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 24, 2017

Abstract

Sammendrag

Preface

Table of Contents

List of Figures vi

List of Tables vii

Notation viii

1 Theory 1
- 1.1 Solving systems of ordinary differential equations . . . 1
  - 1.1.1 The Runge-Kutta family of numerical methods . . . 1
  - 1.1.2 The Runge-Kutta methods under consideration . . . 5
- 1.2 The type of flow systems considered . . . 11
- 1.3 Definition of LCSs for two-dimensional flows . . . 12
  - 1.3.1 Hyperbolic LCSs . . . 13
- 1.4 FTLE fields as predictors for LCSs . . . 14

2 Method 15
- 2.1 The double gyre model . . . 15
- 2.2 Advecting a set of initial conditions . . . 17
  - 2.2.1 Generating a set of initial conditions . . . 17
  - 2.2.2 On the choice of numerical step lengths and tolerance levels . . . 19
- 2.3 Calculating the Cauchy-Green strain tensor . . . 20
- 2.4 Identifying LCS candidates numerically . . . 20
  - 2.4.1 A framework for computing smooth strainlines . . . 21
  - 2.4.2 Extracting hyperbolic LCSs from strainlines . . . 22

3 Results 27

4 Introduction 27
- 4.1 Complex systems –> Need shortcuts . . . 27
- 4.2 Intuitively, what is an LCS? . . . 27
- 4.3 LCS definition . . . 27
- 4.4 Different types of LCSs . . . 27
- 4.5 Hyperbolic LCSs . . . 27

5 Theory 27
- 5.1 Solving ODE systems . . . 27
- 5.2 Flowmaps . . . 27
- 5.3 LCS definition . . . 27
- 5.4 FTLE as LCS predictor . . . 27
- 5.5 Identify hyperbolic LCS from variational theory . . . 28

6 Tool! 28

6.1 Adveksjon 28
6.2 CG tensors 28
6.3 Eigenvalues/Eigenvectors 28
6.4 Identify AB domain 28
6.5 Compute strainlines 28
6.6 Identify intersections 29
6.7 Identify neighbors 29
6.8 Select LCSs 29

7 Experiments **29**

References **30**

APPENDIX A Haller er en dust **31**

List of Figures

1.1 Geometric interpretation of the eigenvectors of the Cauchy-Green strain tensor 12
2.1 Illustration of the set of initial conditions 17
2.2 Illustration of the concept of auxiliary tracers 18
2.3 Illustration of the special linear interpolation used for the ξ_1 eigenvector field 22
2.4 The set $\mathcal{U}_0$ for the double gyre system 23
2.5 The identification process of strainlines which are local strain maximizers . . 25
2.6 Contour plots of the FTLE field and $\lambda_2(\mathbf{x}_0)$ distribution of the double gyre system 26

List of Tables

1.1 Butcher tableau representation of a general s-stage Runge-Kutta method . . . 2
1.2 Butcher tableau representation of general, embedded, explicit Runge-Kutta methods . . . 3
1.3 Butcher tableau representation of the explicit Euler method . . . 5
1.4 Butcher tableau representation of the explicit Heun method . . . 5
1.5 Butcher tableau representation of the explicit Kutta method . . . 6
1.6 Butcher tableau representation of the explicit, classical Runge-Kutta method . 6
1.7 Butcher tableau representation of the Bogacki-Shampine 3(2) embedded Runge-Kutta method . . . 7
1.8 Butcher tableau representation of the Bogacki-Shampine 5(4) embedded Runge-Kutta method . . . 8
1.9 Butcher tableau representation of the Dormand-Prince 5(4) embedded Runge-Kutta method . . . 9
1.10 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.:

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

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

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

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

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

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

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

1 Theory

1.1 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

$$\dot{x}(t) = f\Big(t, x(t)\Big), \quad x(t_0) = x_0, \tag{1.1}$$

where x is an unknown function (scalar or vector) of time t. The function f is defined on an open subset Ω of $\mathbb{R} \times \mathbb{R}^n$. 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 (1.1) is replaced by a family of equations

$$\begin{gathered} \dot{x}_i(t) = f_i\Big(t, x_1(t), x_2(t), \ldots, x_n(t)\Big), \quad x_i(t_0) = x_{i,0}, \quad i = 1, \ldots, n \\ \mathbf{x}(t) = \Big(x_1(t), x_2(t), \ldots, x_n(t)\Big) \end{gathered} \tag{1.2}$$

The system if nonlinear if the function f in equation (1.1), alternatively, if at least one of the functions f_i in equation (1.2), is nonlinear in one or more of its arguments.

1.1.1 *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 frequently used collection of implicit and explicit iterative methods, used in temporal discretization in order to obtain numerical approximations of the *true* solutions of systems like (1.1). 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 in the 2008 printing). The general scheme of what is now known as a Runge-Kutta method is as follows:

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

$$\begin{aligned} k_i &= f\left(t_n + c_i h, x_n + h \sum_{j=1}^{s} a_{i,j} k_j\right), \quad i = 1, \ldots, s \\ x_{n+1} &= x_n + h \sum_{i=1}^{s} b_i k_i \end{aligned} \tag{1.3}$$

is called an *s-stage Runge-Kutta method* for the system (1.1).

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.2 in the 2010 printing) define the *order* of a Runge-Kutta method as follows:

Definition 2. A Runge-Kutta method (1.3) is said to be of *order p* if, for sufficiently smooth systems (1.1),

$$\|x_{n+1} - x(t_{n+1})\| \leq Kh^{p+1}, \tag{1.4}$$

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 (1.3) symbolically, by means of mnemonic devices known as Butcher tableaus (Hairer, Nørsett, and Wanner 1993, p.134 in the 2008 printing). The Butcher tableau for a general s-stage Runge-Kutta method, introduced in definition 1, is presented in table 1.1.

Table 1.1: 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 in the 2010 printing). 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 in the 2008 printing). This is one of the reasons for the attention shift from the latter half of the 1950s and onwards, towards so-called *embedded* Runge-Kutta methods.

The basic idea of embedded Runge-Kutta methods is that they, aside from the numerical approximation x_{n+1}, yield a second approximation $\widehat{x}_{n+1}$. The difference between the two approximations then yields an estimate of the local error of the less precise result, which can be used for automatic step size control. 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 1.2.

Table 1.2: 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

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i k_i \tag{1.5a}$$

is of order p, and

$$\widehat{x}_{n+1} = x_n + h \sum_{i=1}^{s} \widehat{b}_i k_i \tag{1.5b}$$

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

In order to implement automatic step size control, the procedure suggested by Hairer, Nørsett, and Wanner (1993, pp.167–168 in the 2008 printing) was followed closely. A starting step size of $h = 0.1$ was used throughout. For the first solution step, the embedded integration method yields the two approximations x_1 and $\widehat{x}_1$, yielding the difference $x_1 - \widehat{x}_1$ as an estimate of the error of the less precise result. The idea is to enforce the error of the numerical solution to satisfy componentwise:

$$|x_{1,i} - \widehat{x}_{1,i}| \leq \mathrm{sc}_i, \qquad \mathrm{sc}_i = \mathrm{Atol}_i + \max\left(|x_{0,i}|, |x_{1,i}|\right) \cdot \mathrm{Rtol}_i \tag{1.6}$$

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,

$$\mathrm{err} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_{1,i} - \widehat{x}_{1,i}}{\mathrm{sc}_i}\right)^2} \tag{1.7}$$

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

$$h_{\mathrm{opt}} = h \cdot \left(\frac{1}{\mathrm{err}}\right)^{\frac{1}{q+1}}. \tag{1.8}$$

If $\mathrm{err} \leq 1$, 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 $\mathrm{err} > 1$, 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:

$$h_{\mathrm{new}} = \begin{cases} \min(\mathrm{fac}_{\max} \cdot h, \mathrm{fac} \cdot h_{\mathrm{opt}}), & \text{if the solution step is accepted} \\ \mathrm{fac} \cdot h_{\mathrm{opt}}, & \text{if the solution step is rejected} \end{cases} \tag{1.9}$$

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

1.1.2 *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^{st} order accurate, and requires a single function evaluation of the right hand side of the ordinary differential equation (1.1) or (1.2) at each time step. Its Butcher tableau representation can be found in table 1.3. It is the simplest explicit method for numerical integration of ordinary differential equations. The Euler method is often used as a basis to construct more complex methods, such as the Heun method, which is also known as the *improved Euler method* or the *explicit trapezoidal rule*. The Heun method is 2^{nd} order accurate, and requires two function evaluations at each time step. Its Butcher tableau representation can be found in table 1.4.

Table 1.3: Butcher tableau representation of the explicit Euler method.

0	
	1

Table 1.4: Butcher tableau representation of the explicit Heun method.

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

The Kutta method is 3^{rd} order accurate, and requires three function evaluations of the right hand side of the ordinary differential equation (1.1) or (1.2) at each time step. Its Butcher tableau representation can be found in table 1.5. The classical Runge-Kutta method is 4^{th} order accurate, and perhaps the most well-known and frequently used of the four singlestep schemes discussed in this project. One reason for its popularity is that it is exceptionally stable numerically (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 the 2008 printing) – which means that 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 e.g. by using 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 1.6.

Table 1.5: 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 1.6: 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 in the 2008 printing).

Butcher tableau representations of the aforementioned adaptive stepsize methods can be found in tables 1.7–1.10, 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 1.7: Butcher tableau representation of the Bogacki- Shampine 3(2) embedded Runge-Kutta method. The b coefficients correspond to a 3^{rd} order accurate solution used to continue the integration. The $\widehat{b}$ coefficients correspond to a 2^{nd} order accurate interpolant, which can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. The *First Same As Last* property is apparent from the fact that the b coefficients correspond exactly to the last row of coefficients in the Runge-Kutta matrix. For reference, see Bogacki and Shampine (1989).

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

Table 1.8: Butcher tableau representation of the Bogacki- Shampine 5(4) embedded Runge-Kutta method. The b coefficients correspond to a 5^{th} order accurate solution used to continue the integration. The two rows of $\widehat{b}$ coefficients correspond to two independent 4^{th} order accurate interpolants. They can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. The fact that two independent interpolants are included is a part of the reason for which the method nearly behaves like a 6^{th} order method (Hairer, Nørsett, and Wanner 1993, p.194 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 1.9: Butcher tableau representation of the Dormand-Prince 5(4) embedded Runge-Kutta method. The b coefficients correspond to a 5^{th} order accurate solution used to continue the integration. The $\widehat{b}$ coefficients correspond to a 4^{th} order interpolant, which can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. The *First Same As Last* property is apparent from the fact that the b coefficients correspond exactly to the last row of the Runge-Kutta matrix. For reference, see 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	$\frac{7571}{16695}$	$\frac{393}{640}$	$\frac{-92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Table 1.10: Butcher tableau for the Dormand-Prince 8(7) embedded Runge-Kutta method. The b coefficients correspond to an 8^{th} order accurate solution used to continue the integration. The $\widehat{b}$ coefficients correspond to a 7^{th} order interpolant, which can be used to estimate the error of the numerical approximation, and to dynamically adjust the time step. Note that the provided coefficients are rational approximations, accurate to about 24 significant decimal digits – sufficiently well-resolved for most numerical applications. For reference, see Prince and Dormand (1981).

0												
$\frac{1}{18}$	$\frac{1}{18}$											
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$										
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$									
$\frac{5}{16}$	$\frac{5}{16}$	0	$\frac{-75}{64}$	$\frac{75}{64}$								
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$							
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736358}{692538347}$	$\frac{-28693883}{1125000000}$	$\frac{23124283}{1800000000}$						
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$\frac{-180193667}{1043307555}$					
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$\frac{-433636366}{683701615}$	$\frac{-421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$				
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$\frac{-37695042795}{15268766246}$	$\frac{-309121744}{1061227803}$	$\frac{-12992083}{490766935}$	$\frac{6005943493}{2108947689}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$			
$\frac{1201146811}{1299019798}$	$\frac{-1028468189}{846180014}$	0	0	$\frac{8478235783}{508512851}$	$\frac{1311729495}{1432422823}$	$\frac{-10304129995}{1701304382}$	$\frac{-48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$\frac{-45442868181}{339846796}$	$\frac{3065993473}{5917172653}$		
1	$\frac{185892177}{718116043}$	0	0	$\frac{-3185094517}{667107341}$	$\frac{-477755414}{1098053517}$	$\frac{-703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$\frac{-4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$	
1	$\frac{403863854}{491063109}$	0	0	$\frac{-5068492393}{434740067}$	$\frac{-411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$\frac{-13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$\frac{-160528059}{685178525}$	$\frac{248638103}{1413531060}$	0
	$\frac{1400451}{335480064}$	0	0	0	0	$\frac{-59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{56129285}{797845732}$	$\frac{-1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$\frac{-528747749}{2220607170}$	$\frac{1}{4}$
	$\frac{13451932}{455176623}$	0	0	0	0	$\frac{-808719486}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$\frac{-3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$	0

1.2 THE TYPE OF FLOW SYSTEMS CONSIDERED

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

$$\dot{\mathbf{x}} = \mathbf{v}(t, \mathbf{x}), \quad \mathbf{x} \in \mathcal{U}, \quad t \in [t_0, t_1], \tag{1.10}$$

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, i.e., analytical solutions of system (1.10), may or may not be computed. The flow particles are assumed to be infinitesimall 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 (1.10), the flow map is defined as

$$\mathbf{F}_{t_0}^{t}(\mathbf{x}_0) = \mathbf{x}(t; t_0, \mathbf{x}_0), \tag{1.11}$$

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 (1.10) (Farazmand and Haller 2012). For sufficiently smooth velocity fields, the right Cauchy-Green strain tensor field is defined as

$$\boldsymbol{C}_{t_0}^{t}(\mathbf{x}_0) = \left(\nabla \mathbf{F}_{t_0}^{t}(\mathbf{x}_0)\right)^{*} \nabla \mathbf{F}_{t_0}^{t}(\mathbf{x}_0), \tag{1.12}$$

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

$$(\nabla \mathbf{f})_{i,j} = \frac{\partial f_i}{\partial x_j} \tag{1.13}$$

which, for our two-dimensional flow, reduces to

$$\nabla \mathbf{f} = \begin{pmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{pmatrix}. \tag{1.14}$$

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

$$\begin{gathered}\boldsymbol{C}_{t_0}^{t}(\mathbf{x}_0)\boldsymbol{\xi}_i(\mathbf{x}_0) = \lambda_i\boldsymbol{\xi}_i(\mathbf{x}_0), \quad \|\boldsymbol{\xi}_i(\mathbf{x}_0)\| = 1, \quad i = 1, 2, \\ 0 < \lambda_1(\mathbf{x}_0) \leq \lambda_2(\mathbf{x}_0),\end{gathered} \tag{1.15}$$

where, for the sake of notational transparency, the dependence of λ_i and $\boldsymbol{\xi}_i$ on t_0 and t has been suppressed. The geometric interpretation of equation (1.15) is that a fluid element undergoes the most stretching along the $\boldsymbol{\xi}_2$ axis, and the least along the $\boldsymbol{\xi}_1$ axis. This concept is shown in figure 1.1.

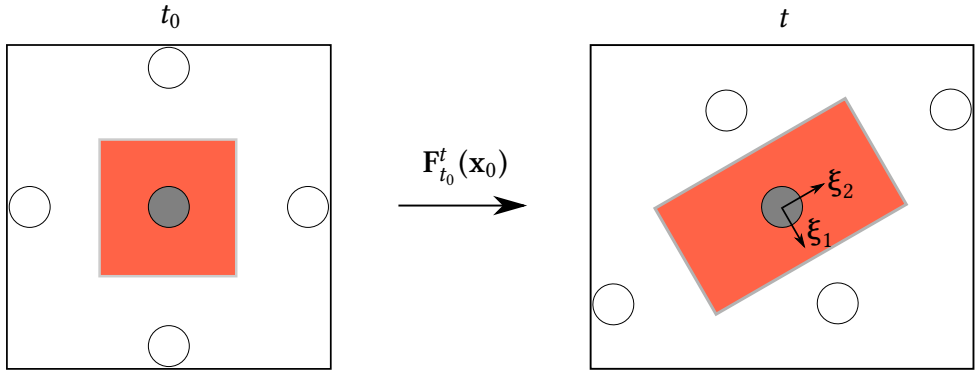

Figure 1.1: Geometric interpretation of the eigenvectors of the Cauchy-Green strain tensor. The central unit cell is stretched and deformed under the flow map $\mathbf{F}_{t_0}^{t}(\mathbf{x}_0)$. The local stretching is the largest in the direction of $\boldsymbol{\xi}_2$, the eigenvector which corresponds to the largest eigenvalue λ_2 of the Cauchy-Green strain tensor, defined in equation (1.15). Along the $\boldsymbol{\xi}_2$ and $\boldsymbol{\xi}_1$ axes, the stretch factors are $\sqrt{\lambda_2}$ and $\sqrt{\lambda_1}$, respectively.

Because the stretch factors along the $\boldsymbol{\xi}_2$ and $\boldsymbol{\xi}_1$ axes are given by the square root of the corresponding eigenvalues, for incompressible flow, the eigenvalues satisfy

$$\lambda_1(\mathbf{x}_0)\lambda_2(\mathbf{x}_0) = 1 \quad \forall \ \mathbf{x}_0 \in \mathcal{U} \tag{1.16}$$

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 begin massless, per definiton.

1.3 DEFINITION OF LCSS 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.

1.3.1 *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 (1.12), 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 (1.11), 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, over which 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 $\mathbf{x}_0 \in \mathcal{M}(t_0)$:*

$$\lambda_1(\mathbf{x}_0) \neq \lambda_2(\mathbf{x}_0) > 1 \tag{1.17a}$$

$$\left\langle \xi_2(\mathbf{x}_0), \boldsymbol{H}_{\lambda_2}(\mathbf{x}_0)\xi_2(\mathbf{x}_0) \right\rangle < 0 \tag{1.17b}$$

$$\xi_2(\mathbf{x}_0) \perp \mathcal{M}(t_0) \tag{1.17c}$$

$$\left\langle \nabla\lambda_2(\mathbf{x}_0), \xi_2(\mathbf{x}_0) \right\rangle = 0 \tag{1.17d}$$

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

$$(\boldsymbol{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}, \tag{1.18}$$

which, for our two-dimensional flow, reduces to

$$\boldsymbol{H}_f = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{pmatrix} \tag{1.19}$$

Condition (1.17a) ensures that the normal repulsion rate is larger than the tangential stretch due to shear strain along the LCS, as per definition 3. Conditions (1.17c) and (1.17d) suffice to ensure that the normal repulsion rate attains a local extremum along the LCS, relative to all nearby material lines. Lastly, condition (1.17b) forces this to be a strict local apex.

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

$$\sigma(\mathbf{x}_0) = \frac{\ln\left(\lambda_2(\mathbf{x}_0)\right)}{2|T|}, \tag{1.20}$$

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.

2 Method

In order to investigate the dependence of LCS identification by means of the variational approach as presented in section 1.3.1 on the choice of numerical integration method, outlined in section 1.1, 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 known.

2.1 THE DOUBLE GYRE MODEL

The double gyre model is defined as a pair of counter-rotating gyres, with a time-periodic perturbation. The perturbation can be interpreted as a solid, as in impenetrable, wall which oscillates periodically, which causes the gyres to periodically contract and expand. In terms of the cartesian coordinate vector $\mathbf{x} = (x, y)$, the system can be expressed mathematically as

$$\dot{\mathbf{x}} = \mathbf{v}(t, \mathbf{x}) = \pi A \begin{pmatrix} -\sin\left(\pi f(t,x)\right)\cos(\pi y) \\ \cos\left(\pi f(t,x)\right)\sin(\pi y)\dfrac{\partial f(t,x)}{\partial x} \end{pmatrix} \tag{2.1}$$

where

$$\begin{aligned} f(t,x) &= a(t)x^2 + b(t)x \\ a(t) &= \epsilon \sin(\omega t) \\ b(t) &= 1 - 2\epsilon \sin(\omega t) \end{aligned} \tag{2.2}$$

and the parameters A, ϵ and ω dictate the nature of the flow pattern. As in the literature, the parameter values

$$\begin{aligned} A &= 0.1 \\ \epsilon &= 0.1 \\ \omega &= \frac{2\pi}{10} \end{aligned} \tag{2.3}$$

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 (2.3).

Note that the velocity field $\mathbf{v}(t, \mathbf{x})$ in equation (2.1) can be expressed in terms of a scalar stream function:

$$\begin{aligned} \psi(t,\mathbf{x}) &= A \sin\Big(\pi f(t,x)\Big) \sin(\pi y) \\ \mathbf{v}(t,\mathbf{x}) &= \begin{pmatrix} -\dfrac{\partial \psi}{\partial y} \\ \dfrac{\partial \psi}{\partial x} \end{pmatrix} \end{aligned} \tag{2.4}$$

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

$$\nabla \cdot \mathbf{v}(t,\mathbf{x}) = -\frac{\partial^2 \psi}{\partial x \partial y} + \frac{\partial^2 \psi}{\partial y \partial x} = 0, \tag{2.5}$$

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 (1.16) to hold for the double gyre flow.

2.2 ADVECTING A SET OF INITIAL CONDITIONS

The variational model is based upon the advection of non-interacting tracers, which is described in section 1.2, by the velocity field defined in equation (2.1). 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 1.1.1. 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 1.1.2.

2.2.1 *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 1000×500 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 1004×504 grid points. The construction of the grid is illustrated in figure 2.1.

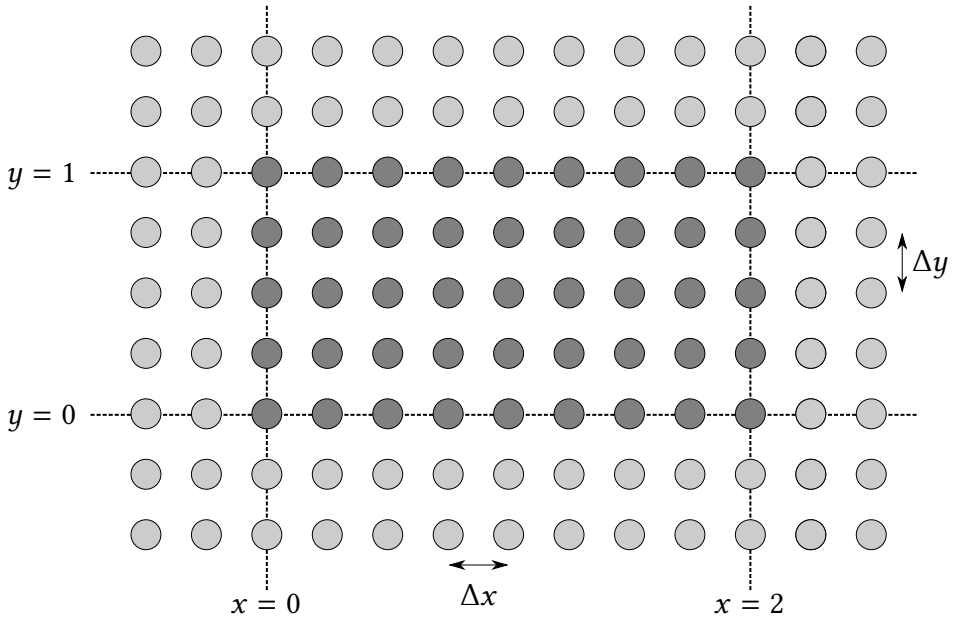

Figure 2.1: Illustration of the set of initial conditions. Dark grey blobs signify the main tracers, i.e., the tracers which discretize the computational domain $[0\ \ 2] \times [0\ \ 1]$. These were linearly spaced in either direction, with twice as many points in the x- direction as the y-direction, 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 (1.12). 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

$$\begin{aligned} \mathbf{x}^r_{i,j} &= (x_i + \delta x, y_j), \quad & \mathbf{x}^l_{i,j} &= (x_i - \delta x, y_j) \\ \mathbf{x}^u_{i,j} &= (x_i, y_j + \delta y), \quad & \mathbf{x}^l_{i,j} &= (x_i, y_j - \delta y) \end{aligned} \tag{2.6}$$

were assigned, where δx and δy are increments smaller than the grid spacings $\Delta x \simeq \Delta y$. Even though this effectively means that five times as many tracers have to be advected, the resulting accuracy in computing the Jacobian of the flow map, by means of the auxiliary tracers, is determined by the independent variables δx and δy. This, in principle, allows for much higher precision than what would be obtained by simply advecting five times as many *equidistant* tracers. The concept of the auxiliary tracers is illustrated in figure 2.2.

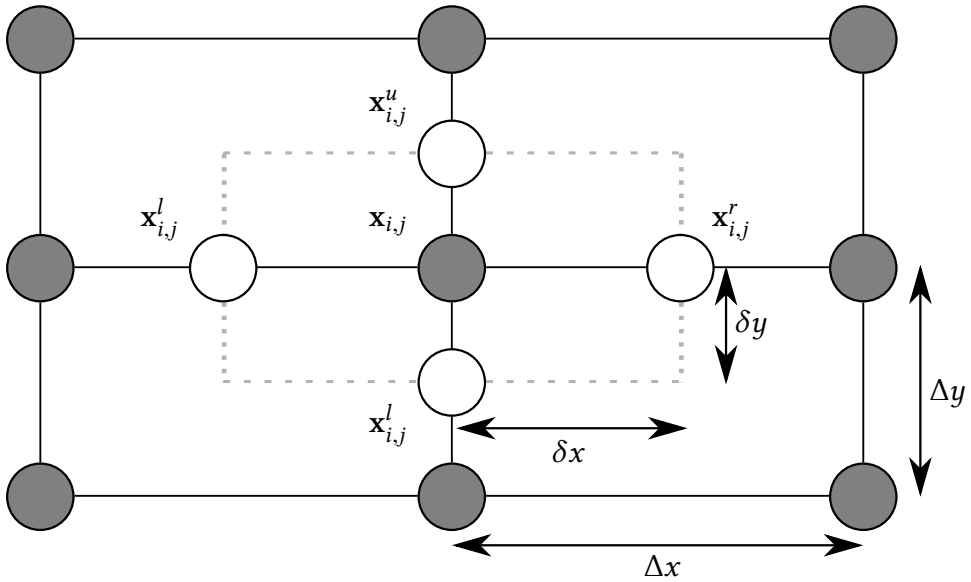

Figure 2.2: Illustration of the concept of auxilary tracers, used in order to compute the Jacobian of the flow map, and by extension, the Cauchy-Green strain tensor field, given by equation (1.12), more accurately. Grey blobs represent the original tracers, whereas white blobs represent the auxiliary ones.

Because of the limited number of decimal digits which can be accuractely represented by floating-point numbers, however, there is a strict lower limit to which it makes sense to lower the values of δx and δy. In particular, 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 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.

2.2.2 *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 integrators under consideration, the Euler method, is 1^{st} order accurate globally, meaning that its local error is of 2^{nd} 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}, i.e., 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.

2.3 CALCULATING THE CAUCHY-GREEN STRAIN TENSOR

Making use of the auxiliary tracer points, as outlined in section 2.2.1, the Jacobian of the flow map was approximated by means of centered differences as

$$\nabla \mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}) \approx \begin{pmatrix} \dfrac{\mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^r) - \mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^l)}{2\delta x} & \dfrac{\mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^u) - \mathbf{F}_{t_0}^t(\mathbf{x}_{i,j}^l)}{2\delta y} \end{pmatrix}. \tag{2.7}$$

The Cauchy-Green strain tensor was then calculated as per equation (1.12). 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 auxiliary 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 δx and δy, 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

$$\widetilde{\nabla \mathbf{F}}_{t_0}^t(\mathbf{x}_{i,j}) \approx \begin{pmatrix} \dfrac{\mathbf{F}_{t_0}^t(\mathbf{x}_{i+1,j}) - \mathbf{F}_{t_0}^t(\mathbf{x}_{i-1,j})}{2\Delta x} & \dfrac{\mathbf{F}_{t_0}^t(\mathbf{x}_{i,j+1}) - \mathbf{F}_{t_0}^t(\mathbf{x}_{i,j-1})}{2\Delta y} \end{pmatrix}. \tag{2.8}$$

The artificial grid extension, as outlined in section 2.2.1, 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 2.4. The numerical approximation of the eigen*values* of the Cauchy-Green strain tensor were thus calculated by means of the main tracers, equation (2.8), while the approximation of the corresponding eigen*vectors* were calculated by means of the auxiliary tracers, equation (2.7).

2.4 IDENTIFYING LCS CANDIDATES NUMERICALLY

Farazmand and Haller (2012) found the identification of the zeros of the inner product in equation (1.17d) of theorem 1 to be numerically sensitive. For this reason, they suggest a reformulated set of conditions which make for more robust numerical implementation, as follows:

$$\lambda_1(\mathbf{x}_0) \neq \lambda_2(\mathbf{x}_0) > 1 \tag{2.9a}$$

$$\left\langle \xi_2(\mathbf{x}_0), \boldsymbol{H}_{\lambda_2}(\mathbf{x}_0)\xi_2(\mathbf{x}_0) \right\rangle \leq 0 \tag{2.9b}$$

$$\xi_1(\mathbf{x}_0) \parallel \mathcal{M}(t_0) \tag{2.9c}$$

$$\begin{aligned} &\overline{\lambda}_2, \text{ the average of } \lambda_2 \text{ over a curve } \gamma, \text{ is maximal on } \mathcal{M}(t_0) \\ &\quad \text{among all nearby curves } \gamma \text{ satisfying } \gamma \parallel \xi_1(\mathbf{x}_0) \end{aligned} \tag{2.9d}$$

The relaxation of condition (1.17b) in theorem 1 from strict inequality to condition (2.9b), which also allows equality, means that LCSs are allowed to have finite thickness. However, the set of criterions (2.9) enforce that all LCSs have uniquely defined local orientations. Conditions (1.17c) and (2.9c) are equivalent, due to the orthogonality of the eigenvectors $\xi_1(\mathbf{x}_0)$ and $\xi_2(\mathbf{x}_0)$, although the form (2.9c) turns out to be more advantageous for use in computations.

Because of the artificial extension of the main computational grid, as outlined in section 2.2.1, 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 2.3. 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.

2.4.1 *A framework for computing smooth strainlines*

Per condition (2.9c), hyperbolic LCSs are composed of material curves tangent to the $\xi_1(\mathbf{x}_0)$ vector field, i.e., the eigenvector field associated with the smaller eigenvalue field $\lambda_1(\mathbf{x}_0)$ of the Cauchy-Green strain tensor field $\boldsymbol{C}_{t_0}^{t}(\mathbf{x}_0)$. The tensor lines tangent to the ξ_1-field will be referred to as *strainlines* in the following, a term coined by Farazmand and Haller (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

$$\mathbf{r}' = \xi_1(\mathbf{r}), \quad \mathbf{r} \in \mathcal{U}, \quad \|\xi_1(\mathbf{r})\| = 1. \tag{2.10}$$

As pointed out by Onu, Huhn, and Haller (2015), the orientational discontinuities of the ξ_1-field are removable, through careful monitoring and local reorientation. This process 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 ξ_1 vectors of adjacent grid points. Rotations exceeding 90° 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°. In the end, linear interpolation is used within the grid element.

Furthermore, should the ξ_1-vector obtained from the local special- purpose linear interpolation outlined above prove to be rotated by more than 90° relative to the ξ_1-vector from the previous level of pseudotime used in the numerical integration of system (2.10), it would be flipped 180°, 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 2.3.

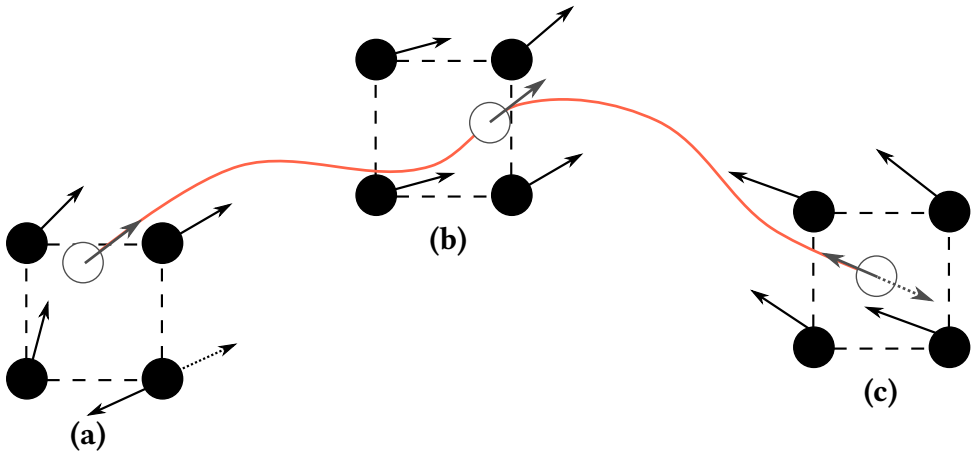

Figure 2.3: Illustration of the special linear interpolation used for the ξ_1 eigenvector field. At point **(a)**, there is an orientational discontinuity at the lower right grid point, which is corrected by rotating the corresponding vector by 180° prior to linear interpolation. At point **(b)**, there is no orientational discontinuity. Lastly, at point **(c)**, the interpolated vector must be flipped due to the overall orientation of the trajectory.

So, in order to compute globally smooth strainlines, equation (2.10) is altered in the following way:

$$\mathbf{r}'(s) = \operatorname{sgn}\left(\left\langle \mathbf{f}\big(\mathbf{r}(s)\big), \mathbf{r}'(s-\Delta)\right\rangle\right) \mathbf{f}\big(\mathbf{r}'(s)\big), \tag{2.11}$$

where $\mathbf{f}$ denotes the special-purpose local linear interpolation of the ξ_1 field, as outlined above and in figure 2.3, Δ is the numerical step length used in the numerical integration, while signum function is defined as

$$\operatorname{sgn}(x) = \begin{cases} 1, & \text{for } x > 0 \\ 0, & \text{for } x = 0 \\ -1, & \text{for } x < 0 \end{cases} \tag{2.12}$$

2.4.2 *Extracting hyperbolic LCSs from strainlines*

If a material line $\mathcal{M}(t_0)$ lies within a strainline, it automatically fulfills condition (2.9c). The segments of the strainlines on which the remaining conditions (2.9a), (2.9b) and (2.9d) 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 (2.9a) and (2.9b) are satisfied, and then integrate the system given by equation (2.11) 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 ξ_1 vector field.

Using all of the points in the $\mathcal{U}_0$ domain would inevitably involve computing a lot of strainlines several times over. In order to reduce the number of redundant calculations, the set of considered strain initial conditions were reduced, with the approach suggested by Farazmand and Haller (2012). In particular, the set $\mathcal{U}_0$ was reduced to its intersections with four horizontal and four vertical lines. The set $\mathcal{U}_0$ for the double gyre system, as well as the aforementioned intersections, are illustrated in figure 2.4. 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 $\mathcal{U}_0$.

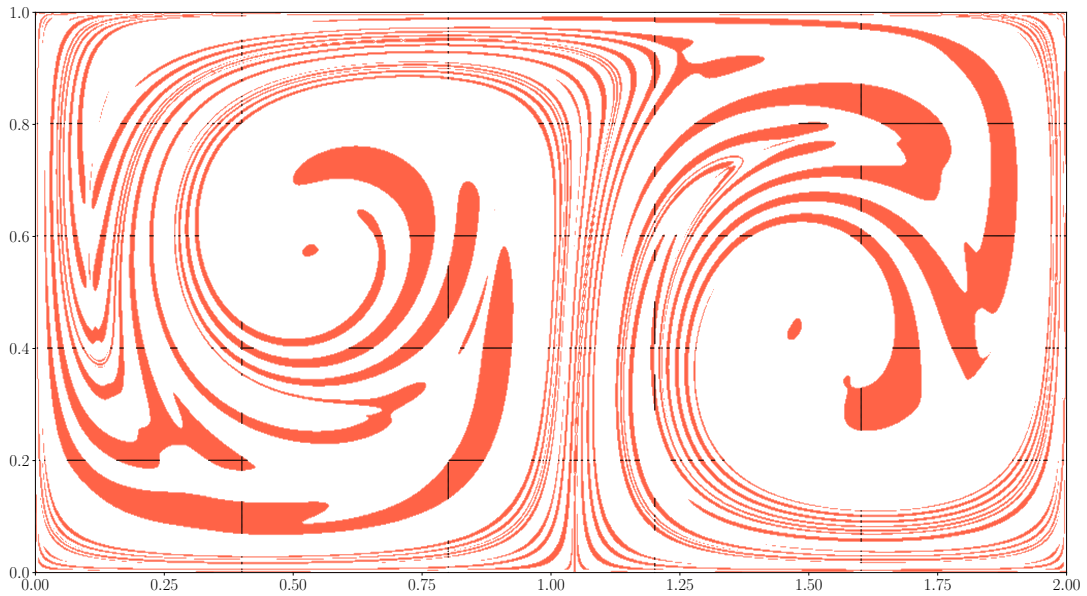

Figure 2.4: The set $\mathcal{U}_0$ for the double gyre system, given by equations (2.1) and (2.3), is shown in red. Its itersection with a set of four horizontal and four vertical, equidistant lines is shown in black. The latter was used as the set of strain initial conditions, in order to eliminate redundant computations of strainlines within $\mathcal{U}_0$.

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

$$\alpha(\mathbf{x}_0) = \left(\frac{\lambda_2(\mathbf{x}_0) - \lambda_1(\mathbf{x}_0)}{\lambda_2(\mathbf{x}_0) + \lambda_1(\mathbf{x}_0)}\right)^2 \tag{2.13}$$

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 2.4, 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° rotated ξ_1 vector field in the system given by equation (2.11). 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 2.2. 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 2.2.2; 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 (2.9a) or (2.9b) 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 continious 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 (2.9a) and (2.9b), the next step is imposing condition (2.9d), 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 $\overline{\lambda}_2(\gamma_0)$, the average of λ_2 on the curve γ_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 2.5.

Intersections between strainlines and the lines in $\mathcal{L}$ are found by means of linear interpolation. Should a strainline segment prove to be a local maximizer along at least one of the lines in $\mathcal{L}$, the strainline segment is labelled as a LCS. Adjacent intersections who are separated by a distance larger than a preselect threshold are excluded from the local maximization process, as indicated in figure 2.5. 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

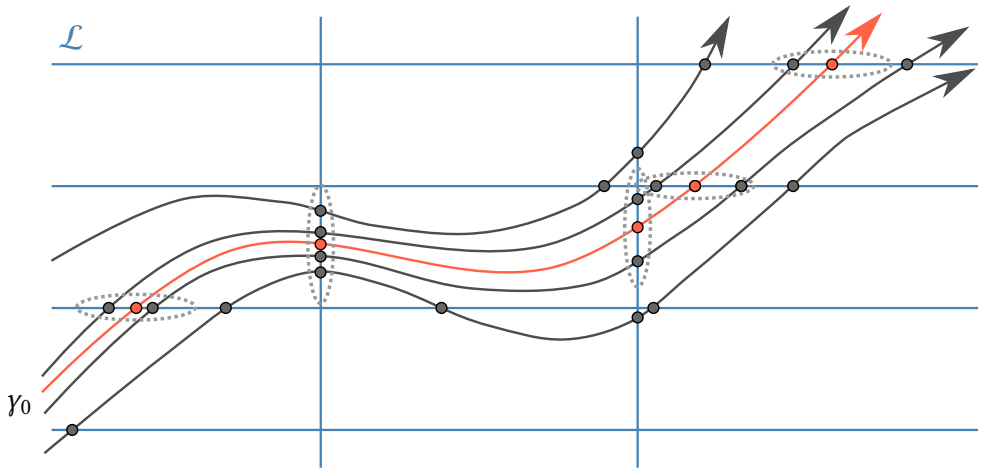

Figure 2.5: Illustration of the identification process of strainlines which are local strain maximizers. Local apices of $\overline{\lambda}_2(\gamma)$, the average of λ_2 over a curve γ, are found along a set $\mathcal{L}$ of horizontal and vertical lines. The strainline segment γ_0 is identified as a local maximizer if it contains a locally maximal $\overline{\lambda}_2$ value along at least one of the lines in $\mathcal{L}$. 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 γ_0.

shorter than the pre-set small length $l_{\text{min}} = 1$ were discarded as LCS candidates, as suggested in Farazmand and Haller (2012).

One shortcoming in the work of Farazmand and Haller (2012), is that they do not provide a 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 unambigously 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 2.4) in relation to the $\lambda_2(\mathbf{x}_0)$ distribution, as well as the FTLE field (described in section 1.4), where the latter two are shown in figure 2.6.

By inspection of figure 2.4, 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 2.6, 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 horizontal lines, given by $x = 0.15$, $x = 0.15$, $y = 0.05$ and $y = 0.95$, respectively, proved sufficient in order to accurately reproduce the same LCS as found in Farazmand and Haller (2012). We can only assume that Farazmand and Haller must have done something similar, as they have been unavailable for communication. The aforementioned LCS is presented in chapter 3, together with the approximations found by means of the different integrators mentioned in section 1.1.2, for the parameters mentioned in section 2.2.2.

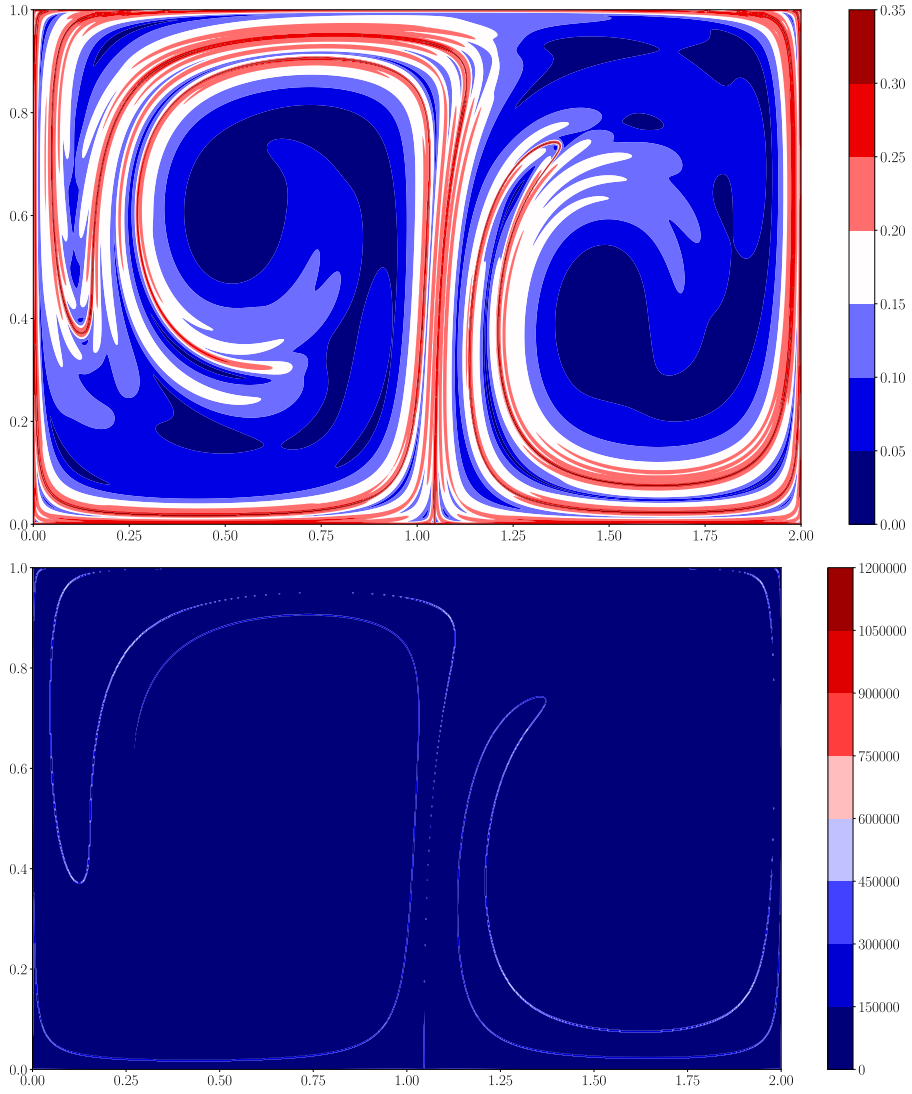

Figure 2.6: Contour plots of the FTLE field (top) and $\lambda_2(\mathbf{x}_0)$ (bottom) distribution of the double gyre system, given by equation (2.1). Due to the different scalings, different levels of detail are resolved. Most notably, the $\lambda_2(\mathbf{x}_0)$ contains a thin, patchy ridge of extremal values, whereas a similar, albeit continious 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 2.4, in order to select the lines in $\mathcal{L}$, illustrated in figure 2.5.

3 Results

4 Introduction

Only interested in hyperbolic LCSs as they are the only ones relevant for transport barriers.

4.1 Complex systems –> need shortcuts

4.2 Intuitively, what is an LCS?

4.3 LCS definition

4.4 Different types of LCSs

4.5 Hyperbolic LCSs

–> Connect to application

5 Theory

5.1 Solving ODE systems

—> General ODE systems —> Numerical integrators dump –> Interpolation necessary for discrete systems

5.2 Flowmaps

—> Introduce system and limitations –> Introduce the concept of a flow map

5.3 LCS definition

–> Different kinds of LCSs (hyperbolic, elliptic and parabolic, cf. LCS tool) –> More mathematical definitions? Ask Thör

5.4 FTLE as LCS predictor

–> Prone to false positives and negatives –> Definition somewhat arbitrary (what is a ridge)? –> Strogatz' motivation as a simple explanation of why we consider it at all?

5.5 IDENTIFY HYPERBOLIC LCS FROM VARIATIONAL THEORY

–> Mathematically involved.

6 Tool!

6.1 ADVEKSJON

–> Si noe om system, glatte vektorfelt/hastighetsfelt –> Integrasjonsteknikker

6.2 CG TENSORS

–> Auxiliary grid –> Extended grids i fire retninger –> Beregn CG tensors –> Centered differencing, consistently for all main particles –> Har med gitterpunkter på utsiden av hoveddomenet for å inkludere diskontinuitet i oppførsel i hastighetsfeltet

6.3 EIGENVALUES/EIGENVECTORS

–> Auxiliary grid –> Laplacian, extended grid layer 2 for centered differencing

6.4 IDENTIFY AB DOMAIN

–> Klargjør måten vi tolket Laplacian på

6.5 COMPUTE STRAINLINES

–> Define G0 along vertical and horizontal lin –> Avoid redundant computations of trajectories –> Integrate forwards and backwards —-> (Notice that strainlines "fall out" of AB domain, likely due to num. error) –> Special linear interpolation with local direction correction –> Higher order spline interpolations are inappropriate because of oriental –> discontinuities (in case of vectors) and great variance (in case of evals) –> Linear spline interpolation without orientation fix caused random turns –> at discontinuities. –> Stop criteria –> Alpha scaling introduced by Haller gave unpredictable leaps —–> After linear interp? –> Used just one integrator here, because [...] –> Choice of integration step (needs test!) –> Note that this step is very sensitive to the flow map details, –> components in the strain tensors down to the $10^{-}15$ level. –> LCS results sensitive to continious failure length, needed to increase —–> it in order to replicate results from Haller due to different AB domain

6.6 IDENTIFY INTERSECTIONS

–> Which lines and why (maximize intersections with as few lines as possible) –> Include all intersetions between a strainline and a vert / horz linear

6.7 IDENTIFY NEIGHBORS

–> Neighbor length essential for LCS results

6.8 SELECT LCSS

—> Identify LCS as local maxima of λ_2 which are also long enough –> Needs at least one neighbor other than itself –> **Cut tail of strainlines which exit AB domain and do not return** –> That part is no LCS! –> Parts/sections of strainlines may qualify as LCSs

7 Experiments

–> What did we try and why?

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.

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.

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, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer 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, consectetuer 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 consectetuer.

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.