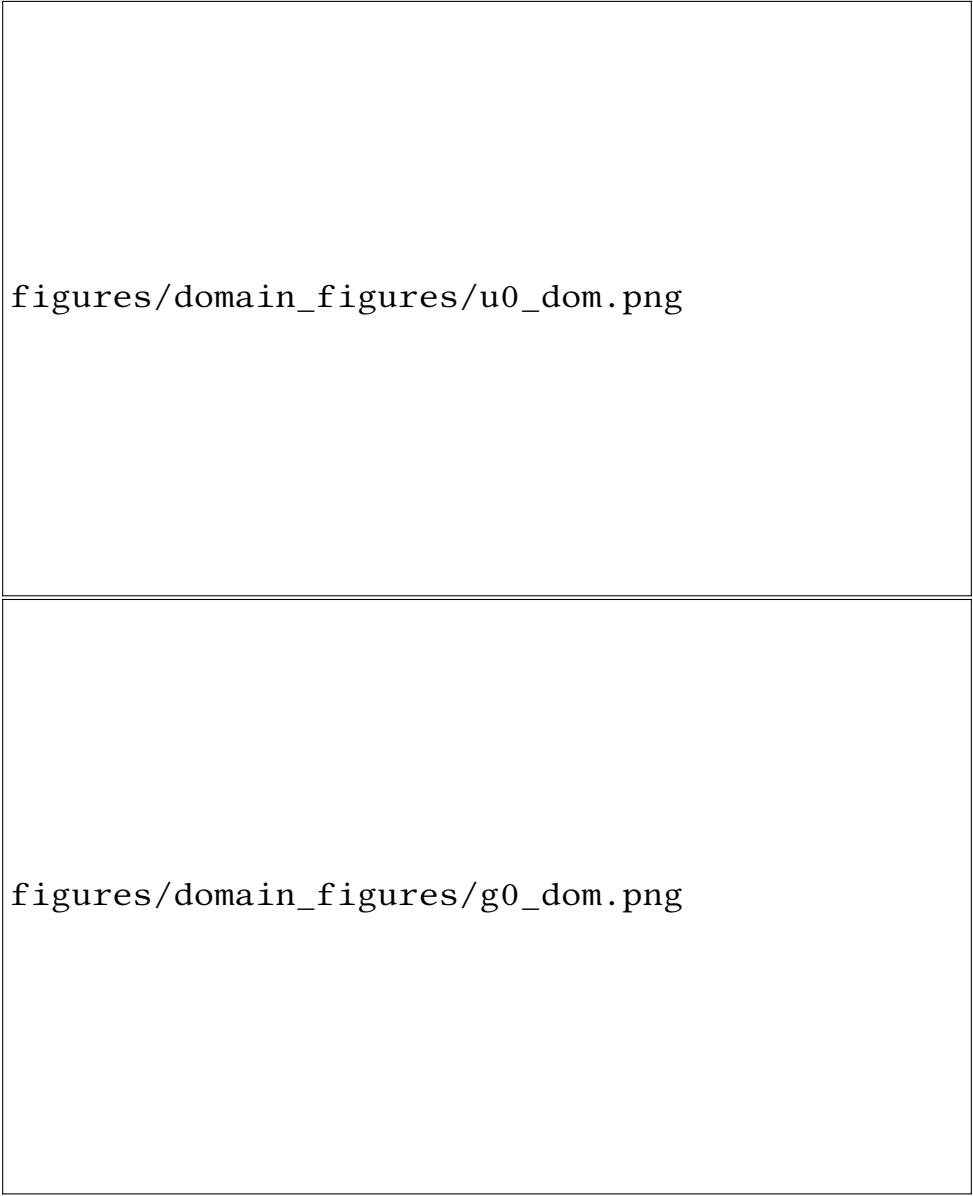
### 3.4.2 Extracting hyperbolic LCSs from strainlines

If a material line  $\mathcal{M}(t_0)$  lies within a strainline, it automatically fulfills condition (3.13c). 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 of the original  $\xi_1$  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  $\xi_1$  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  $\xi_1$  indiscernible. As a computational measure of this degeneracy, the scalar field defined as

$$\alpha(\mathbf{x}_0) = \left( \frac{\lambda_2(\mathbf{x}_0) - \lambda_1(\mathbf{x}_0)}{\lambda_2(\mathbf{x}_0) + \lambda_1(\mathbf{x}_0)} \right)^2 \quad (3.17)$$



figures/domain\_figures/u0\_dom.png

figures/domain\_figures/g0\_dom.png

**Figure 3.4:** The set  $\mathcal{U}_0$  for the double gyre system, given by equations (3.1) and (3.3), is shown at the top. Its intersection 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  $\mathcal{U}_0$ .

was used. 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 the predefined threshold of  $10^{-6}$ , the point  $\mathbf{x}$  was flagged as degenerate, thus stopping the strainline integration.

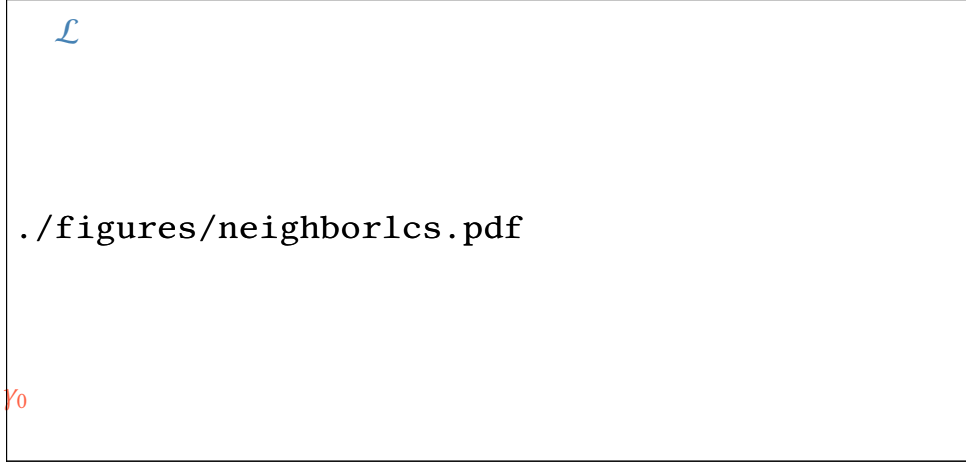
Because there is no a priori way of knowing that the selected strain initial conditions, shown in figure 3.4, are not in fact located somewhere in the middle of a strainline, two strainline segments were computed and merged for each initial condition. One moving forwards in pseudotime, and one moving backwards, i.e., using the  $180^\circ$  rotated  $\xi_1$  vector field in the system given by equation (3.15). This ensured that no strainline was cut prematurely, purely as a consequence of the reduced number of strain initial conditions. Furthermore, the classical Runge-Kutta method with a pseudotime step length  $\Delta = 10^{-3}$  was used in order to compute all of the strainlines, regardless of the integration method used in the advection of the tracers, described in section 3.2. This choice was made because this project is centered around the dependence of the choice of integration method in the tracer advection. The step length was chosen by similar logic as presented in section 3.2.2; however, there is no way of knowing in advance, how long (in terms of numerical integration steps) any given strainline will be. For this reason, it is impossible to estimate the accumulated floating-point arithmetic errors in advance. Regardless, we expect diminishing returns in terms of numerical accuracy by lowering the step length even further, because the accuracy is always bounded from below by the inherent accuracy of the double-precision floating-point numbers.

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, where strainline integration is only stopped if one of the aforementioned LCS conditions fail repeatedly over the 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.

### 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. 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 comparing the values of  $\bar{\lambda}_2(\gamma_0)$ , the average of  $\lambda_2$  on the curve  $\gamma_0$ , at the neighboring intersections of all sufficiently close strainline segments along each of the lines in  $\mathcal{L}$ . The process is illustrated in figure 3.5.

Intersections between strainlines and the lines in  $\mathcal{L}$  are found by means of linear interpolation.



**Figure 3.5:** Illustration of the identification process of strainlines which are local strain maximizers. Local apices of  $\bar{\lambda}_2(\gamma)$ , the average of  $\lambda_2$  over a curve  $\gamma$ , are found along a set  $\mathcal{L}$  of horizontal and vertical lines. The strainline segment  $\gamma_0$  is identified as a local maximizer if it contains a locally maximal  $\bar{\lambda}_2$  value along at least one of the lines in  $\mathcal{L}$ . Intersections of nearby strainlines, which 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  $\gamma_0$ .

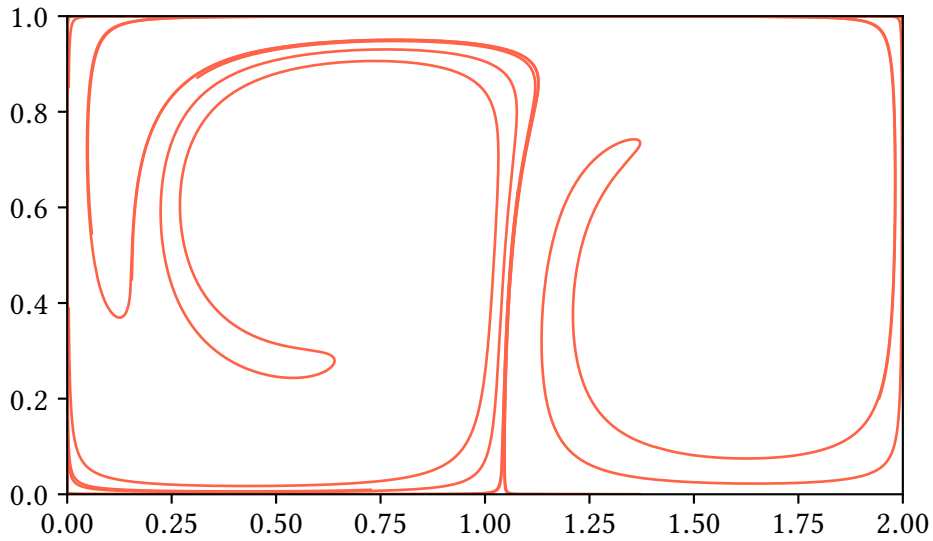
Should a strainline segment prove to be a local maximizer along at least one of the lines in  $\mathcal{L}$ , the strainline segment is labelled as a LCS. Adjacent intersections who are separated by a distance larger than the preselect threshold  $l_n = 0.2$  are excluded from the local maximization process, as indicated in figure 3.5. 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 the pre-set small length  $l_{\min} = 1$  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 general, robust way of selecting the lines in  $\mathcal{L}$ . However, a robust way of identifying strainlines which are local strain maximizers seemingly has not yet been unambiguously determined in the literature. 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 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

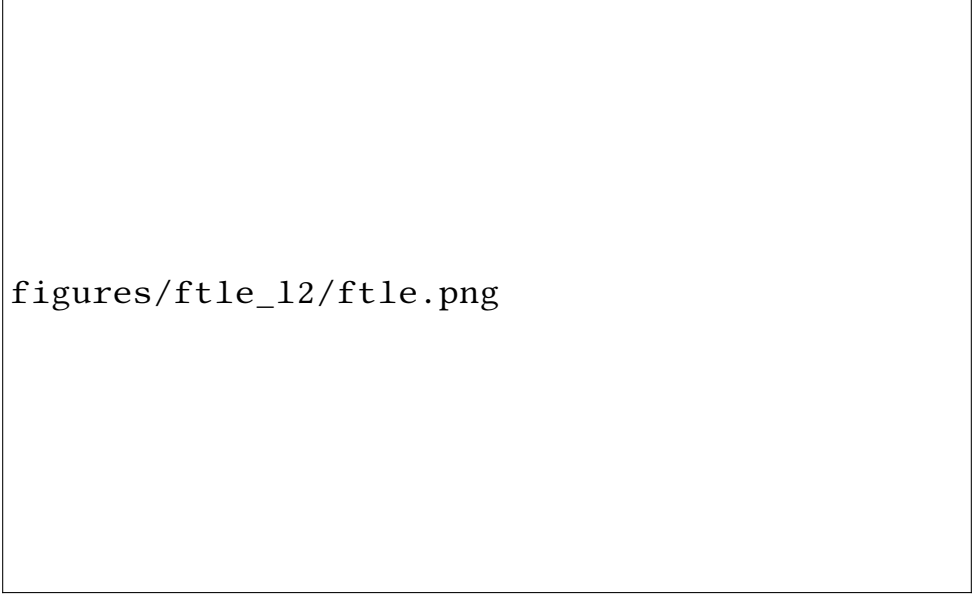
applicable on the entirety of the computational domain.

In order to eliminate the weakest among the LCSs, it is thus clear that the lines in  $\mathcal{L}$  must be selected carefully. Here, we used the approach of generating the set  $\mathcal{L}$  in order to maximize the number of intersections with the computed strainlines, using as few lines as possible. 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). We can only assume that Farazmand and Haller must have done something similar, as they have been unavailable for communication.



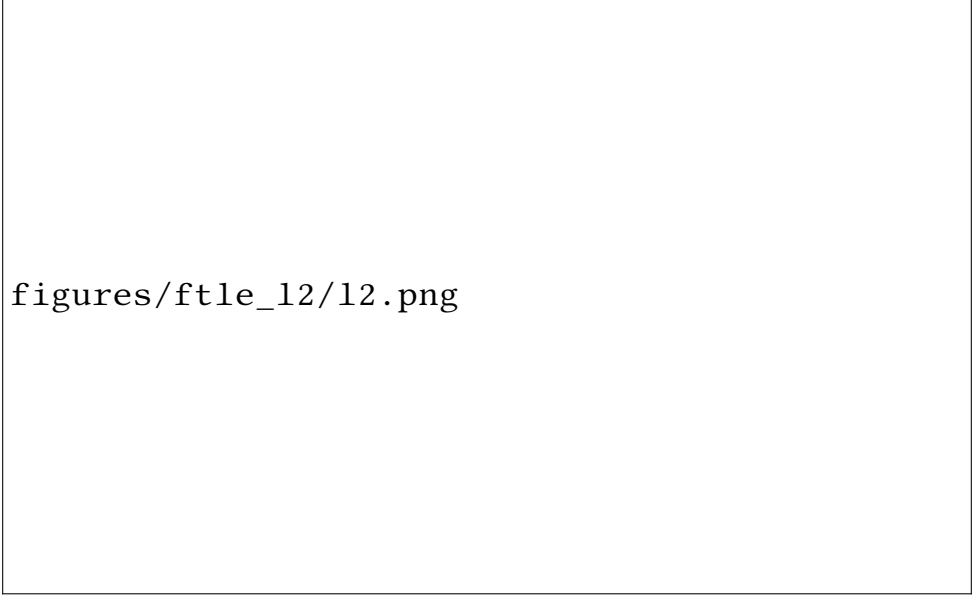
**Figure 3.6:** The repelling reference LCS of the double gyre system, i.e., the LCS obtained by means of the reference numerical integrator, as outlined in section 3.2.2. This curve agrees visually with the LCS found in the literature, for instance in Farazmand and Haller (2012). It was used as the benchmark for comparing the accuracy of the considered numerical integrators, outlined in section 2.1.2, for varying numerical step lengths and tolerance levels.

The reference LCS, that is, the LCS obtained with the reference numerical integrator, as mentioned in section 3.2.2, is presented in figure 3.6. It agrees visually with the LCS found in the literature, for example by Farazmand and Haller (2012). This curve 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.



figures/ftle\_12/ftle.png

(a) FTLE field



figures/ftle\_12/12.png

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

**Figure 3.7:** Plots of the FTLE field (a) and  $\lambda_2(\mathbf{x}_0)$  distribution (??) 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 contours 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 3.5.

### 3.5 ESTIMATION OF ERRORS

Aside from the qualitative 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 (2.7), was computed as the root mean square deviation (hereafter abbreviated as RMSD) with regards to the reference, as follows:

$$\text{RMSD}_{\text{flow map}} = \sqrt{\frac{1}{\tilde{N}} \sum_{\mathbf{x}_0 \in \tilde{\mathcal{U}}} \left( \widehat{\mathbf{F}}_{t_0}^t(\mathbf{x}_0) - \mathbf{F}_{t_0}^t(\mathbf{x}_0) \right)^2}, \quad (3.18)$$

where the summation is over all of the tracers in the computational domain  $\tilde{\mathcal{U}}$ , including the tracers which constitute the artificially extended domain *and* the auxiliary tracers, as outlined in section 3.2.1.  $\tilde{N}$  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 eigenvalues,  $\lambda_1(\mathbf{x}_0)$  and  $\lambda_2(\mathbf{x}_0)$  of the Cauchy-Green strain tensor field, as given by equation (2.11) was also computed completely analogously:

$$\text{RMSD}_{\text{eigenvalue}} = \sqrt{\frac{1}{N} \sum_{\mathbf{x}_0 \in \mathcal{U}} \left( \widehat{\lambda}(\mathbf{x}_0) - \lambda(\mathbf{x}_0) \right)^2}, \quad (3.19)$$

where the summation is over all tracers within the computational domain  $\mathcal{U}$ , for which the Hessians of the largest eigenvalue were computed, as per equation (3.13).  $N$  is the corresponding number of tracers.  $\widehat{\lambda}(\mathbf{x}_0)$  is the approximation of the true eigenvalue  $\lambda(\mathbf{x}_0)$  located at  $\mathbf{x}_0$ , obtained by means of the numerical integrator in question.

The best way of estimating the error in the computed eigenvectors of the Cauchy-Green Strain tensor field is by means of their orientation, because the eigenvectors themselves are normalized to unit length. Since the iteration of strainlines was performed in both directions of pseudotime, as described in section 3.4.1, a  $180^\circ$  shift in direction would not influence the LCS calculation in any way. Thus, the eigenvector directions were computed as azimuthal angles in the interval  $[0^\circ, 180^\circ)$ . The RMSD for the direction of the eigenvector fields was then calculated as follows:

$$\begin{aligned} \xi(\mathbf{x}_0) &= \left( \xi_x(\mathbf{x}_0), \xi_y(\mathbf{x}_0) \right), \quad \phi(\mathbf{x}_0) = \arctan \left( \frac{\xi_y(\mathbf{x}_0)}{\xi_x(\mathbf{x}_0)} \right), \\ \text{RMSD}_{\text{eigenvector direction}} &= \sqrt{\frac{1}{N} \sum_{\mathbf{x}_0 \in \mathcal{U}} \left( \widehat{\phi}(\mathbf{x}_0) - \phi(\mathbf{x}_0) \right)^2}, \end{aligned} \quad (3.20)$$

where the conventions for  $\mathcal{U}$  and  $N$  are the same as for the RMSD of the eigenvalues.  $\hat{\phi}(\mathbf{x}_0)$  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 and false negatives should be treated separately, as they would influence the overall flow patterns in the system in different ways. Because numerical noise could result in LCS curve segments being erroneously identified as either, a lower numerical threshold  $l_{\text{noise}} = 0.01$ , that is, ten times the numerical integration step used in order to compute the constituent strainlines, was used. Any point on a LCS curve which was farther away from all points on the reference LCS than  $l_{\text{noise}}$  was identified as a false positive. Likewise, any point on the reference LCS curve which was farther away from all points on the LCS curve under consideration than  $l_{\text{noise}}$  was flagged as a false negative. In order to estimate the accumulated offset of the false LCS segments, the midpoint rule was used;

$$\text{Offset}_{\text{false LCSs}} = \sum_{\substack{\mathbf{x} \in \gamma \\ \mathbf{x} \notin \hat{\gamma}}} \min_{\hat{\mathbf{x}} \in \hat{\gamma}} \|\mathbf{x} - \hat{\mathbf{x}}\| \cdot \Delta, \quad (3.21)$$

where the sum is over all points with a greater offset than the aforementioned  $l_{\text{noise}}$ , and  $\Delta = 0.001$  is the numerical integration step length used in order to compute strainlines. Concerning false positives,  $\hat{\gamma}$  denotes the reference LCS curve, and  $\gamma$  the approximation under consideration. For false negatives, the roles are reversed.

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  $l_{\text{noise}}$ ), the root mean square of the distance separating each point on the LCS curve with the nearest point on the reference, was calculated:

$$\text{RMSD}_{\text{LCS}} = \sqrt{\frac{1}{\check{N}} \sum_{\mathbf{x} \in \gamma} \min_{\hat{\mathbf{x}} \in \hat{\gamma}} \|\mathbf{x} - \hat{\mathbf{x}}\|^2}, \quad (3.22)$$

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.  $\check{N}$  is the number of points which constitute  $\hat{\gamma}$ , the LCS curve found by means of the numerical integrator in question, which are within the distance  $l_{\text{noise}}$  of any point on  $\gamma$ , the reference LCS curve. The idea is that LCS curves need not necessarily start from, or end at, the same points in space. For this reason, a robust way of estimating the error of a given LCS curve is the summation of the smallest distances separating each point on the curve from the reference.

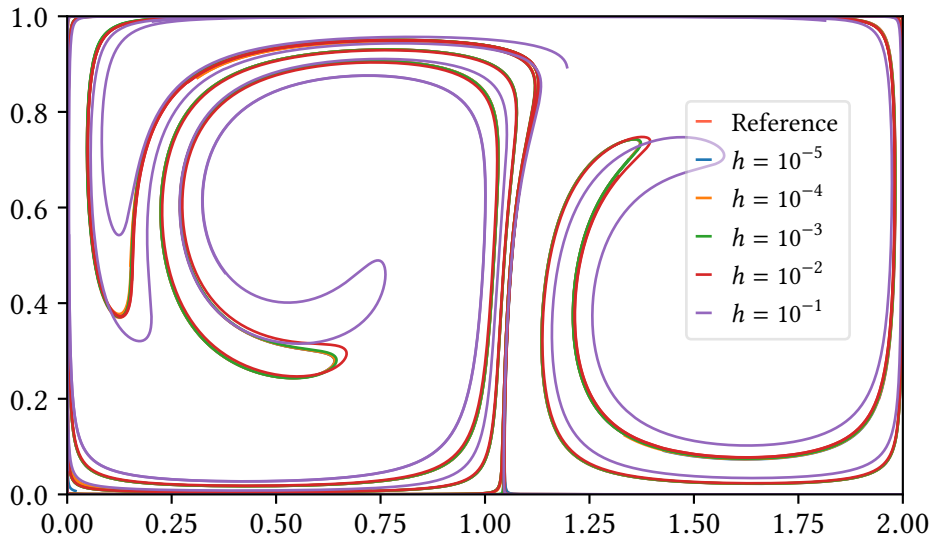


## 4 Results and Discussion

### 4.1 THE LCS CURVES OBTAINED USING THE DIFFERENT SCHEMES

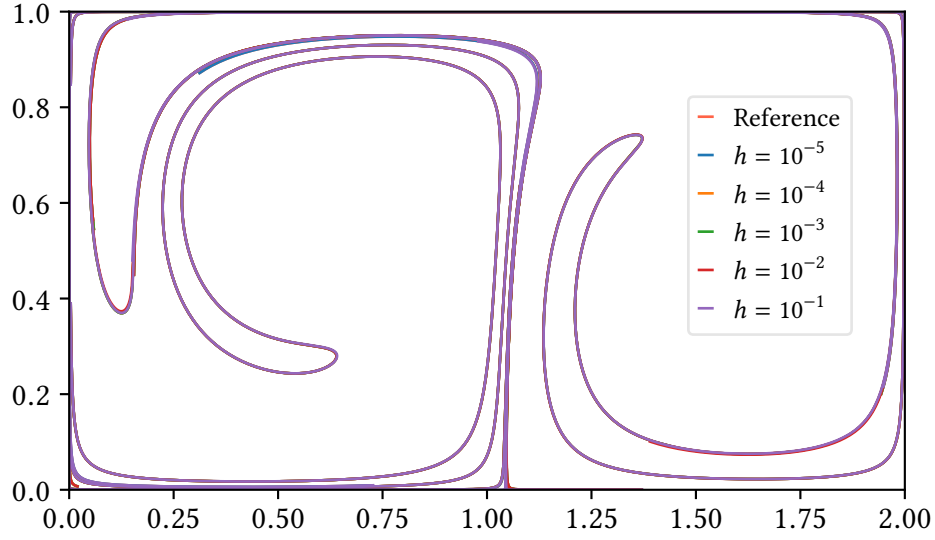
Here, the repelling LCS curves found by means of the variety of numerical integration schemes under consideration, are presented. All of the LCS curves obtained for a given integrator, for all numerical time step lengths or tolerance levels are included in one figure, where the reference LCS, shown in figure 3.6, 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

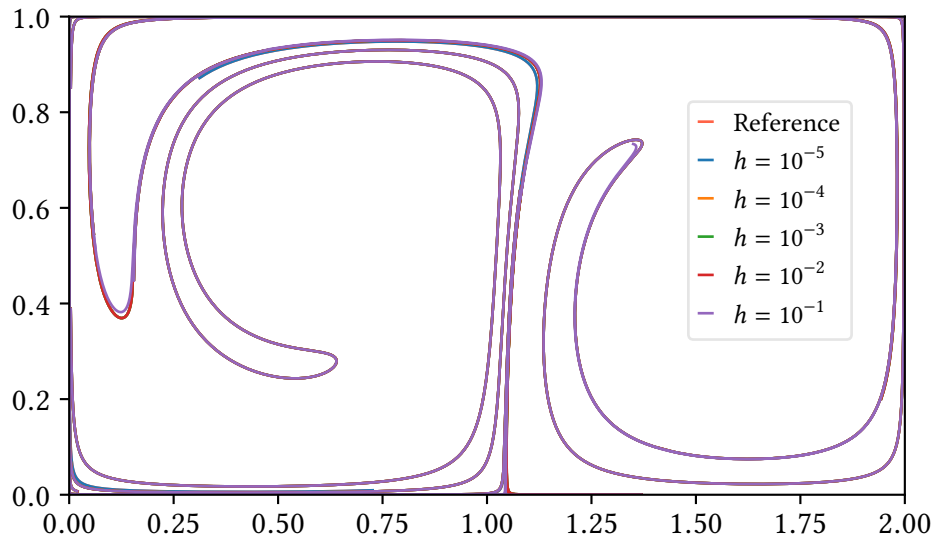


**Figure 4.1:** LCS curves found by means of the Euler integration scheme. The reference LCS, as shown by itself in figure 3.6, is plotted on the bottom layer. There is a clearly visible offset compared to the reference, for all but the two smallest numerical step lengths considered. The two latter LCS curves appear to conform well.

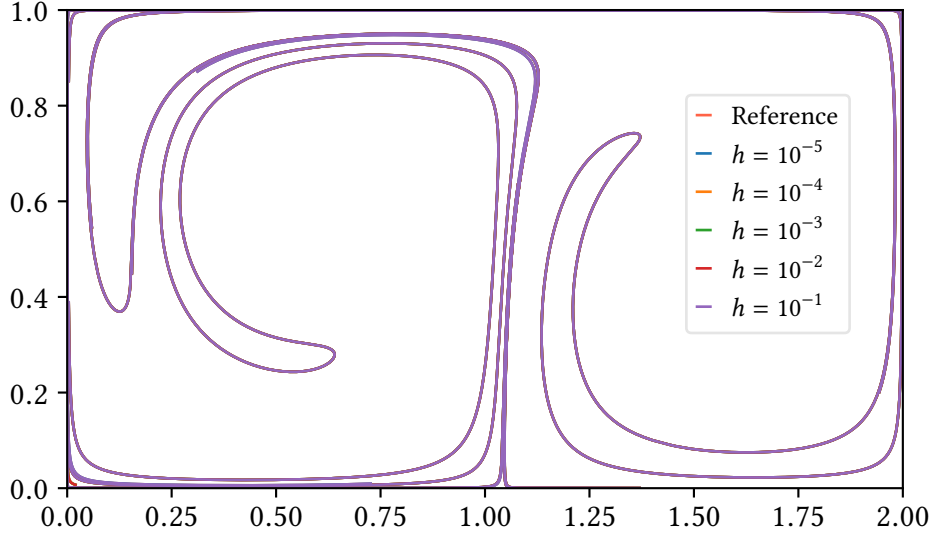
The LCS curves obtained from the Euler scheme using relatively large step lengths are the most obviously incorrect ones, in comparison to the reference. There appears to be very little separating the performance of the other singlestep methods. Interestingly, the Heun scheme appears to yield more accurate 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<sup>nd</sup> order accurate in the time step, whereas the Kutta scheme is 3<sup>rd</sup> order accurate. On a less surprising note, seeing as it is 4<sup>th</sup> 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 by itself in figure 3.6, is plotted on the bottom 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 by itself in figure 3.6, is plotted on the bottom layer. There are some disparities with regards to the reference both for the smallest and the largest numerical time step lengths considered. These are most pronounced in the lower left corner, and near the leftmost 'U' shape by  $y = 0.4$ .

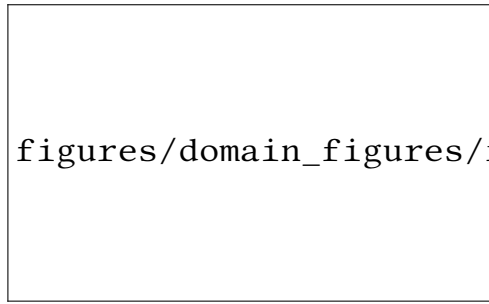


**Figure 4.4:** LCS curves found by means of the classical Runge-Kutta integration scheme. The reference LCS, as shown by itself in figure 3.6, is plotted on the bottom layer. One of the very few visible discrepancies belongs to the second largest numerical time step length considered, and is located in the lower left corner.

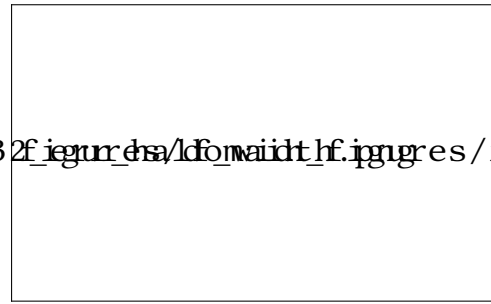
#### 4.1.2 LCS curves stemming from adaptive stepsize methods

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  $\text{tol} = 10^{-1}$ . 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. In fact, most likely because of the large density of points in these domains, keeping track of the neighbors for each individual strainline and its intersections with the lines in  $\mathcal{L}$  turned out to be an unconquerable task. It is known that storing the necessary data requires in excess of 28 GB of memory, at which point even the NTNU supercomputer, Vilje, proved insufficient.

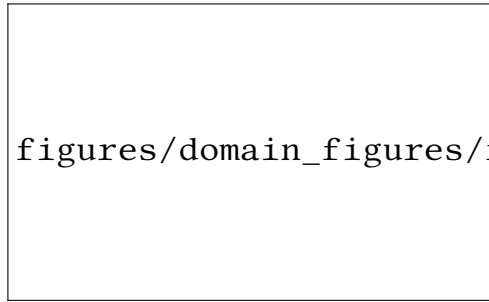
Notably, all of the embedded integration methods result in LCSs which are dissimilar from the reference for the tolerance level  $\text{tol} = 10^{-2}$ , where only the curve obtained by the Dormand-Prince 8(7) scheme is anything alike the reference. More curiously, the Bogacki-Shampine 3(2) method appears to outperform its theoretically more accurate sibling, 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.



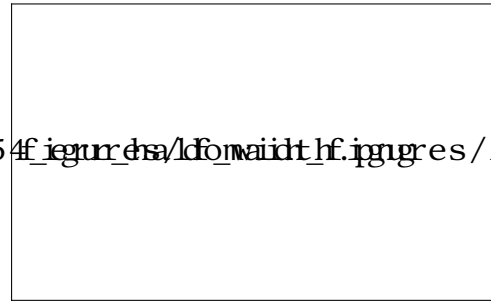
(a) Bogacki-Shampine 3(2)



(b) Bogacki-Shampine 5(4)

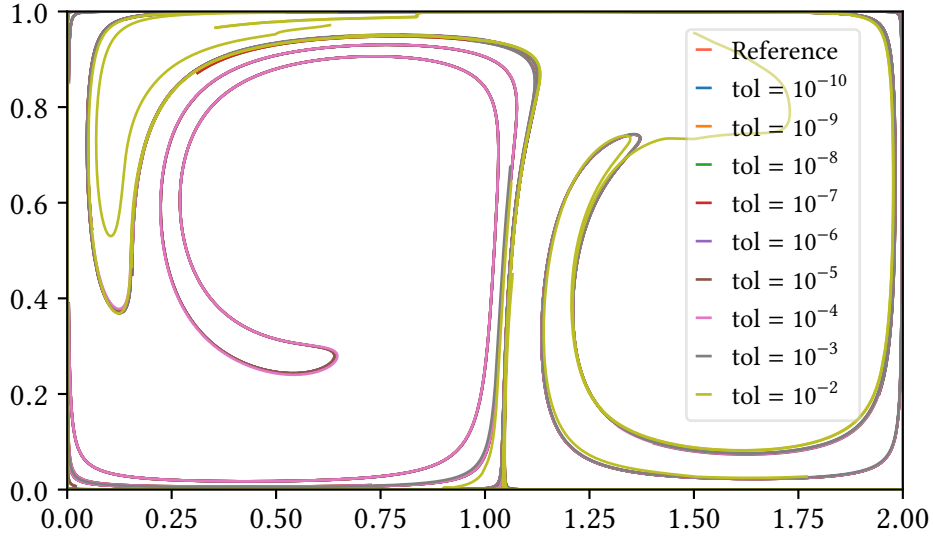


(c) Dormand-Prince 5(4)

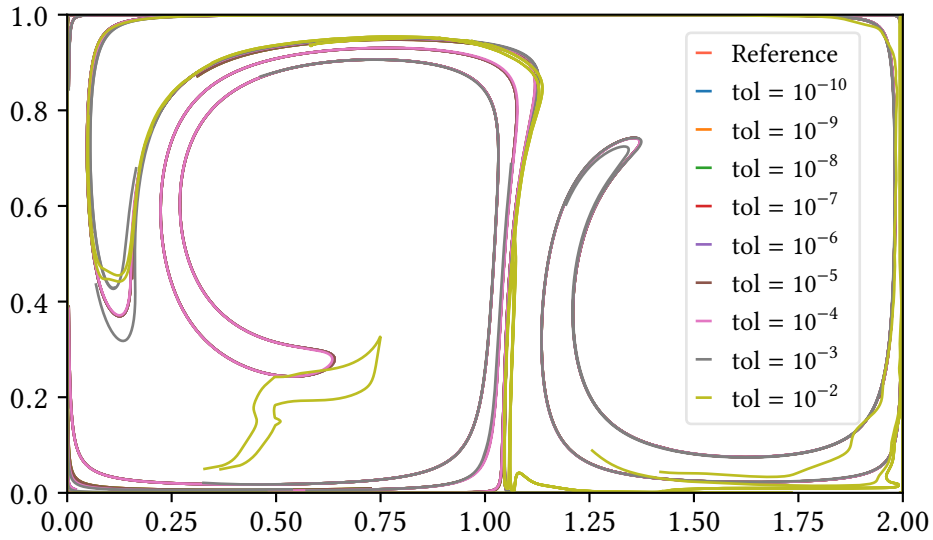


(d) Dormand-Prince 8(7)

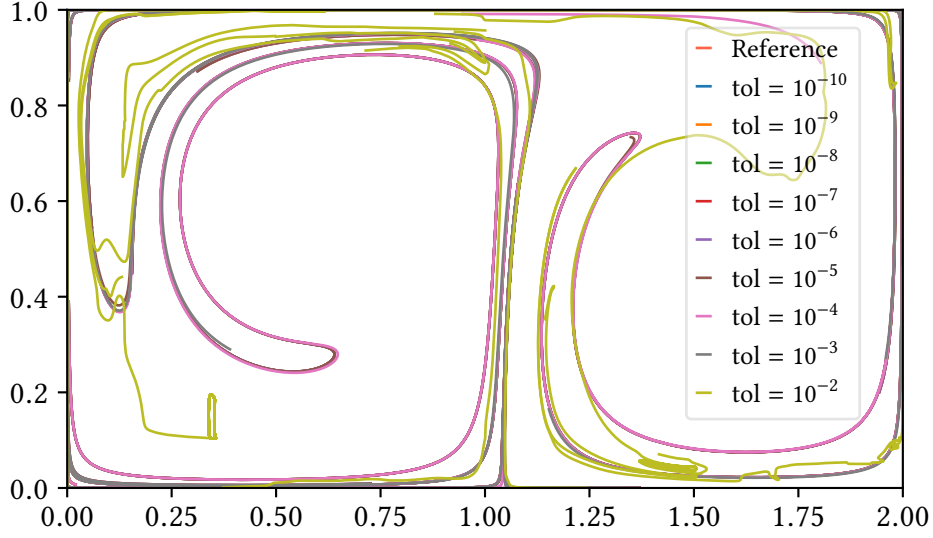
**Figure 4.5:** The  $\mathcal{U}_0$  domains obtained with the adaptive stepsize integration schemes, with numerical tolerance level  $\text{tol} = 10^{-1}$ . Out of the four schemes considered, only the Dormand-Prince 8(7) method is remotely close to reproducing the reference domain, shown in 3.4.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



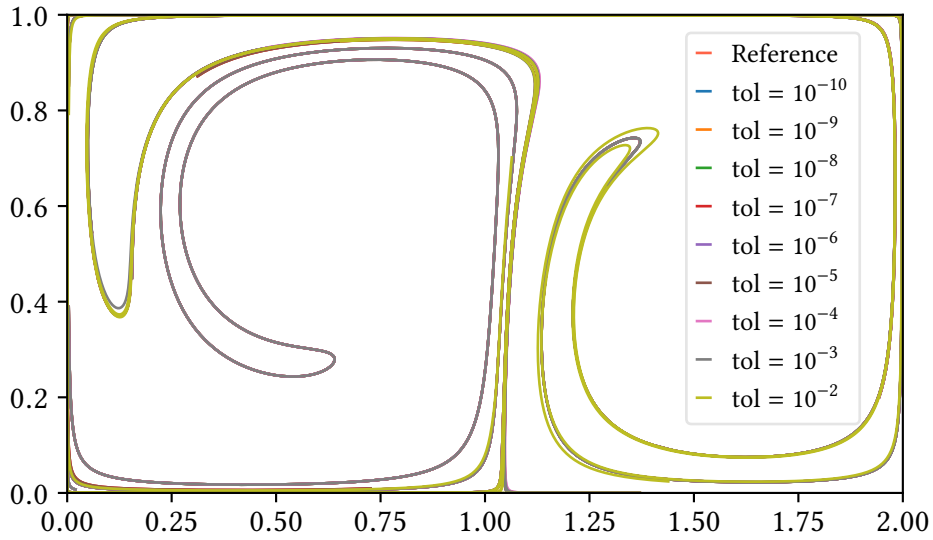
**Figure 4.6:** LCS curves found by means of the Bogacki-Shampine 3(2) integration scheme. The reference LCS, as shown by itself in figure 3.6, is plotted on the bottom layer. Note that the LCS for the lowest tolerance level considered, that is,  $\text{tol} = 0.1$ , is not included. This is because the corresponding  $\mathcal{U}_0$  domain, shown in figure 4.5a, and the reference  $\mathcal{U}_0$ , shown in figure 3.4 are nothing alike. The second lowest tolerance level, i.e.,  $\text{tol} = 10^{-2}$ , is the main source of discrepancies in this case. However, there are also visible artifacts in the lower left corner, and near  $x = 1$ .



**Figure 4.7:** LCS curves found by means of the Bogacki-Shampine 5(4) integration scheme. The reference LCS, as shown by itself in figure 3.6, is plotted on the bottom layer. Note that the LCS for the lowest tolerance level considered, that is,  $\text{tol} = 0.1$ , 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 dissimilar. Here, there are visible discrepancies for all tolerance levels  $\text{tol} > 10^{-6}$ . These are most prominent by the ‘U’ shape near  $y = 0.4$ , and in the lower left corner.



**Figure 4.8:** LCS curves found by means of the Dormand-Prince 5(4) integration scheme. The reference LCS, as shown by itself in figure 3.6, is plotted on the bottom layer. Note that the LCS for the lowest tolerance level considered, that is,  $\text{tol} = 10^{-1}$ , 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 dissimilar. Here, there are visible disparities for for all tolerance levels  $\text{tol} > 10^{-6}$ . These are most noticeable near  $x = 1$ , and in the lower left corner, due to the curve stemming from  $\text{tol} = 10^{-2}$  being so erratic.



**Figure 4.9:** LCS curves found by means of the Dormand-Prince 8(7) integration scheme. The reference LCS, as shown by itself in figure 3.6, is plotted on the bottom layer. Note that the LCS for the lowest tolerance level considered, that is,  $\text{tol} = 10^{-1}$ , 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 unlike one another. Here, the most immediately discernible dissimilarities emanate from the tolerance level  $\text{tol} = 10^{-2}$ . Close inspection, however, reveals discrepancies for all tolerance levels  $\text{tol} > 10^{-6}$ , particularly in the lower left corner.

## 4.2 MEASURES OF ERROR

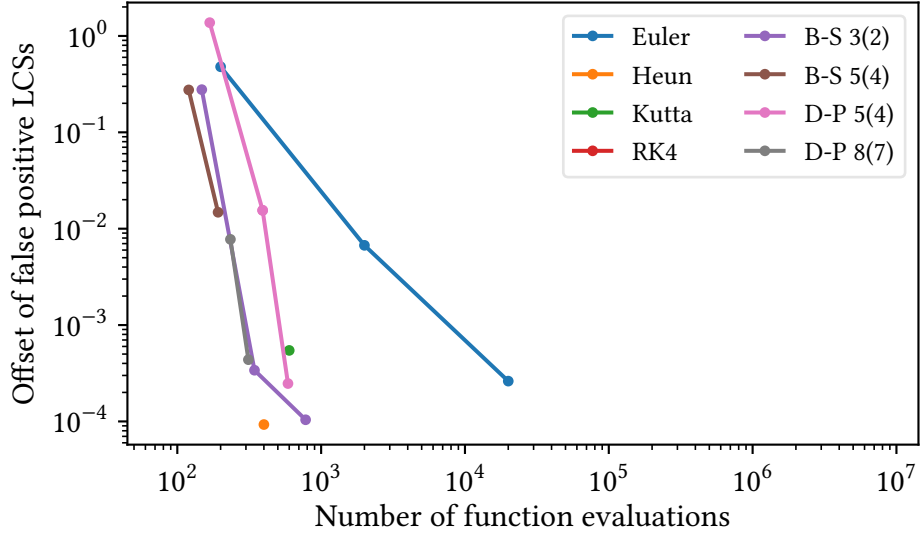
Here, the measures of error introduced in section 3.5 for the different numerical integration schemes considered, is presented. For each different type of error, two figures are included. One in which the error of the singlestep methods is shown as a function of the integration step length, and one in 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 law in  $h$ , 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 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 unaccepted trial integration steps. The number of function evaluations for each 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.

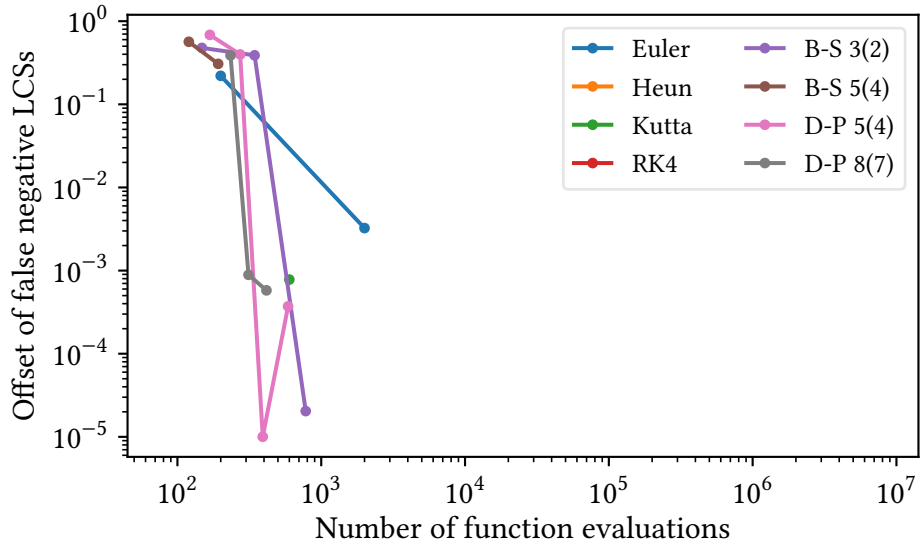
### 4.2.1 *Computed deviations in the LCS curves*

Figures 4.10 and 4.11 indicate that the offset of false positives and negatives, as defined in equation (3.21), 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, aside from providing an indication that the LCS present in the double gyre system is robust.

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 4.10 and 4.11, 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 with the expected power laws of the numerical errors as a function of the time step, cf. definition 2. Furthermore, offsets for the adaptive stepsize methods are not included for the tolerance level  $10^{-1}$ , because the LCS identification proved to be an insurmountable task 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.



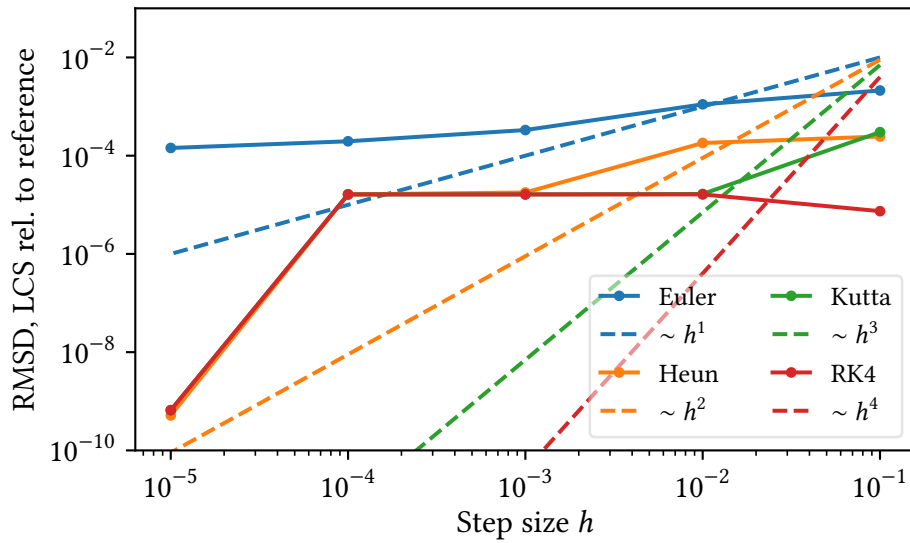
**Figure 4.10:** Offset of the LCS curve segments identified as false positives as a function of the required number of function evaluations. The integration schemes for which no points are visible, yielded no false positives. For the adaptive stepsize methods, the *average* number of function evaluations across all of the advected tracers is used as the metric.



**Figure 4.11:** Offset of the LCS curve segments identified as false negatives as a function of the required number of function evaluations. 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 metric.

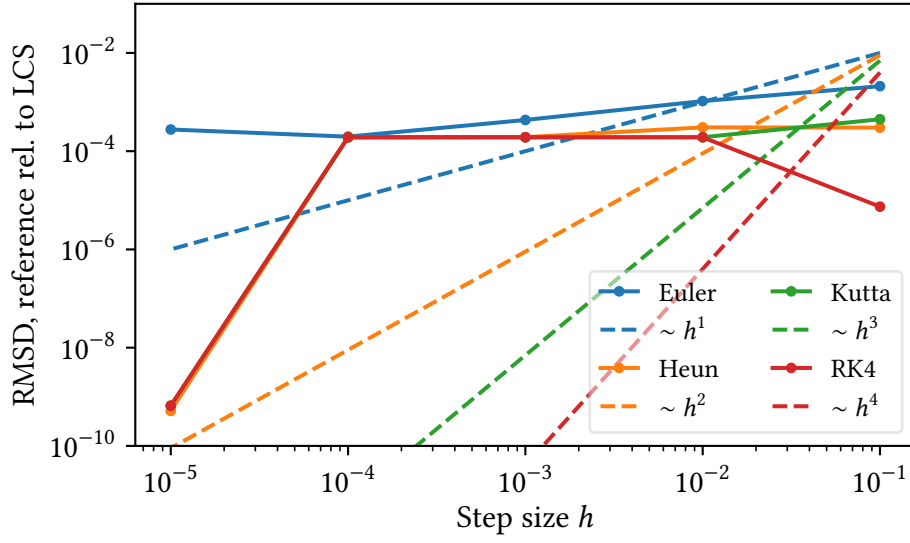


Figures 4.12 and 4.13 indicate that the RMSD of the LCS curves, as defined in equation (3.22), does not follow the expected power laws of the numerical errors for the various singlestep integrators. Moreover, the two measures of the RMSD show the same quantitative behaviour, which is to be expected. 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  $10^{-2}$ . In addition, 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.

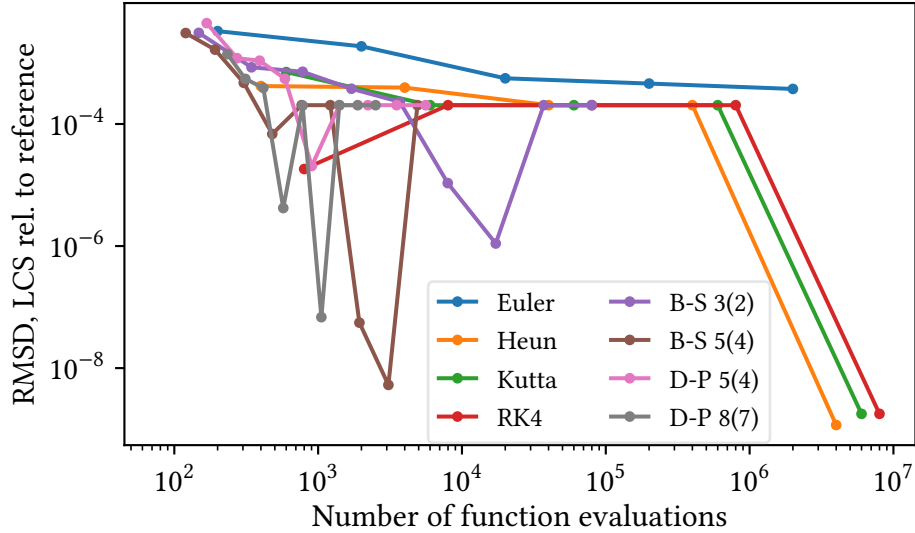


**Figure 4.12:** RMSD of the computed LCS curves relative to the reference as a function of numerical step length, for the singlestep methods considered. Curves with slopes corresponding to the expected power laws in error for the various methods are included in order to facilitate visual comparisons.

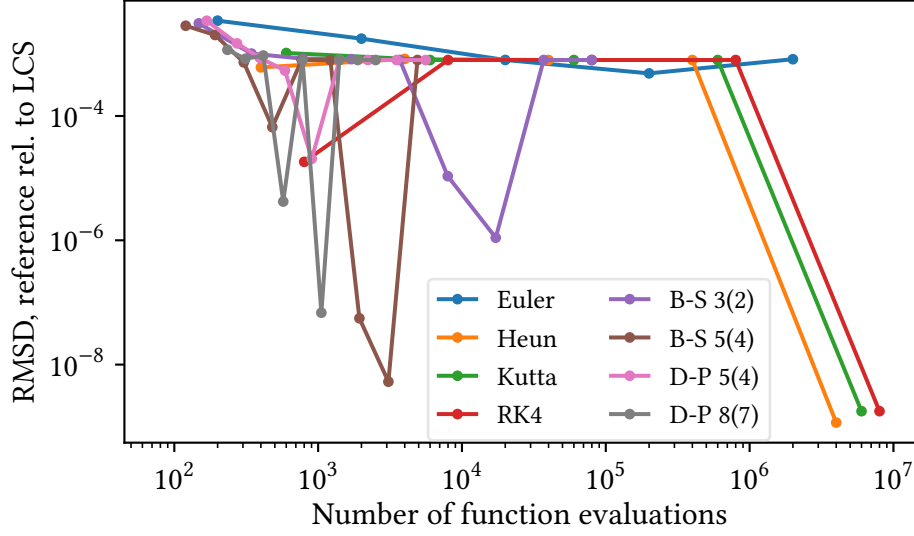
Interestingly, a similar pattern is apparent from figures 4.14 and 4.15, in that, the RMSD of the higher-order methods decreases steadily, until a certain point where it suddenly jumps to the same steady level as the (high order) singlestep methods. The most reasonable explanation is that when the number of function evaluations reaches some threshold, the accumulated floating-point arithmetic errors gain a larger influence than the inherent precision of the integrators. By similar logic, the sudden drop in RMSD for the high order singlestep methods with numerical step length  $h = 10^{-5}$ , as can be seen in figures 4.12 and 4.13, is most likely an artifact of the random nature of the accumulated floating-point errors. Finally, figures 4.14 and 4.15 suggests that the high order adaptive stepsize methods, namely the Bogacki-Shampine 5(4) and Dormand-Prince 8(7) schemes, provide the most efficient means of obtaining accurate LCS curves for the considered double gyre system.



**Figure 4.13:** RMSD of the reference LCS relative to the computed LCS curves as a function of numerical step length, for the singlestep methods considered. Curves with slopes corresponding to the expected power laws in error for the various methods are included in order to facilitate visual comparisons.



**Figure 4.14:** RMSD of the computed LCS curves relative to the reference as a function of the required number of function evaluations. For the adaptive stepsize methods, the *average* number of function evaluations across all of the advected tracers is used as the metric.

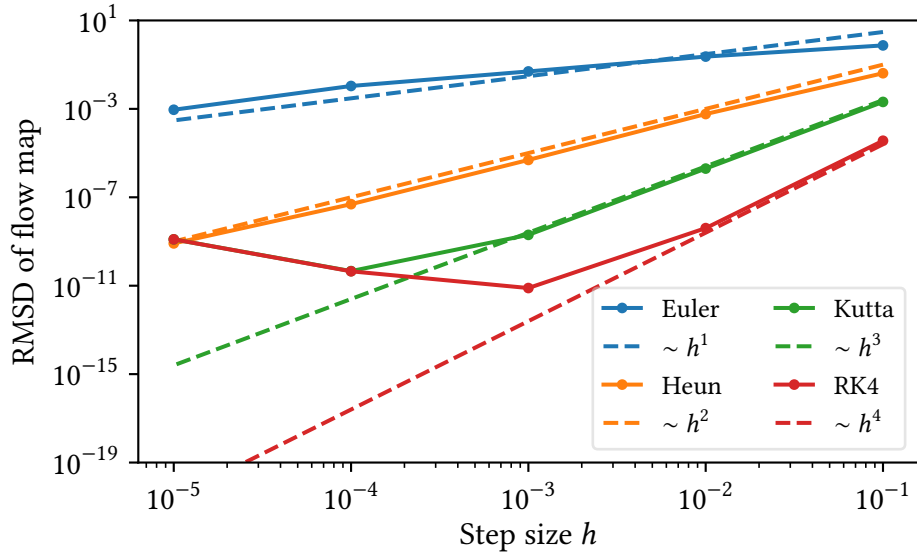


**Figure 4.15:** RMSD of the reference LCS relative to the computed LCS curves as a function of the required number of function evaluations. For the adaptive stepsize methods, the *average* number of function evaluations across all of the advected tracers is used as the metric.

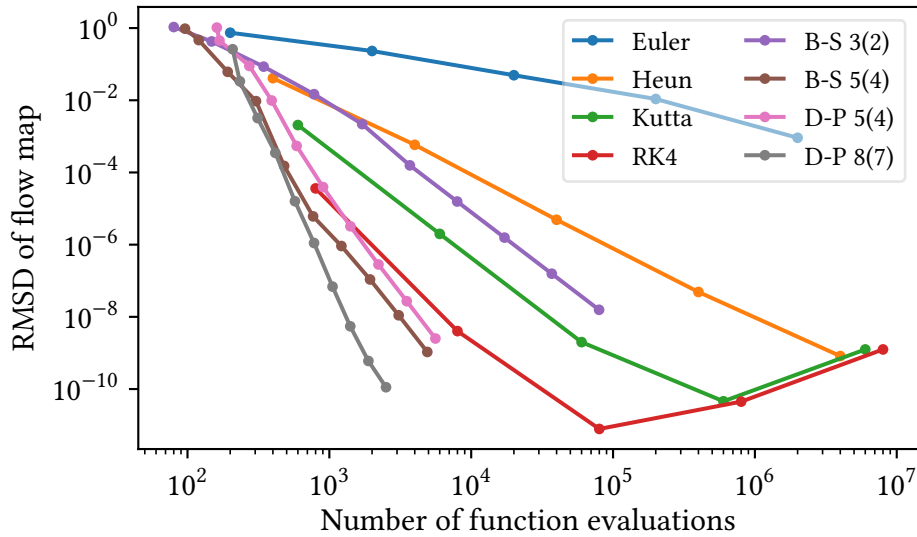
#### 4.2.2 Computed deviations in the flow maps

Figure 4.16 indicates that the RMSD of the flow maps, as defined in equation (3.18) 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 RMSD seem to exhibit boundedness 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 inflect upwards for sufficiently small step lengths. Based on this evidence, the chosen step lengths appear to be appropriate. Although neither the RMSD of the Euler scheme nor that of the Heun scheme appears to have reached their respective inflection points, the trends nevertheless indicate that there would not be much to gain in terms of accuracy before the accumulated floating-point errors outweigh the inherent precisions of the two schemes.

Figure 4.17 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 the ‘cheapest’ in terms of the required number of function evaluations when only a very crude approximation is needed, that is, using a very high numerical tolerance level. Furthermore, the its higher order sibling is able to keep up with the Dormand-Prince 8(7) method regarding efficiency, until a mean of approximately 600 function evaluations. This complies with the notion that the Bogacki-Shampine 5(4) method is highly optimized and practically behaves as an even higher order order method, as mentioned in section 2.1.2. Interestingly, there does not seem to be a direct correspondence between the RMSD of the flow maps, and that of the computed LCS curves, presented in section 4.2.1.



**Figure 4.16:** RMSD of the flow maps as a function of numerical step length, for the singlestep methods considered. Curves with slopes corresponding to the expected power law in error for the various methods are included in order to facilitate visual comparisons.

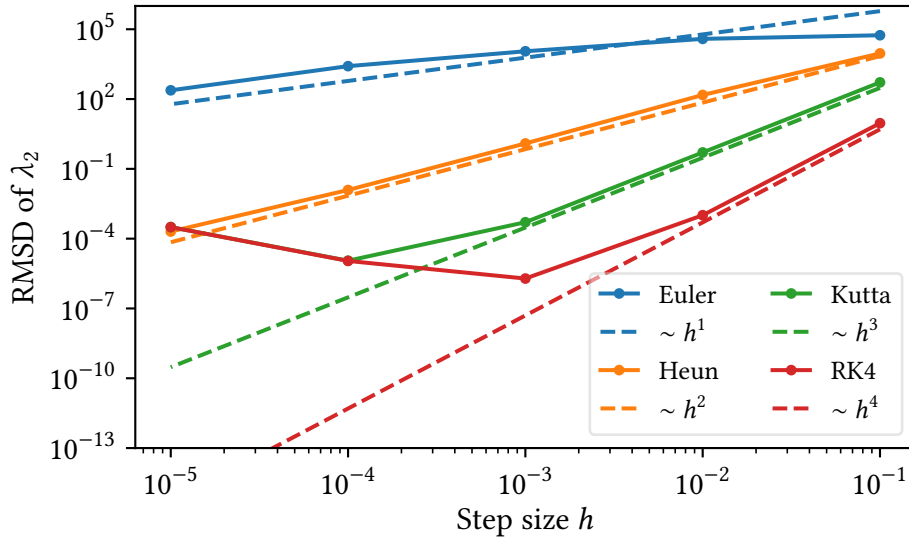


**Figure 4.17:** RMSD of all flow maps as a function of the required number of function evaluations. For the adaptive stepsize methods, the *average* number of function evaluations across all advected tracers is used as the metric.

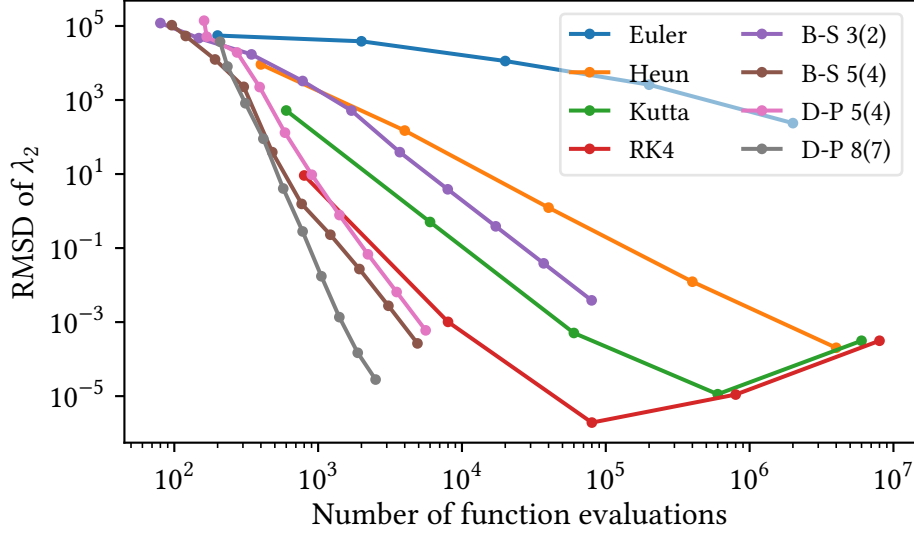
### 4.2.3 Computed deviations in the strain eigenvalues and -vectors

For brevity, only the RMSD of  $\lambda_2$  and the direction of  $\xi_2$ , 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  $\lambda_1$  scales similarly to that of  $\lambda_2$ , 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  $\lambda_2$  than  $\lambda_1$ . This is explicitly observable from conditions (3.13b) and (3.13d), and favors  $\lambda_2$  as the most crucial eigenvalue.

Figure 4.18 indicates that the RMSD of  $\lambda_2$ , as defined in equation (3.19), follow 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 4.2.2. Furthermore, inspection of figure 4.19 reveals that the RMSD of the  $\lambda_2$  scales similarly to the RMSD of the flow maps for all of the integration schemes, differing only by a numerical prefactor. This does make sense intuitively, seeing as the Cauchy-Green strain eigenvalues are computed by means of the *main* tracers, as described in section 3.3.



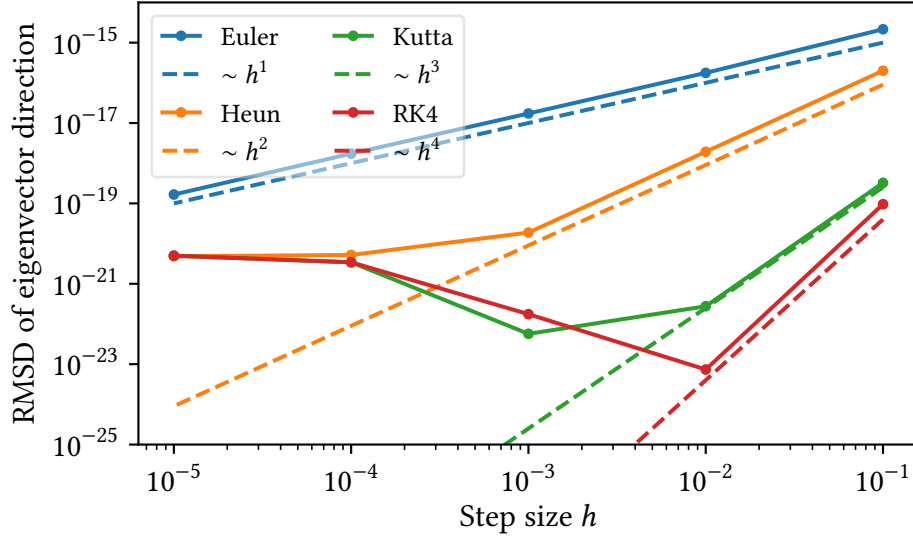
**Figure 4.18:** RMSD of the  $\lambda_2$  field as a function of numerical step length, for the singlestep methods considered. Curves with slopes corresponding to the expected power laws in error for the various methods are included in order to facilitate visual comparisons.



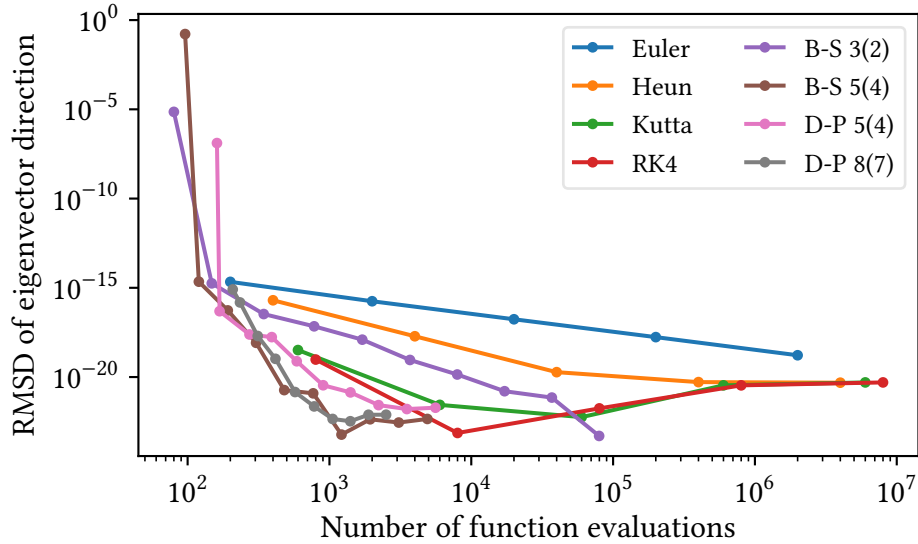
**Figure 4.19:** RMSD of the  $\lambda_2$  field as a function of the required number of function evaluations. For the adaptive stepsize methods, the *average* number of function evaluations across all advected tracers is used as the metric.

Figure 4.20 reveals that the RMSD of the eigenvector direction, as defined in equation (3.20), is negligible for all numerical step lengths. In fact, the numerical errors are comparable to, or even smaller than, the inherent double-precision floating-point number accuracy, which is of order  $10^{-16}$ . For this reason, the agreement with the anticipated power laws should not be taken as any kind of conclusive evidence. Figure 4.21, shows that this is in fact the case for the adaptive stepsize methods too, aside from the tolerance level  $\text{tol} = 10^{-1}$ , 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 this tolerance level, though, the error in eigenvector direction is comparable to, or smaller than, the floating-point accuracy. 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 driven just about entirely by the error in the computed eigenvalues.



**Figure 4.20:** RMSD of the strain eigendirections as a function of numerical step length, for the singlestep methods considered. Curves with slopes corresponding to the expected power law in error for the various methods are included in order to facilitate visual comparisons.



**Figure 4.21:** RMSD of the strain eigendirections as a function of the required number of function evaluations. For the adaptive stepsize methods, the *average* number of function evaluations across all advected tracers is used as the metric.

### 4.3 GENERAL REMARKS

First off, the computed LCS used as the reference, shown in figure 3.6, 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  $\text{tol} = 10^{-1}$ , shown in figure 4.5, differ greatly to the *correct* domain, which correlates well with the computed RMSD of the flow maps, shown in figure 4.17. In particular, the error for the aforementioned tolerance level is of order 1, which 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 4.2.3, 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 found by applying finite differences to the *main* tracers. The eigenvalues are essential 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 strain eigendirections are important too, of course — however, the error in the computed strain eigendirections is consistently negligible. This is most likely due to them being computed based on the denser *auxiliary* set of tracers, which per construction results in more accurate finite differences.

The double gyre model considered in this project is clearly not representative of a generic system, in terms of exact numerical step lengths or tolerance levels which are sufficient 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 timestep integration scheme, any single integration time step should not be so large that *too* much detail in the local and instantaneous velocity profile 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 skips over a grid cell entirely, when moving from one time level to the next.



When computing transport based on discrete data sets, such as snapshots of the instantaneous velocity profiles in oceanic currents, spatial and temporal interpolation becomes necessary. Together with the inherent precision of the measurement data, the choice of interpolation scheme(s) set a lower bound in terms of the accuracy 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.

#### 4.4 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 (2.12), 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 this property 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 inherent inaccuracy of double-precision floating-point numbers, 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 spacing  $\Delta x \simeq \Delta y \simeq 0.02$  was chosen, because it was the resolution 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, grid spacings 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 finite differencing. An entirely different approach, as suggested by Onu, Huhn, and Haller (2015), is to simply *define* the smaller strain eigenvalue as the reciprocal of the larger one, that is,  $\lambda_1 \equiv \lambda_2^{-1}$ . This was not done here, because the flow incompressibility has no obvious practical consequences for LCSs. In addition, subject to the accuracy of the numerical integration scheme, no tracer which starts out within the computational domain  $[0, 2] \times [0, 1]$  ever leave it, due to the normal component of the velocity field being zero along the domain edges. This follows from the definition of the associated velocity field, given in equation (3.1).

## 4.5 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, is that this necessitates an increase in memory usage, which has the practical consequence that less tracers can be advected at once. The immediately obvious workarounds are to either use a high performance scientific computer, or 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 leads to a more granular 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. However, 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 involve the dynamic introduction of increasingly finer grids in regions where the local velocities would have the largest Euclidean norm, for instance. 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 likelihood 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 scrapped. Regardless, this technique seems promising — at least on paper, which warrants further investigation.

## 4.6 ABOUT THE COMPUTATION OF STRAINLINES

### 4.6.1 *The special linear interpolation of strain eigendirections*

### 4.6.2 *The use of a single numerical integrator*

### 4.6.3 *The identification process of local strain maximizing strainlines*

- Approach does produce a consistent LCS picture for all numerical integrators considered, provided sensible integration steps or tolerance levels are chosen. This indicates that the calculation of this type of transport barrier is not particularly sensitive to the integration method.
- Most crucial part: Compute advection correctly. The error of the computed strain eigenvalues scales like the advection error, while the error in the strain eigendirections is negligible for all time steps. The computed LCSs seem to exhibit sensitive dependence on the calculation of the eigenvalues, which is not particularly

surprising, considering the LCS conditions of eq. 3.13

- The precision of the strain eigendirections is likely a direct consequence of the auxiliary tracers in their computation.
- The number of points in the  $\mathcal{U}$  domain is different to the one obtained by Haller et al, by approximately 3%. This, however, is likely related to how the grid of tracers was set up. In their paper, Farazmand and Haller do not provide details.
- Notably, the reference LCS as shown in figure (.) consists of seven different strainline segments, unlike the structure found by Haller et al., which is claimed to be a single coherent strainline.
- The time lengths or tolerance levels should be chosen based on the system under consideration. In particular, an individual time step should not be so large that too much detail in the local, instantaneous velocity field is glossed over.
- Possible approach in order to refine the computations: Use fewer ‘base’ tracers, and an *adaptive multigrid method*. Main con: May lead to inconsistent centered differencing when approaching the Jacobian etc. of the flow map
- Regarding the completely different domains obtained via the embedded methods for the highest tolerance level: Check if this corresponds th the error in the flow map larger than some threshold. If (likely) so, this is further evidence that one should exert great effort in computing the flow map correctly.
- In terms of application to discrete velocity data sets, where both spatial and temporal interpolation may be necessary, the interaction between interpolation and integration scheme is likely to have a great overall impact. In particular, the underlying interpolation scheme sets a lower accuracy bound for the entire advection process, practically enforcing restrictions on the numerical step lengths or tolerance levels which can be considered sensible. This is also based on the considered velocity field.
- Regarding the incompressibility of the velocity field
  - Incompressibility property is conserved until approximately  $t = 5$  units. Generally can’t expect this property to hold numerically over time.
  - There is no assurance that neighboring tracers will remain nearby after the advection, leaving the finite difference approximation of the local strain and stretch invalid. This issue is expected to be most prominent in regions of high

~~repulsion, which are precisely where accuracy is imperative.~~

- Parameter choices
  - Numerical step lengths and tolerance levels
  - The use of RK4 for all strainline iterations
    - Alternative: Use a high order adaptive step method, paying close attention to the strainline curvature
  - Represent tracer positions by means of higher precision floating-point numbers
    - Yields more accurate finite-difference approximations
    - Requires more memory, i.e., less tracers can be advected, leading to a more granular flow map representation. Perhaps most relevant for well-behaved velocity fields, for which the LCS approach is not really practically relevant — why not simply advect a few particles and see where they end up?
- The identification process of local strain maximizing strainlines
  - The cutting of strainline tails
  - Lines in  $\mathcal{L} \rightarrow$  not necessarily a robust approach for general flows
  - Alternative: Clustering algorithm
  - Approach where both the strainline lengths and avg  $\lambda_2$  are taken into consideration, i.e., selecting a longer line with slightly smaller avg  $\lambda_2$  over a shorter line with larger.
- The special linear interpolation used for the eigenvectors
  - Possible alternative: Higher order interpolation
- Chose mean error, rather than max, because:
  - In terms of flow map: Error should be limited from above, by the domain extent, as the normal component of the velocity at the boundaries is zero
  - Generally no way of telling where the maximum error occurs

- Regarding the use of RMSD:
  - A practical application of empirical standard deviation
  - The distribution of errors across the domain is unknown. From the Central Limit thm., however, the mean error is nearly distributed as a Gaussian.

## 5 Conclusions

---

## References

---

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

## A Haller er en dust

---

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est.



Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.