NTNU
Norwegian University of
Science and Technology

Identifying Lagrangian Coherent Structures Using Clustering

Andreas Bjerke Skagestad

Master of Science in Physics and Mathematics
Submission date: June 2018
Supervisor: Tor Nordam, IFY

Norwegian University of Science and Technology
Department of Physics

Abstract

In these times of rapid environmental changes, our understanding of nature becomes ever more crucial. For example, the spread of oil spills and microplastics can have far reaching consequences for the ecology. Previous attempts at explaining the dynamics of ocean currents have been comprised of e.g. studying complicated and tangled plots of trajectories. Recent developments linking non-linear dynamics with fluid mechanics have led to the emergence of Lagrangian Coherent Structures (LCSs). LCSs focus on the existence of repelling and attracting material lines which govern the dynamics of flow fields. Farazmand and Haller (2012) proposed a computational approach to identify LCSs from their variational theory. An important part of this method was the development of material lines, and the identification of LCSs among these lines. In order to identify the LCSs, the concept of locally most repelling and attracting must be defined. The work in this thesis builds on that of Farazmand and Haller (2012) and aims at developing an alternative, more objective method for selecting the locally most repelling and attracting lines based on clustering.

Clustering is a class of unsupervised machine-learning algorithms which uncovers underlying patterns in the data. Applying clustering to the set of material lines would result in groups of similar lines. Then the most repelling or attracting material line from each of the clusters would be classified as the LCSs of the system.

This approach was tested on an analytically defined test case known as the double gyre, and a set of modelled ocean data[1]. Three different clustering algorithms were tested; DBSCAN, Agglomerative-clustering, and Affinity propagation. Additionally, different methods for dimensional reduction were also tested to see if the set of lines could be represented in a more compact way. The LCSs classified in the double gyre field matched the results from Farazmand and Haller (2012). The resulting set of LCSs calculated for the modelled data is also presented. In order to test the validity of the uncovered LCSs, they were advected by the flow to study their effect on nearby trajectories. The approach which gave the most promising results was to first calculate the pairwise similarities between lines, and then to use Affinity propagation to cluster the matrix comprised of these distances. The Fréchet distance was used to measure the pairwise similarity. This approach seemed to consistently group the lines such that a set of LCSs which explained the underlying dynamics of the field, were found.

[1]The code used in this project can be found on GitHub, `https://github.com/andreasskag/LCS-identification`

Acknowledgements

I would like to thank my supervisor, Tor Nordam, for his exceptional guidance. I would also like to thank Milla M. T. Fjellestad for proofreading my work and for all of her support during these trying times. Finally, I want to thank my parents, Eva Bjerke & Geir Skagestad, for teaching me the value of hard work.

A.B.S.

Contents

1 Introduction

The study of flow fields is important for forecasting many phenomena that occur around the world, both natural and man-made. Having accurate models to describe the world around us enables us to make predictions about the future. However, this can be rather difficult, as the dynamics of flow fields can be incredibly complex[1]. One example where it is possible to use a model to portray a real life scenario is the study of ocean currents. Through an understanding of the dynamics of ocean currents one may be able to contain environmental catastrophes. A classic example of this is the Deepwater Horizon oil spill of 2010 wherein the lack of accurate tools for predicting the spread of the plume hindered effective countermeasures and also led to major economical damages to the tourism industry along the Southwest Florida coastline.[2]. This was the largest marine oil spill in history and a major environmental disaster. The oil wound up along the American coastline, reaching as far as Florida. Three years after the accident, contaminants could still be detected on the beaches of Louisiana and Alabama[3, 4].

The standard procedure for studying ocean velocity fields and making predictions have been to study the trajectories of tracer particles advected by the flow. This is an arduous task which results in having to analyze intricate plots of tangled trajectories from a large ensemble of particles. Additionally, the path a particle will travel is also sensitive to its initial conditions[5]. Again, the Deepwater Horizon oil leakage can be used as an example. A simplified view of the oil leakage can be to view the oil as particles emerging from a point-source. Modelling this scenario using tracer particles would then require seeding the field at discrete time-steps. The timing of when to release the particles will affect the trajectory they travel and the same goes for the choice of initial position. To improve on this model, an approach linking non-linear dynamics with fluid mechanics have led to the emergence of Lagrangian Coherent Structures (LCSs)[1].

The theory of LCSs focuses on the existence of repelling and attracting material lines that govern the dynamics of the flow[5]. The locally most repelling and attracting lines are classified as LCSs. Several advances have been made in recent years to uncover this hidden skeleton of material lines. Haller (2011) developed a variational theory that clarified the relationship between LCSs and the Cauchy-Green (CG) strain tensor[6]. The CG strain tensor quantifies the ideal deformation caused by the flow. Farazmand and Haller (2012) later proved the necessary conditions for classifying LCSs and developed an algorithm for calculating the LCSs of various flow fields[7]. An important part of this method is the development of material lines and the process of identifying the LCSs among these lines.

There are different methods for identifying the LCSs of a system. One of the key points is to define the concept of locally most repelling and attracting. The work in this project builds on that of Farazmand and Haller (2012) and aims at developing an alternative, more objective method for selecting the locally most attracting and repelling lines based on clustering.

Clustering is a group of machine-learning algorithms used for identifying groups of objects in a dataset[8]. It is a form of unsupervised learning, which means that it does not require any labelling of the data[9]. Recognizing patterns in data and grouping it accordingly is a usually a trivial task for a human, but can be exceedingly hard to do for a computer. At least this is true for cases limited to a maximum of three dimensions[10]. A good example of where humans excel are recognizing handwritten digits. Even if the numbers are barely legible or the background is cluttered, most people would be able to read it. Trying to create a program to discern the different digits used to be a very difficult task. With the new awakening of machine-learning comes new tools for dealing with such problems. Reading handwritten digits can be viewed as a classification problem. For example, if the numbers ranged from 0 to 9, then there would be ten classes in the data[11]. The program should therefore be able to take a new digit, and put it into one of the existing clusters, thereby identifying which number it is. There are several clustering algorithms, each with different strengths and weaknesses. In this project, multiple algorithms were tested in order to find those best suited for the nature of the data.

In the task of identifying LCSs, the uncovered material lines can consist of thousands of points. When doing clustering, each line is considered a point in a high dimensional space, where each dimension is considered a feature of the line. The dimensionality increases the complexity of the problem and can result in important features of the data being drowned. Therefore different schemes for dimensional reduction and feature extraction were also tested.

This paper consists of three parts. The first section introduces the theory of Lagrangian Coherent Structures and the previous approaches for identifying the LCSs. Then the groundwork regarding the new proposed method for identifying LCSs is presented. This includes some theoretical background on the subject of dimensional reduction, using both linear and nonlinear combinations of features. Some examples of clustering problems and algorithms are also presented, including the algorithms used in this project. The second section applies the theory on the double gyre velocity field, explaining the procedure. This is a standard velocity field for testing LCS theories[1]. The final part of the paper tests the different approaches for LCS identification by means of clustering on modelled ocean data. This is compared with the results from the double gyre field. Additionally, the effectiveness of these approaches are discussed with testing on an ensemble of particles, before presenting the concluding remarks.

2 Theory

Lagrangian Coherent Structures (LCSs) govern the dynamics of a flow field[5]. The understanding of these dynamics are important for accurate forecasting of many phenomena, for example the ramification of oil spills or the spread of radioactive fallout. In figure 1 we see a grid of particles at four different times being advected by a velocity field. Through inspection we can see intricate patterns forming, dictating the dynamics of the flow. However, trying to explain the hidden skeleton of

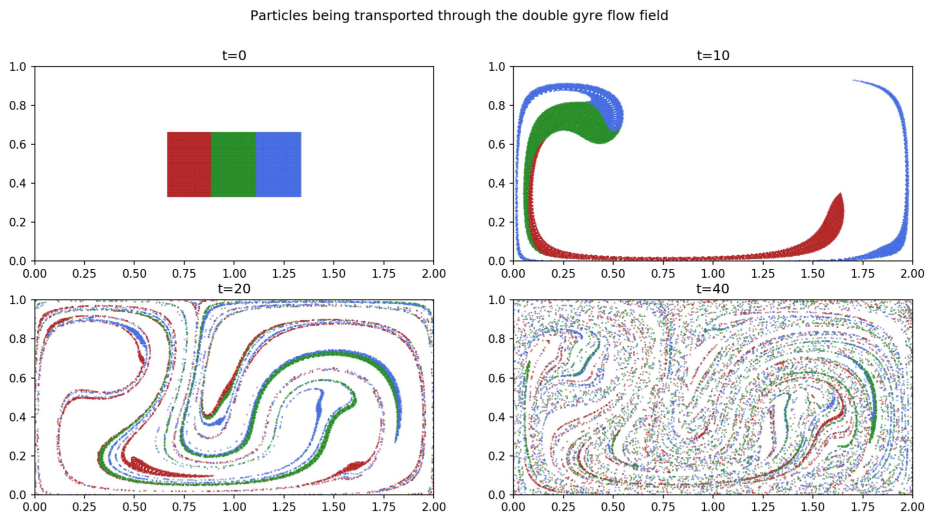


Figure 1: A rectangle of particles are being advected by the double gyre velocity field. First the blocks are only stretched out before mixing starts. As time goes on any resemblance to the original pattern is removed and only chaos remains.

the flow by studying the individual trajectories is not recommended[1]. The tracer particles follow chaotic paths and are highly sensitive to their starting positions. There is also the added complexity of the plots when considering multiple particles at the same time as paths intertwine with each other. This is shown in figure 2. In figure 2 (c) one of the particles from (a) is shown in the reference frame of the other. This changes the relative trajectory. A theory explaining the dynamics of the flow-field should be objective as the dynamics do not depend on the reference frame in which it is studied.

In this paper only two dimensional fields are considered. The first step to finding the material lines is to advect tracer particles through the field by solving the ordinary differential equation (ODE)

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

where $\mathbf{x} = (x_1, x_2)$ is the position of a particle at time t and $\mathbf{v}(\mathbf{x}, t)$ is the velocity given by the field. This equation can be solved numerically by using one of the standard integration schemes for ODEs.

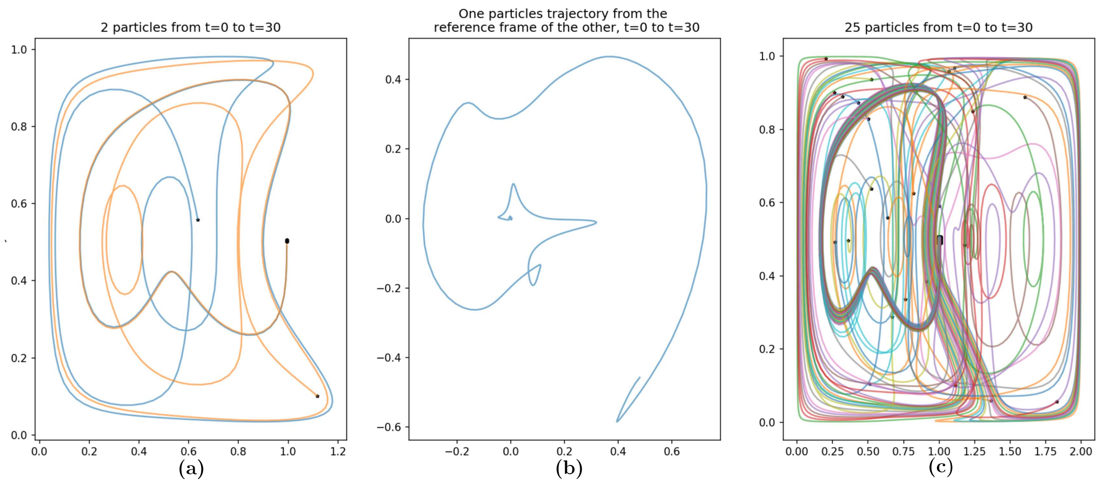


Figure 2: The first plot presents the trajectories of two particles initialized with a tiny separation. As they are advected by the field, the trajectories become more complicated. (b) shows how the reference frame affects the trajectory. Finally (c) illustrates how the plot quickly becomes less manageable as more particles are included.

2.1 Numerical Integration of ODE's

Solving an ODE such as eq. (1) means to find $\mathbf{x}(t)$. The first step in solving this equation numerically is to discretize time,

$$t_n = t_0 + n\Delta t. \tag{2}$$

Δt is a small timestep. We introduce the notation $\mathbf{x}_n$ which is the numerical approximation to the true solution at time t_n, $\mathbf{x}(t_n)$. Note that $\mathbf{x}(t_n)$ is generally not known. The global error is then given by $E = |\mathbf{x}_n - \mathbf{x}(t_n)|$. An important class of methods for solving ODEs with given initial values is the Runge-Kutta methods, which are given on the general form [12]

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{3}$$

where b_i are called the weights and $\mathbf{k}_i$ are the velocity evaluated at different positions and times. s is the number of stages used in the method. For a fourth-order method $s = 4$, but for a fifth-order method six stages are needed[12]. The case were $s = 1$ and $b_1 = 1$ is the standard Eulers method. This method is 1st-order in time. While it requires less functional evaluations for each step, the global error scales as $E \propto \mathcal{O}(\Delta t)$. Therefore a much smaller Δt is required for the same level of accuracy as the higher-order methods. In this project, the second-order method called Heun's method and the fourth-order Runge-Kutta (RK4) method are used. Heun's method is given by

$$\begin{aligned}
\mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{\Delta t}{2}(\mathbf{k}_1 + \mathbf{k}_2), \\
\mathbf{k}_1 &= \mathbf{v}(\mathbf{x}_n, t_n), \\
\mathbf{k}_2 &= \mathbf{v}(\mathbf{x}_n + \Delta t \mathbf{k}_1, t_{n+1}).
\end{aligned} \tag{4}$$

The global error, E for Heun's scheme scales as $\mathcal{O}(\Delta t^2)$. The RK4 utilizes more terms than Heun's and will therefore require more function calls to be evaluated. The RK4 is given by

$$\begin{aligned}
\mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4),\\
\mathbf{k}_1 &= \mathbf{v}(\mathbf{x}_n, t_n),\\
\mathbf{k}_2 &= \mathbf{v}(\mathbf{x}_n + \frac{\Delta t}{2}\mathbf{k}_1, t_n + \frac{\Delta t}{2}),\\
\mathbf{k}_3 &= \mathbf{v}(\mathbf{x}_n + \frac{\Delta t}{2}\mathbf{k}_2, t_n + \frac{\Delta t}{2}),\\
\mathbf{k}_4 &= \mathbf{v}(\mathbf{x}_n + \Delta t\mathbf{k}_3, t_n + \Delta t).
\end{aligned} \tag{5}$$

For the RK4 method, the error scales as $\mathcal{O}(\Delta t^4)$, hence a longer timestep Δt can be used, resulting in fewer steps overall.

2.2 The Finite-Time Lyapunov Exponent (FTLE)

For many common transport problems, the trajectory of a particle through the velocity field is sensitively dependent on initial position. Consider two particles starting at $\mathbf{x}_0$ and $\mathbf{x}_0 + \delta_0$, where δ_0 is a small separation. This separation may grow in time and is assumed proportional to [13]

$$|\delta(t)| \approx |\delta_0| e^{\sigma t}, \tag{6}$$

where σ is the largest Lyapunov exponent. If $\sigma > 0$ the flow is said to be chaotic. For $\sigma \leq 0$ the particles will either travel with a constant separation, or they will converge until the separation is essentially zero.

An approach to analyzing a flow field is to cover the field by a grid of tracer particles, and let these be transported by the flow. The Lyapunov exponents are calculated for all particles over a finite time-interval to reveal the FTLE-field,

$$\sigma = \frac{1}{T}\log(\frac{|\delta(t)|}{|\delta_0|}) = \frac{1}{T}\log(\sqrt{\lambda}), \tag{7}$$

where T is the period, the separation fraction, $|\delta(t)|/|\delta_0|$ is equal to the root of the largest eigenvalue of the Cauchy-Green strain tensor, λ. σ is known as the Finite-Time Lyapunov exponent. This field can be used to identify regions with strong shear stress. Early attempts at LCS detection linked LCSs to the ridges of the FTLE field[14]. The FTLE field can however produce false positives and false negatives when detecting LCSs, and should therefore be used with caution[5]. The reason for the ambiguity of the FTLE field is that it ignores the direction $\boldsymbol{\zeta}_i$ of the largest stretching at $\mathbf{x}_0$. Some attempts have been made to work around this problem, for example by defining the LCS as second derivatives of the FTLE field. Second derivative ridges of the FTLE field seldom exist in generic fluid flow which makes the approach a dead end[5].

2.3 Lagrangian Coherent Structures

Lagrangian Coherent Structures (LCSs) act as organizers of flow fields. As the fields evolve, so do the LCSs, which due to their Lagrangian nature are transported with the flow. Identifying these structures can be used to understand phenomena such as the Great Red Spot on Jupiter or complex patterns present in the oceans here on Earth[1].

The variational theory of hyperbolic LCSs developed by Haller (2011) defines LCSs in terms of the Cauchy-Green (CG) strain tensor, which quantifies deformation from the flow[6]. The first step of identifying LCSs in flow fields is therefore to study the deformation caused by the flow over the time interval of interest.

The position of a tracer particle in the field at time t is $\mathbf{x}_t = \mathbf{x}(t; t_0, \mathbf{x}_0)$, where $\mathbf{x}_0$ is the position at the initial time t_0. Next we define the flow-map as the function that sends $\mathbf{x}_0$ to $\mathbf{x}_t$,

$$F_{t_0}^t(\mathbf{x}_0) \equiv \mathbf{x}_t. \tag{8}$$

For any smooth velocity field, the flow-map $F_{t_0}^t$ will also be smooth, and the gradient $\nabla F_{t_0}^t$ will be a positive definite, invertible matrix[5]. This allows us to define the Cauchy-Green (CG) strain tensor field as

$$[\nabla F_{t_0}^t(\mathbf{x}_0)]^\dagger \nabla F_{t_0}^t(\mathbf{x}_0) = C_{t_0}^t(\mathbf{x}_0). \tag{9}$$

$C_{t_0}^t(\mathbf{x}_0)$ is the CG strain tensor and is symmetric and positive definite. This means that the eigenvalues are real and positive and that the eigenvectors are orthogonal. For d dimensions we have the following relations

$$\begin{aligned} C_{t_0}^t \boldsymbol{\zeta}_i &= \lambda_i \boldsymbol{\zeta}_i, \quad 0 < \lambda_1 \leq ... \leq \lambda_d, \\ |\boldsymbol{\zeta}_i| &= 1, \\ i &= 1, .., d. \end{aligned} \tag{10}$$

Here λ_i is the i-th eigenvalue corresponding to the i-th eigenvector, $\boldsymbol{\zeta}_i$.

Consider a material line $\mathcal{M}(t_0)$ calculated for some time t_0, then a time-dependent material line, $\mathcal{M}(t)$ is attained by letting the flow-map act on it, $\mathcal{M}(t) = F_{t_0}^t(\mathcal{M}(t_0))$[6]. To measure the rate of normal repulsion or attraction, we look at the dot product of the flow-gradient, $\nabla F_{t_0}^t(\mathbf{x}_0)$, and the unit normal $\mathbf{n}_0$ to each point $\mathbf{x}_0 \in \mathcal{M}(t_0)$. Figure 3 illustrates how a material line is advected by the flow. The point $\mathbf{x}_0$ is mapped to $\mathbf{x}_t$ by the flow map $\mathbf{x}_t = F_{t_0}^t(\mathbf{x}_0)$. The unit vector orthogonal to the line $\mathcal{M}(t)$ in point $\mathbf{x}_t$ is denoted $\mathbf{n}_t$. To see how the rate of repulsion changes with $\mathcal{M}(t)$ it has to be projected onto $\mathbf{n}_t$. This projection is denoted $\nu_{t_0}^t$ for simplicity,

$$\nu_{t_0}^t = \langle \mathbf{n}_t, \nabla F_{t_0}^t(\mathbf{x}_0)\mathbf{n}_0 \rangle. \tag{11}$$

If $\nu_{t_0}^t > 1$ over the interval $[t_0, t]$ then $\mathcal{M}(t)$ has been overall repelling over the interval. If $\nu_{t_0}^t < 1$ over the interval, then $\mathcal{M}(t)$ has been overall attracting. $\nu_{t_0}^t$

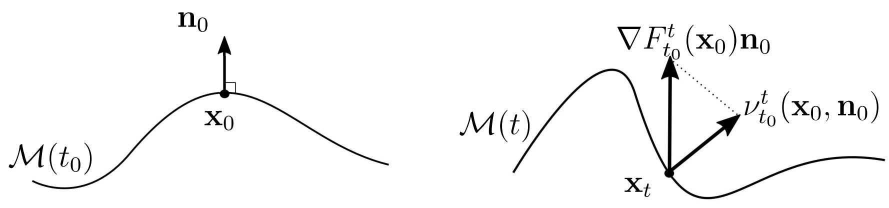

$\mathbf{n}_0$
$\mathbf{x}_0$
$\mathcal{M}(t_0)$
$\nabla F_{t_0}^t(\mathbf{x}_0)\mathbf{n}_0$
$\nu_{t_0}^t(\mathbf{x}_0, \mathbf{n}_0)$
$\mathcal{M}(t)$
$\mathbf{x}_t$

Figure 3: The material line is advected by the flow map. The rate of repulsion is the projection of $\nabla F_{t_0}^t(\mathbf{x}_0)\mathbf{n}_0$ onto $\mathbf{n}_t$. $\mathbf{n}_t$ is a unit vector normal to $\mathcal{M}(t)$ at $\mathbf{x}_t$. This rate of repulsion is denoted $\nu_{t_0}^t(\mathbf{x}_0, \mathbf{n}_0)$ for simplicity. If the rate of repulsion $\nu_{t_0}^t$ is larger than 1 averaged over the material line, then the material line is said to be a repelling material line. If $\nu_{t_0}^t < 1$ over the interval, the material line is said to be an attracting material line. The locally most repelling or attracting material lines are classified as LCSs.

can be computed using the CG strain tensor by using that

$$\begin{aligned} 0 =& \langle \mathbf{e}_0, \mathbf{n}_0 \rangle, \quad [\nabla F_{t_0}^t]^\dagger [\nabla F_t^{t_0}]^\dagger = [\nabla F_t^{t_0} \nabla F_{t_0}^t]^\dagger = 1 \\ 0 =& \langle \nabla F_{t_0}^t \mathbf{e}_0, [\nabla F_t^{t_0}]^\dagger \mathbf{n}_0 \rangle. \end{aligned} \tag{12}$$

The vector $\mathbf{e}_0$ is the unit tangent vector to $\mathcal{M}(t_0)$, and † denotes matrix transposition. Using the previous result coupled with the fact that $\nabla F_{t_0}^t \mathbf{e}_0$ is also tangent to $\mathcal{M}(t)$ we can express $\mathbf{n}_t$ as

$$\mathbf{n}_t = \frac{[\nabla F_t^{t_0}]^\dagger \mathbf{n}_0}{|[\nabla F_t^{t_0}]^\dagger \mathbf{n}_0|}. \tag{13}$$

Plugging this into the expression for $\nu_{t_0}^t$ leads to a simplified expression for $\nu_{t_0}^t$ using the CG strain tensor and $\mathbf{n}_0$. This is useful when computing the rate of repulsion as it does not depend on the geometry of $\mathcal{M}(t)$.

$$\begin{aligned} \nu_{t_0}^t &= \langle \mathbf{n}_t, \nabla F_{t_0}^t \mathbf{n}_0 \rangle = \frac{\langle \mathbf{n}_0, \nabla F_t^{t_0}(\mathbf{x}_t) \nabla F_{t_0}^t(\mathbf{x}_0) \mathbf{n}_0 \rangle}{\sqrt{\langle \mathbf{n}_0, [\nabla F_t^{t_0}][\nabla F_t^{t_0}]^\dagger \mathbf{n}_0 \rangle}} \\ &= \frac{1}{\sqrt{\langle \mathbf{n}_0, [C_{t_0}^t(\mathbf{x}_0)]^{-1} \mathbf{n}_0 \rangle}}. \end{aligned} \tag{14}$$

Using this result allows us to define a repelling LCS as the locally most repelling material line $\mathcal{M}(t)$ over the interval $[t_0, T]$ where the normal repulsion rate satisfies the conditions

$$\begin{aligned} &\nu_{t_0}^T(\mathbf{x}_0, \mathbf{n}_0) > 1, \\ &\nu_{t_0}^T(\mathbf{x}_0, \mathbf{n}_0) > |\nabla F_{t_0}^T(\mathbf{x}_0)\mathbf{e}_0|. \end{aligned} \tag{15}$$

These conditions must hold for all $\mathbf{x}_0 \in \mathcal{M}(t_0)$. The latter condition states that the normal repulsion must be greater than the tangential repulsion for it to be considered a normally repelling material line. A normally attracting material line is defined as a repelling line over the backward time interval $[T, t_0]$. Because LCSs are calculated over finite intervals, there is no guarantee that one LCS should persist over several intervals unless the new intervals are small perturbations to the original interval.

Farazmand and Haller proved that the following four conditions must be met in order to have a line be classified as an LCS[7],

$$\begin{aligned} &(A) \quad \lambda_1 \neq \lambda_2 > 1, \\ &(B) \quad \langle \boldsymbol{\zeta}_2(\mathbf{x}_0), \nabla^2 \lambda_2(\mathbf{x}_0) \boldsymbol{\zeta}_2(\mathbf{x}_0) \rangle \leq 0, \\ &(C) \quad \boldsymbol{\zeta}_1(\mathbf{x}_0) || \mathcal{M}(t_0), \\ &(D) \quad \bar{\lambda}_2, \text{ the average of } \lambda_2 \text{ over a curve } \gamma, \text{ is maximal on } \mathcal{M}(t_0) \text{ among} \\ &\qquad \text{all nearby curves } \gamma \text{ satisfying } \gamma || \boldsymbol{\zeta}_1(\mathbf{x}_0). \end{aligned} \tag{16}$$

The first condition is the same as in eq. (15) while conditions B, C and D ensure that the material line is the *locally most* repelling line. Farazmand and Haller developed a numerical approach to identifying LCSs as material lines satisfying conditions A-D, however, their algorithm for condition D is not fully described and depends to some degree of personal judgment. It is an alternative method for enforcing this last condition, D, that is the objective of this paper. Note that the ∇^2 in condition B denotes the Hessian, not the Laplacian.

2.4 Computing LCSs

As conditions A-D are defined in terms of the eigenvalues and eigenvectors of the CG strain tensor, these must be calculated. We begin by calculating the gradient of the flow map, $\nabla F_{t_0}^t(\mathbf{x}_0)$. The gradient of the flow map can not be calculated analytically so an approximation has to be made. The standard approach for this is to use a uniform grid of tracers and advect them from t_0 to t. Then the deformation can be calculated from the uniform grid to obtain an approximation for $\nabla F_{t_0}^t(\mathbf{x}_0)$[5]. The number of tracer particles can be increased to get a better estimate, but this could in turn introduce noise into the calculations. Features with small spatial size can become more dominating with a higher resolution[7]. However, to obtain an accurate estimate for the eigenvectors, a large ensemble of particles would have to be used. In order to increase the accuracy of the calculation of the flow-map gradient $\nabla F_{t_0}^t(x)$ without having to resort to an enormous grid, an auxiliary grid is added around the uniform grid[7]. This is illustrated in figure 4. The estimate obtained for $\nabla F_{t_0}^t$ can be greatly improved by using an auxiliary grid with spacing $\delta x << \Delta x$ and $\delta y << \Delta y$. Because of this small spacing the gradient of the flow map can be accurately approximated by the centered differences,

$$\nabla F_{t_0}^t(\mathbf{x}_i) \approx (\frac{F_{t_0}^t(\mathbf{x}_i^r) - F_{t_0}^t(\mathbf{x}_i^l)}{2\delta x} \frac{F_{t_0}^t(\mathbf{x}_i^u) - F_{t_0}^t(\mathbf{x}_i^d)}{2\delta y}), \tag{17}$$

where $\mathbf{x}_i = [x_i, y_i]$ denotes the position of particle i and the auxiliary grid is given by

$$\begin{aligned} \mathbf{x}_i^u &= (x_i, y_i + \delta y), \quad \mathbf{x}_i^d = (x_i, y_i - \delta y), \\ \mathbf{x}_i^l &= (x_i - \delta x, y_i), \quad \mathbf{x}_i^r = (x_i + \delta x, y_i). \end{aligned} \tag{18}$$

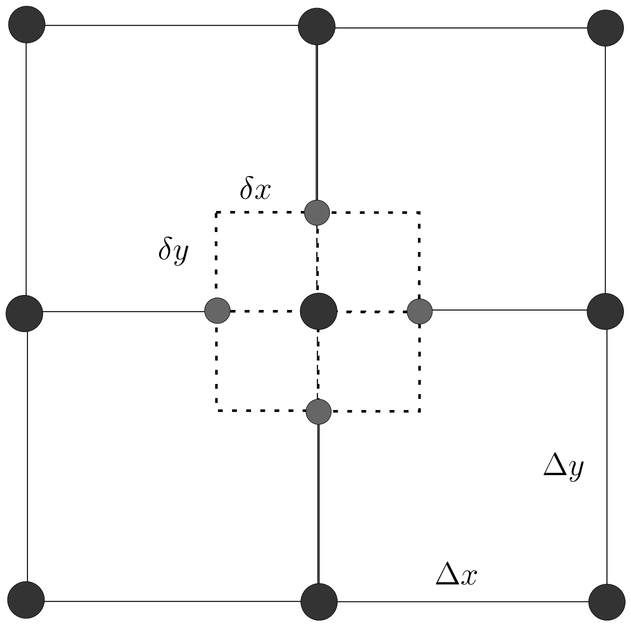


Figure 4: The uniform grid has spacing Δx and Δy. By introducing an auxiliary grid one achieves improved accuracy and a reduction in noise when computing the gradient of the flow-map. The auxiliary grid spacing is δx and δy.

The reason for using an auxiliary grid instead of simply using 5 times more grid points is that it gives a much better estimate for $\nabla F_{t_0}^{t}(\mathbf{x}_0)$ using nearly the same computational cost as using a larger uniform grid. However, because of the small distances between the grid points, the auxiliary-grid experiences less deformation than the main grid. This makes it unsuited for calculating the eigenvalues[7]. Therefore the uniform grid is used for calculating the eigenvalues λ_i and the auxiliary grid is used for calculating the eigenvectors $\boldsymbol{\zeta}_i$.

2.4.1 Calculating Strainlines

Following the notation of Farazmand and Haller (2012), the lines tangent to $\boldsymbol{\zeta}_1$ are referred to as strainlines. Computation of the strainlines is done by first determining the points that satisfy conditions A and B in eq. (16). This is justified because only strainlines conforming to these conditions can be LCSs. Then trajectories are devloped from these points by solving the ordinary differential equation

$$\mathbf{r}' = \boldsymbol{\zeta}_1(\mathbf{r}), \quad |\boldsymbol{\zeta}_1(\mathbf{r})| = 1. \tag{19}$$

Any trajectory of eq. (19) is by definition everywhere tangent to $\boldsymbol{\zeta}_1$, and thus these lines satisfy condition C. Here we computed $\boldsymbol{\zeta}_2$ and $\boldsymbol{\zeta}_1$ using the *numpy* eigensolver on the CG strain tensors for both grids, which returns the unit eigenvectors and the eigenvalues[15].

The eigenvectors $\boldsymbol{\zeta}_1$ and $\boldsymbol{\zeta}_2$ are calculated from the CG strain tensor. The orientation of these vectors is not uniquely defined. This results in possible orientational discontinuities along strainlines during the integration[7]. Discontinuities like these

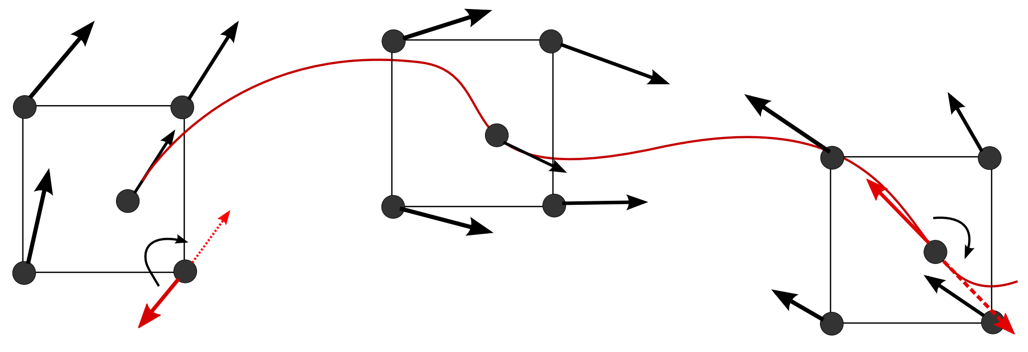


Figure 5: The field of eigenvectors $\boldsymbol{\zeta}$ may inhibit orientational discontinuities. It is therefore important to check the direction and possibly rotate the vector during the strainline integration. Discontinuities like these can seldom be removed globally and trying to remove them locally will simply shift the discontinuity elsewhere. In (a) one of the surrounding vectors has to be flipped 180°before $\boldsymbol{\zeta}(\mathbf{r})$ can be found through interpolation. For the plot in (b), all the vectors point in the same direction, so no extra action is needed. The last plot illustrates the case where all the local eigenvectors point in the same direction, but the resulting $\boldsymbol{\zeta}(\mathbf{r})$ points in the opposite direction of the earlier integration steps. In this case the resulting vector needs to be rotated 180°.

can not usually be removed globally and it is therefore necessary to check the orientation and correct for discontinuities locally as one goes along the integration. This can be done by selecting a reference among the nearby grid points, flipping the vectors if needed. When the orientation of the nearby points has been fixed, they can be used to interpolate the value for $\boldsymbol{\zeta}_1(\mathbf{r})$. This new value is then compared to the previous orientation in the integration, correcting it if necessary. The orientational discontinuity is illustrated in figure 5. As we can see from the figure, the orientation of the vector at the grid point located in the middle needs to be rotated 180 degrees. In addition to orientattional discontinuities, another issue with the integration which needs to be accounted for is the appearance of fixed points. At certain points $\lambda_1 = \lambda_2$, which makes the eigenvectors $\boldsymbol{\zeta}_i$ undefined. This is solved by introducing the scaling factor suggested by Tchon et al [16],

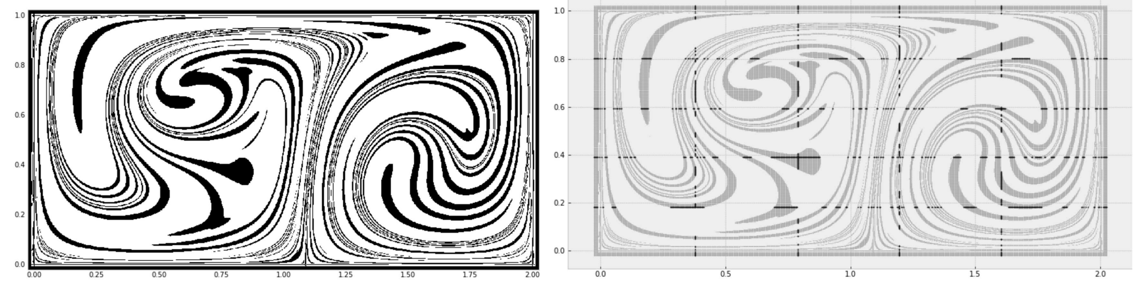
$$\alpha(\mathbf{x}) = (\frac{\lambda_1(\mathbf{x}) - \lambda_2(\mathbf{x})}{\lambda_1(\mathbf{x}) + \lambda_2(\mathbf{x})})^2. \tag{20}$$

Equation (19) is then altered to give a globally smooth set of strainlines

$$\mathbf{r}'(s) = sign\langle \boldsymbol{\zeta}_1, \mathbf{r}'(s-\Delta)\rangle \alpha(\mathbf{r}(s))\boldsymbol{\zeta}_1(\mathbf{r}(s)), \tag{21}$$

where $sign\langle \boldsymbol{\zeta}_1, \mathbf{r}'(s-\Delta)\rangle$ returns the sign of the inner product of $\boldsymbol{\zeta}_1$ and $\mathbf{r}'(s-\Delta)$. This ensures that $\mathbf{r}'(s)$ has the same orientation as at the previous step, $s-\Delta$.

The integration of eq.(21) is computationally intensive which means that one should avoid unnecessary work if possible. In figure 6 (a) we can see the field satisfying condition A and B for an example flow system (see section 3.1). The



(a) **(b)**

Figure 6: (a) The set of points satisfying conditions A and B given by eq.(16). To reduce the number of redundant computations a grid is used to extract a set of points. This effectively reduces the number of points to integrate over by two magnitudes, as shown in (b).

figure was made advecting a grid $\mathcal{G}$ of 1000×500 points, calculating the CG strain tensor, and checking if the λ_2's and $\boldsymbol{\zeta}_2$'s satisfied the two first conditions in eq. (16). The number of points $\mathcal{G}_0 \in \mathcal{G}$ that satisfied these conditions was more than 10^5. It can be expected that a number of these points belong to the same strainline which means that running the integration over all points is going to result in a lot of redundant work. By introducing a number of horizontal and vertical lines, and extracting the points that coincide with these lines, the number of points can be reduced by several orders of magnitude[7]. This is illustrated in figure 6 (b). In this plot only about $2 \cdot 10^3$ points coincide with the lines, reduced from the much larger set of 10^5 points satisfying conditions A and B. From the reduced set of initial conditions, strainlines were developed as trajectories of eq. (21). From each initial condition, a trajectory was started both in the direction of $\boldsymbol{\zeta}_1$ and $-\boldsymbol{\zeta}_1$, whereafter the orientation was corrected dynamically as described. The trajectories were stopped if they left the domain U, or if conditions A and B failed repeatedly over a length l_f [7].

2.4.2 LCS Extraction

Condition D in eq. (16) states that the average λ_2 over a curve γ has to be maximal among all nearby curves $\boldsymbol{\zeta}_1 || \gamma$ for it to be classified as an LCS. This introduces the concepts of locality and likeness in order to extract the most dominating strainlines. Farazmand and Haller suggests adding a set of horizontal and vertical lines, L, forming a coarse grid[7]. The coarseness depends on the size of the eddies present in the flow. The strainlines crossing a line in L within some distance ϵ to each other are compared, and the one with the maximum average eigenvalue $\bar{\lambda}_2$ is classified as an LCS. This is illustrated in figure 7. The figure plots three curves, γ_i. The curves are compared to their neighbors at each intersection with the gridlines, L. A red ellipse represents the lines which are within a distance, ϵ, of each other. The dashed ellipses illustrate that the lines are too far away from each other to be compared.

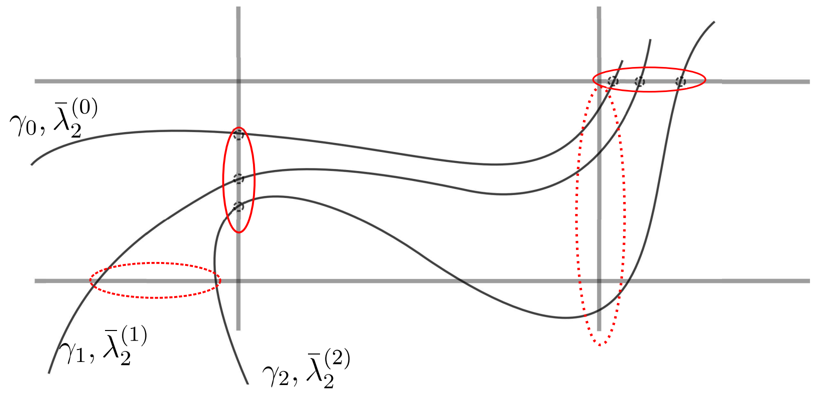

$\gamma_0, \bar{\lambda}_2^{(0)}$
$\gamma_1, \bar{\lambda}_2^{(1)}$
$\gamma_2, \bar{\lambda}_2^{(2)}$

Figure 7: The approach for comparing strainlines as suggested by Farazmand and Haller (2012). Strainlines, denoted γ, which intersects the horizontal or vertical lines are compared if they are within some distance, ϵ, of each other. The strainline which has the largest mean eigenvalue, $\bar{\lambda}_2$ is classified as a LCS. The red ellipses illustrate curves which are compared, while the curves enclosed by the dashed ellipses are too far away from each other.

The number of horizontal and vertical grid-lines are not specified in Farazmand and Haller (2012), and neither is the location of these lines. Both of these parameters might affect which strainlines are identified as LCSs. Looking at figure 7, if the vertical line to the left of the figure was shifted a couple of centimeters more to the left, the curve, γ_2 would not be intersecting with it anymore, resulting in γ_2 not being compared with the other curves at that location. Other tools could be used in order to obtain some intuition on how to create the grid of lines. An example is the FTLE field. The ridges of the FTLE field has previously been linked to LCSs, and can therefore be used to get an indication of where to expect LCSs to appear[14]. However, caution is adviced as the FTLE field is known to exhibit both false positives and false negatives when used to identify LCSs[5]. Ultimately, this method of using a grid of horizontal and vertical lines introduces some degree of personal preference into the process of identifying LCSs.

This project aims at finding a more robust way of identifying the LCSs present in flow fields using a form of unsupervised machine-learning called clustering[9]. Clustering finds correlation in data and groups data objects, or patterns, into clusters. The idea is that it should be possible to group similar strainlines, and then select the most dominating strainline in each cluster as the LCSs of the system. The use of clustering algorithms could make the process of extracting the LCSs more objective, i.e. removing the need to inspect the field or tuning the grid lines.

2.5 LCS Advection

By following the procedures explained in section 2.3 and 2.4.2 the LCSs at time t_0 can be identified. To obtain the LCSs at a later time t the LCSs can be advected by the flow map[7].

$$\mathcal{M}(t) = F_{t_0}^{t}(\mathcal{M}(t_0)). \tag{22}$$

$\mathcal{M}(t_0)$ is here a LCS. Note that there is no guarantee that the material line $\mathcal{M}(t) = F_{t_0}^{t}(\mathcal{M}(t_0))$ is still considered a LCS of the system. The LCS was calculated over some interval $[t_0, T]$. At a later time $t > t_0$ this line might not satisfy all of the conditions in eq. (16) anymore. Another issue is that the advection of a locally most repelling line is intrinsically unstable. This instability is caused by the tendency of small displacements to grow exponentially[13]. This can result in a small initial error which grows exponentially with time. For attracting material lines the distance decreases proportionally to an inverse exponential function. The attracting lines are therefore much more stable than their repelling counterparts.

2.6 Dimensional Reduction

The material line integration gives a set of N lines with varying number of points l_n. Before any clustering can be performed, all lines have to be described by vectors of equal dimensions. This is because in clustering, the dimension of the vector is considered the features describing the data. This number needs to be high enough so that the longest lines in the set can be represented with good resolution. For this project, the number of points used was set to $M = 2000$. This gave sufficient resolution for all of the lines. The lines were resampled to the desired number of points using cubic spline interpolation. This lead to each line being described by two thousand features. When clustering the data, each line, or pattern, is represented as an M dimensional point in feature-space[8]. These are the features that the lines are grouped by. The clusters are commonly formed by using some metric to measure the similarity of the features, for example the Euclidean distance in feature-space. The problem with high dimensional data is that the clustering is more computationally intensive and that the dimensionality may degrade the reliability of the results. This is known as the curse of dimensionality[17]. The term was coined by R. Bellman in 1957. This is because for a fixed number of samples, as the number of features increases, so too does the number of parameters that have to be estimated. These estimates will then be less reliable as there is less data to use relative to the number of parameters[8].

Feature selection and **Feature extraction** are two methods of doing dimensional reduction. The former tries to pick the features which best describes the data, while the latter seeks to reduce the number of features through linear- and nonlinear-transformations. Because the data being analyzed are lines where each feature is a point in two dimensions, it is reasonable to assume that we will not be able to use feature selection to pick a small number of points which describe each line. While it is possible to use a coarser subset of points and still have a line with some resemblance to the original, some information is lost due to the poor resolution. Feature extraction on the other hand, aims at retaining as much of the information stored in the data as possible, resulting in a more compact representation[18]. The main focus here will therefore be feature extraction. The two most used techniques for feature extraction are Principal Component Analysis (PCA) and Kernel Principal Component Analysis (K-PCA)[19]. In this paper a more recent method called t-Distributed Stochastic Neighbor Embedding (t-SNE) has also been tested along with PCA and K-PCA[11].

2.6.1 Principal Component Analysis

A common method of feature extraction is the linear transformation, **Principal Component Analysis** (PCA)[18]. The goal of PCA is to reduce the high dimensional data to a lower dimensionality where each new feature is described by a linear combination of the original features. The PCA is optimalized such that the new representation retains as much information from the high dimensional representation as possible, resulting in the new representation being a more compact form of the original data. For N lines each consisting of M points, the dimension of the feature space is equal to the number of points, $d = M$. A line is therefore given on the form

$$\mathbf{x}_i = \begin{bmatrix} x_i^1 \\ x_i^2 \\ \vdots \\ x_i^M \end{bmatrix}. \tag{23}$$

The complete set of lines may be represented by a $N \times M$ matrix,

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ x_1^2 & x_2^2 & \ddots & x_N^2 \\ \vdots & \ddots & \ddots & \vdots \\ x_1^M & \cdots & \cdots & x_N^M \end{bmatrix}. \tag{24}$$

The task at hand is therefore to find the linear map $\mathbf{W}$ which finds the optimal linear combination of features so that

$$\begin{aligned} \mathbf{W}^T\mathbf{x}_i &= \mathbf{y}_i, \\ \mathbf{y}_i &= \begin{bmatrix} y_i^1 & y_i^2 & \cdots & y_i^m \end{bmatrix}^T, \\ \mathbf{W} &= \begin{bmatrix} \mathbf{w}^1 & \mathbf{w}^2 & \cdots & \mathbf{w}^m \end{bmatrix}, \end{aligned} \tag{25}$$

where $m < M$ is the new dimension of the data. There are multiple algorithms for computing the PCA. In this report **Singular Value Decomposition** (SVD) was used.

The SVD is an efficient method for computing PCA. The algorithm consists of relatively few steps. The first is to gather the data in a matrix on the same form as in eq. (24). Then SVD is applied to the matrix $\mathbf{X}$, which computes a decomposition

$$\mathbf{X} = \begin{cases} \mathbf{V}[\mathbf{D} \quad \mathbf{0}]\mathbf{U}, & \text{if } M \leq N \\ \mathbf{V}\begin{bmatrix} \mathbf{D} \\ \mathbf{0} \end{bmatrix}\mathbf{U}, & \text{if } M > N. \end{cases} \tag{26}$$

Where $\mathbf{V}$ and $\mathbf{U}$ are $M \times M$ and $N \times N$ unitary matrices respectively. The matrix $\mathbf{D}$ is a diagonal matrix consisting of the non-singular values of $\mathbf{X}$ and is of dimension $N \times N$ or $M \times M$ depending on which one is smallest. The notation $[\mathbf{D} \quad \mathbf{0}]$ and $\begin{bmatrix} \mathbf{D} \\ \mathbf{0} \end{bmatrix}$ represents $M \times N$ and $N \times M$ matrices. The two matrices are comprised of the non-singular values of $\mathbf{X}$ along the diagonal, with an additional $|N - M|$ columns or rows of zeros. The reduced representation $\mathbf{y}$ is then formed by the first m principal components of a test vector $\mathbf{x}$,

$$\mathbf{y} = \mathbf{V}_m^T \mathbf{x}. \tag{27}$$

$\mathbf{V}_m$ is the $m \times N$ matrix formed by the first m rows of V.

Two examples of optimality criteria for PCA is the **Mean-square-error criterion** and the **Maximum-entropy criterion**. The first is based on the reconstruction error. This is the difference between the $\mathbf{x}$ obtained from using the reduced representation $\mathbf{y}$ and the original data $\mathbf{X}_i$. The latter criteria is based on how much relevant information is retained by the reduced representation, $\mathbf{y}$[18]. The error obtained by either of these optimality criteria is then used to update the values in $\mathbf{V}$, and then the process of calculating $\mathbf{y}$ and reconstructing $\mathbf{x}$ is repeated until an optimum has been reached.

2.6.2 Kernel-PCA

PCA can be viewed as a special case of kernel-PCA (K-PCA) where K-PCA is applied using a linear kernel. The point of using K-PCA instead of the regular PCA is that linear combinations of features will not always yield good results. The different classes of data can not always be linearly separated. This is caused by the nonlinear relation between the features[18]. By using a kernel to map the data to some Hilbert vector space, $\mathcal{H}$, where

$$\mathbf{x} \to \boldsymbol{\phi}(\mathbf{x}), \tag{28}$$

which is of a dimension $d' \geq d$, the points can generally be separated. The function $\boldsymbol{\phi} \in \mathcal{H}$ is defined by

$$\mathbf{K}(x, y) = \boldsymbol{\phi}^T(\mathbf{x})\boldsymbol{\phi}(\mathbf{y}), \tag{29}$$

where $\mathbf{K}(\mathbf{x}, \mathbf{y})$ is the kernel. The kernel must obey the following conditions in order to be a distance metric,

$$\begin{aligned} &|\mathbf{x} - \mathbf{y}| \geq 0 \\ &|\mathbf{x} - \mathbf{y}| = |\mathbf{y} - \mathbf{x}| \\ &|\mathbf{x} - \mathbf{y}| \leq |\mathbf{x} - \mathbf{z}| + |\mathbf{z} - \mathbf{y}|. \end{aligned} \tag{30}$$

A traditional choice for measuring the distance between two points is the Euclidean distance which is given by

$$||\mathbf{x} - \mathbf{y}|| = \sqrt{\mathbf{x} \cdot \mathbf{x} + \mathbf{y} \cdot \mathbf{y} - 2\mathbf{x} \cdot \mathbf{y}}. \tag{31}$$

The only kernel functions which satisfy the distance axioms from eq.(30) are Mercer kernels. These are kernel functions which satisfy the Mercer condition[18]

$$\int K(\mathbf{x}, \mathbf{y}) h(\mathbf{x}) h(\mathbf{y}) \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{y} \geq 0, \tag{32}$$

for any squarely integrable function $h(\mathbf{x})$. If the condition is met, a mapping exists

$$\boldsymbol{\phi} : \mathbf{x} \to \boldsymbol{\phi}(\mathbf{x}) \in \mathcal{H}, \tag{33}$$

where the inner product of $\mathcal{H}$ is represented by $K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}^T(\mathbf{x})\boldsymbol{\phi}(\mathbf{y})$. This allows us to define the distance metric for the Mercer kernels as

$$\sqrt{K(\mathbf{x}, \mathbf{x}) + K(\mathbf{y}, \mathbf{y}) - 2K(\mathbf{x}, \mathbf{y})}. \tag{34}$$

A few examples of Mercer kernels are the polynomial kernel function, Radial Basis Function (RBF) and the cosine kernel function. These are given by the following expressions,

$$\begin{aligned} K_{pol}(\mathbf{x}, \mathbf{y}) &= \left(C + \frac{\mathbf{x} \cdot \mathbf{y}}{\sigma^2}\right)^p \\ K_{rbf}(\mathbf{x}, \mathbf{y}) &= \exp\left(-\gamma ||\mathbf{x} - \mathbf{y}||^2\right) \\ K_{cos}(\mathbf{x}, \mathbf{y}) &= \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||\mathbf{y}||}. \end{aligned} \tag{35}$$

For $\gamma = 1/\sigma^2$, where σ^2 is the variance, the RBF becomes the Gaussian kernel. The C in the expression for the polynomial kernel is an arbitrary constant. The dimensionality of this new vector space is determined by the components of the dot products of the input vectors. The kernels are expressed as a series-expansion which means that if the RBF kernel is used, the new space will have an infinite dimension. This mapping can be illustrated using the polynomial kernel in eq. (35). If we have a two dimensional input vector $\mathbf{x} = [x_1, x_2]$ and a second order polynomial kernel where $C = 0$, then the new space is described by [19]

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2 &= \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y}) \\ &= [x_1^2, x_1x_2, x_2x_1, x_2^2]^T[y_1^2, y_1y_2, y_2y_1, y_2^2], \end{aligned} \tag{36}$$

where $\boldsymbol{\phi}(\mathbf{x})$ is given by

$$\boldsymbol{\phi}(\mathbf{x}) = [x_1^2, x_1x_2, x_2x_1, x_2^2]. \tag{37}$$

This new vector $\boldsymbol{\phi}(\mathbf{x})$ is of dimensionality four, which is larger than the dimensionality of the original vector. For higher order kernels, like the RBF or a high order polynomial, the new space can be of a very large dimensionality, sometimes even infinite. This makes it cumbersome to work with and would require immense computational power. Fortunately it is possible to only use a subspace of this new Hilbert space for the Kernel-PCA.

The kernel-mapping results in a new set of data, $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \cdots, \boldsymbol{\phi}(\mathbf{x}_N)]$. This makes it possible to define an empirical representation based on the number of patterns in the data, N,

$$\mathbf{k}(\mathbf{x}) = \mathbf{\Phi}^T \boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}) \\ K(\mathbf{x}_2, \mathbf{x}) \\ \vdots \\ K(\mathbf{x}_N, \mathbf{x}) \end{bmatrix}. \tag{38}$$

This empirical representation is of dimension N. If N is lower than the dimensionality of the Hilbert space, then $\boldsymbol{\phi}(\mathbf{x})$ can be substituted for the empirical representation $\mathbf{k}(\mathbf{x})$. This is called the kernel-trick[19].

2.6.3 t-Distributed Stochastic Neighbor Embedding

The Stochastic Neighbor Embedding (SNE) is a technique for visualizing high-dimensional data in two to three dimensions[20]. It starts with transforming the high dimensional Euclidean distance into probabilities. These probabilities represent how likely it is that a data point i would choose another point, j, as its neighbor. t-Distributed Stochastic Neighbor Embedding (t-SNE) is a variation of the SNE and remedies some of its shortcomings. One of the faults that t-SNE corrects is the tendency of the SNE to crowd points together[11]

SNE converts the distance between data points into conditional probabilities. For two points $\mathbf{x}_i$ and $\mathbf{x}_j$, we have the probability $p_{j|i}$ of selecting point $\mathbf{x}_j$ as $\mathbf{x}_i$'s nearest neighbor. The probability $p_{j|i}$ is given by

$$p_{j|i} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2\right)}{\sum_{k \neq i} \exp\left(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma^2\right)}, \tag{39}$$

where σ is variance of the Gaussian and may vary over the domain and is linked with the perplexity of the data. It is therefore necessary to find the optimal value of σ which gives the wanted perplexity. The perplexity, Λ is defined as

$$\Lambda(P_i) = 2^{H(P_i)}, \tag{40}$$

where $H(P_i)$ is the Shannon entropy and P_i is the probability distribution corresponding to a Gaussian distribution with variance σ_i. The Shannon entropy is given by

$$H = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{41}$$

and is measured in bits. Λ can be linked with the number of neighbors to use in the calculation and is closely related to the topology of the resulting clusters. A low Λ results in the algorithm largely focusing on the local structure. A parallel can be made to the problem of overfitting in machine-learning[21]. When the t-SNE only considers the local structure, it ignores the more general global structure. This will in turn result in a large number of small clusters.

In addition to the high-dimensional mapping $p_{j|i}$, a low-dimensional mapping q_{ji} is introduced, using the points $\mathbf{y}_i, \mathbf{y}_j$ which is defined on some space with dimension $d' < d$.

$$q_{ji} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathbf{y}_i - \mathbf{y}_k||^2)^{-1}}. \tag{42}$$

The high-dimensional probability space follows a Gaussian distribution while the low-dimensional space, q_{ji} follows a Student t-distribution with a single degree of freedom. It is the distribution in the low-dimensional space which gives the t-SNE its name, and it is also where the t-SNE diverges from the SNE[11]. The broader tail of the Student t-distribution ensures that moderate distances are transformed into larger distances on the map. This ensures that objects that are moderately different are placed far enough from each other. In order to have a measure of how well Q represents P we use the Kullback-Leibler divergence as a cost-function, which is defined as

$$C = \sum_i KL(P||Q) = \sum_i \sum_j p_{ji} \log \frac{p_{ji}}{q_{ji}}, \tag{43}$$

where P and Q are the joint probability distributions over all points in the high- and low-dimensional spaces correspondingly. Because the measures p_{ij} and q_{ij} measure the pairwise distances between points in their respective spaces, we can set $p_{ii} = q_{ii} = 0$. We then set $p_{ji} = \frac{p_{j|i} + p_{i|j}}{2n}$ which removes the problem of outlier terms not contributing to the cost function. It also ensures that both p_{ij} and q_{ij} are symmetric. Using the above definitions of q, p and C, the gradient of the cost function is then given by

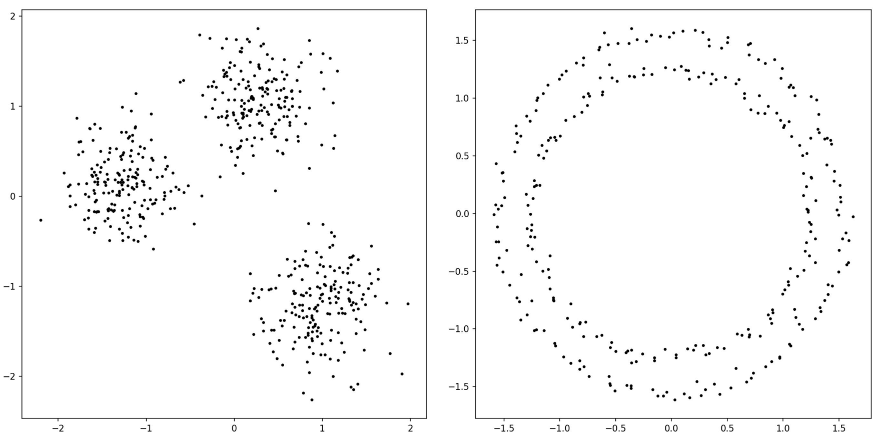
$$\frac{\delta C}{\delta \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}. \tag{44}$$

The cost function is then minimized using gradient descent until the global minimum is found and a good mapping has been achieved.

2.7 Clustering

Clustering is a method utilizing unsupervised machine learning to group similar objects into clusters. It is an unsupervised method because the data is not assigned labels[8]. The problem is illustrated in figure 8. We have no information what the dots represents, but in figure 8 (a) they are clearly split into three distinct groups in this two dimensional feature space.

In clustering, data is represented as a point in some d dimensional feature space. An example could be categorizing flowers, where color and petal size could be two features of the data. Such a point in feature space is often referred to as a pattern. Another good example of unsupervised clustering is the categorization of stars. By gathering information on the spectra and luminosity of more than a hundred thousand stars, scientists were able to perform unsupervised clustering and categorize them into types such as "main sequence stars", "red giants" and



(a) (b)

Figure 8: The data consists of unlabeled points spread in $\mathbb{R}^2$. While there is no available information of what these points represent, it is apparent that in the first figure there are three distinct groups. In the second figure we can see that there are two rings. Clustering aims at uncovering these structures and grouping similar data.

"white dwarfs"[9]. The spectra and luminosity in the example are two features describing a star. For cases dealing with clustering in two dimensions, humans are able to do it without the help of machines. However, for problems in dimensions higher than three, it will be an arduous task to project the data onto various axes and try to identify clusters. Furthermore, in a lot of cases, the patterns are described by more than two features[10].

A typical approach to clustering data involves five steps, of which the three first are discussed in detail here, as the two last are less connected to the actual clustering[8]. The first step is **pattern representation**, which involves preparing the data. Usually the data consists of patterns represented as vectors or points in a multidimensional space. This step can involve selecting which features to use for the clustering and scaling them. For some algorithms the number of patterns to uncover has to be chosen prior to clustering. The different parameters that have to be adjusted depends on the chosen algorithm. Such a parameter could for example be the number of neighboring points needed to form a cluster. Some clustering algorithms are designed to identify the number of different groups to cluster each pattern into by analyzing the data. An appropriate pattern representation can make the clustering process easy and understandable. For example, the patterns in figure 8 (b) would be difficult to separate in Cartesian coordinates, but switching to a polar representation could make it more manageable.

The second step is to define a **measure of similarity** between patterns. A common measure for this is the Euclidean distance in feature-space. There are many ways to define similarity and the choice of method depends on the nature of the data to cluster. Some other metrics are the Cosine distance, Manhattan distance and the Minkowski distance. The latter is known as the generalized distance metric and is defined as

$$D = (\Sigma_{k=1}^{M} |X_{ik} - X_{jk}|^{p})^{1/p}. \tag{45}$$

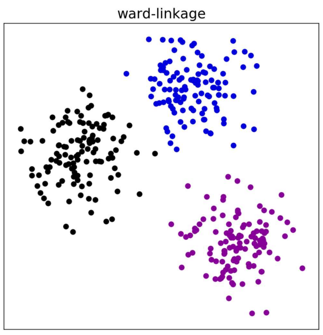
D is the distance between the two patterns, X_i, X_j, summed over each feature k. For $p = 2$ this equation is reduced to the Euclidean distance. The cosine distance uses the angle between the vectors and the radius. For problems where the points form a ring around the origin in a two dimensional feature space, a cosine distance could be better than the Euclidiean distance. These distance metrics do not take the ordering or location of the points along each strainline into account. In section 2.8 the Fréchet distance is presented. This metric can be used to measure the similarity between lines in $\mathbb{R}^2$[22]. The remaining steps are clustering the data, data abstraction, and evaluating the output. Data abstraction and evaluation can consist of mapping the data to another representation which can be more easily interpreted. Consider the set of strainlines after they have been reduced to some lower dimensionality by K-PCA or t-SNE. When clustering this data, it might be useful to study both the clusters formed in the low dimensional representation, and how this is mapped back to the high dimensional representation.

The clustering can begin when the two first tasks are complete. There are several algorithms for performing clustering of data. The two main groups of methods are hierarchical and partitioning methods[8]. Han and Kamber (2001) also proposed three more subgroups. These three are density-based-, model-based- and grid-based-clustering[23]. In this report Agglomerative-clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Affinity propagation were tested. The two first methods are hierarchical- and density-based-methods respectively and the latter is a partitioning method.

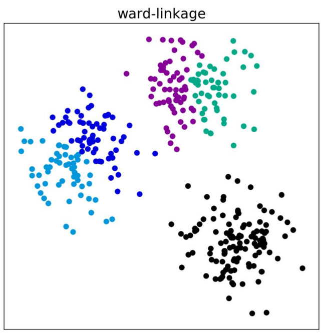
The *sci-kit learn* library was used for clustering data in python[24]. This library includes implementations of a wide range of machine-learning algorithms and methods for dimensional reduction.

2.7.1 Agglomerative-Clustering

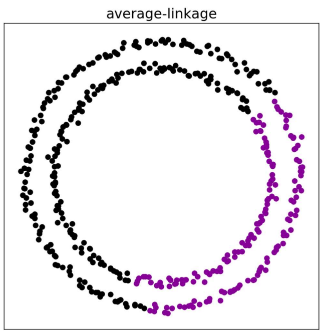
Agglomerative-clustering is a hierarchical method which begins with each pattern forming its own cluster. Then each cluster is merged with the closest neighbour. This process is repeated until some stopping criteria is reached. For the implementation in *sci-kit learn* this criterion is a chosen number of clusters. Most hierarchical clustering algorithms are either **Single**-link or **Complete**-link algorithms. The Single-link approach measures the similarity between two clusters as the minimum of the pairwise distances between the patterns in each cluster. This algorithm results in the creation of loose clusters, which have a tendency to bridge into other clusters. The Complete-linkage approach uses the maximum pairwise distance to measure similarity between two clusters. This tends to produce tighter clusters than for the single-link algorithms[23]. **Ward**-linkage minimizes the sum of squared differences within each cluster. This results in clusters with smaller variance in the data. The last linkage available from the *sci-kit learn* library is the **Average**-linkage. The Average-linkage minimizes the average pairwise distance between clusters[24].



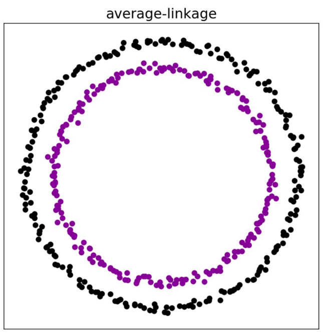
(a)



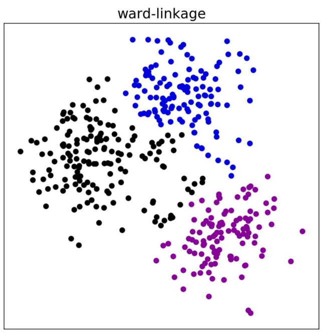
(b)



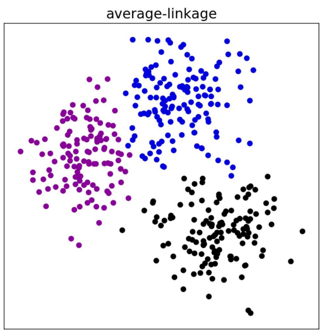
(c)



(d)



(e)



(f)

Figure 9: The Agglomerative-clustering algorithm requires the number of clusters to be known prior to clustering. This is illustrated in (a) and (b), where the number of clusters have been set to 3 and 5 respectively. The algorithm finds the wanted number of clusters whether they are present or not. The plots in (c) and (d) shows how the algorithm deals with more complex structures. In (d) a nearest neighbour graph is used, thus ensuring that the cluster does not jump over to the other circle. The last two figures show the difference between using the Average- and Ward-linkage approaches.

In figure 9 the performance of the Agglomerative algorithm is shown for 3 different clustering problems. This algorithm expects that the number of clusters are known prior to clustering. In figure 9 (a) and (b) the number of clusters were set to 3 and 5 respectively. It is clear that there are only 3 clusters in the data, but the algorithm groups the patterns into 5 clusters for the second case anyway. In the remaining subfigures the number of clusters is set to 2. Figure 9 (c) and (d) show how the algorithm performs for more complex geometries. A nearest neighbour graph has to be used in order to group the patterns correctly. The nearest neighbour graph limits the number of patterns connected to each other. For the case with the two circles, this effectively forces the algorithm to prioritize connecting the points on each respective ring. Figure 9 (d) and (c) the Agglomerative-clustering was performed with and without a nearest neighbor graph. Figure 9 (e) and (f) shows the difference between two of the linkage methods available in the *sci-kit learn* library.

2.7.2 DBSCAN

The DBSCAN algorithm identifies clusters as areas of higher density surrounded by areas of lower density[25]. This allows the clusters to take on any shape. To run this clustering algorithm, m, the minimum number of points needed for an ensemble to be considered a cluster, must be chosen. Additionally, the limit, r, for what is defined as dense enough must also be set[24]. The algorithm then searches through the patterns for a sample where there are m patterns within a distance r. This is then done iteratively until all patterns that belong to the core is found. When the cluster cores have been identified, the rest of the patterns can be assigned to the nearest cluster. Note that all of the patterns are not necessarily assigned to a cluster.

The DBSCAN algorithm do not require any prior knowledge about the number of clusters present in the data. In figure 10 (a) the method found the correct number of clusters and identified them. There are some outliers that were not assigned to any clusters. For figure 10 (b) and (c), the maximum distance between neighbours were set to 0.15 and 0.2 respectively. This shows how important it is to set a good value for ϵ. The final plot shows that the algorithm can handle other geometries as well.

2.7.3 Affinity Propagation

The third clustering algorithm tested in this paper is Affinity propagation. The method takes an initial measure of pairwise similarities as inputs and then messages are sent back and forth between the points. These messages are real numbers representing the affinity between points. This algorithm views each point as a node in a network, where some nodes are exemplars, representing a group of other points. Affinity propagation can uncover the number of clusters in the data with minimal input from the user. The algorithm uses a set of preferences to decide which patterns are suitable to be chosen as exemplars[26]. An exemplar is a point which is the best representative of a group. In the *sci-kit learn* library, if these are not specified by the user, the median of the input similarities is used. The implementation in *sci-kit learn* can also take a damping factor as input. This ensures that the algorithm do not exhibit oscillating solutions.

There are two types of messages being sent; The responsibility, and the availability between the points. The responsibility quantifies how suited an exemplar k is for representing the point i. The responsibility is given by the following formula

$$r(i,k) \leftarrow s(i,k) - \max_{k' s.t. k' \neq k} \{a(i,k') + s(i,k')\}, \tag{46}$$

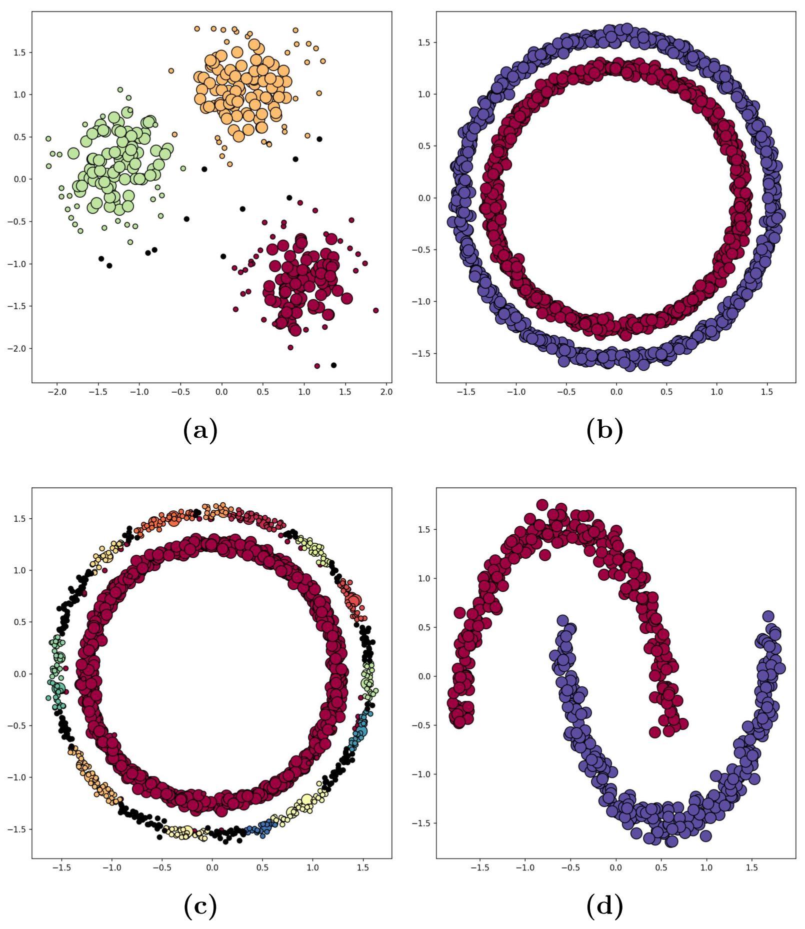


Figure 10: DBSCAN algorithm used on a variation of clustering problems. This algorithm is designed to identify the appropriate number of clusters. It is a density based model which means that it connects areas of similar density. Figure (a), (b) and (d) illustrates how it adapts to more complex geometries. In figure (c) the parameter governing the maximum distance between neighbours and the number of samples that were needed in each cluster was set too high.

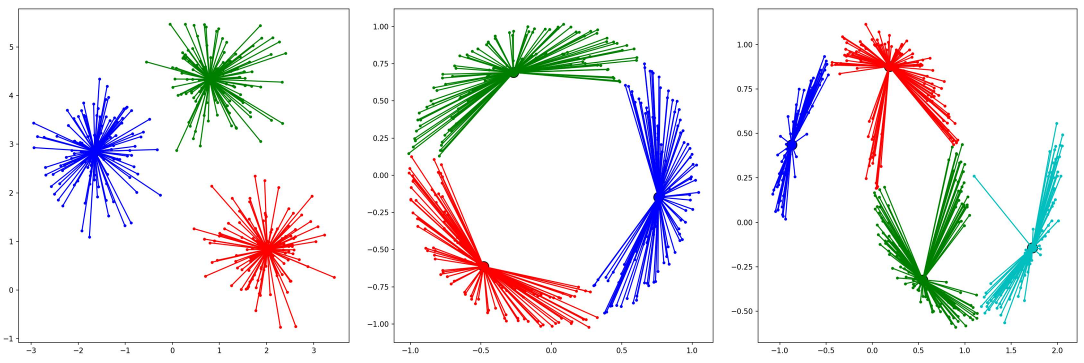
where $r(i,k)$ is the responsibility, $a(i,k)$ is the availability, and $s(i,k)$ is the similarity of the two points. The availability is sent from the exemplar k to the point i and represents the evidence for point i to choose k as its exemplar. The availability is updated as

$$a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' s.t. i' \notin \{i,k\}} \max\{0, r(i',k)\}\right). \tag{47}$$

The availability is upper bound so that only numbers less than or equal to 0 is allowed. $r(k,k)$ is the self-responsibility and represents the accumulated evidence that point k is an exemplar based on its input preference. The self-availability represents the evidence of point k being an exemplar based on the evidence gathered from the other points. The self-availability, $a(k,k))$ is updated differently than $a(i,k), i \neq k$.

$$a(k,k) \leftarrow \sum_{i' s.t. i' \neq k} \max\{0, r(i',k)\}. \tag{48}$$

The messages require only simple calculations and the algorithm has proved to be efficient at a range of clustering problems. Figure 11 shows how the clustering



(a) (b) (c)

Figure 11: Affinity propagation is designed to uncover the number of clusters. It works by passing messages between pairs containing the similarity to between them. Figure (a) shows the algorithm correctly identify the clusters and assign the points into its respective cluster without much problem. For more complex structures the algorithm seems to fall short.

algorithm performs faced with different geometries. Because of the centroid nature of the algorithm it gives poor results when the data can not be represented as blobs. Only the clusters in figure 11 (a) are correctly identified. In (b) and (c) it is apparent that this approach wont give the desired results. This can however be mended by preprocessing the data, either by changing the coordinate system or performing some transformations to it, like PCA.

2.8 Measure of Similarity

The patterns being analyzed are lines in a two dimensional space. It therefore seems reasonable to simply parameterize the lines so that they consist of the same number of points and take the sum of the pairwise Euclidean distance between two lines as a measure of their similarity. This will give a fair result, but the lines can have varying length, arbitrary endpoints and orientation, which might make some of the distances somewhat arbitrary. A program could be made to check if lines have overlapping segments, and then to calculate the distance between these. The problem with this approach is that the lines may be similar over a small interval, but can quickly diverge at a later point. There is also the added complexity of the code and the fact that it will require longer to run. Instead we use a metric known as the Frechét distance[22]. An analogy for this distance is the shortest leash a person can have, walking his or her dog when the person is bound to one path, and the dog to another. Both the person and the dog may vary their speed, but cannot go backwards. This method takes into account the location and ordering of the respective points[27].

The mathematical definition of the Fréchet distance, δ_F, is given by

$$\delta_F(f,g) := \inf_{\substack{\alpha[0,1]\to[a,a'] \\ \beta[0,1]\to[b,b']}} \max_{t\in[0,1]} ||f(\alpha(t)) - g(\beta(t))||. \tag{49}$$

f and g are curves in $\mathbb{R}^2$ each defined on some interval $[a,a']$ and $[b,b']$ respectively. These coordinates are then mapped to the strictly increasing functions $\alpha, \beta \in [0,1]$. $\alpha(0) = a$, $\alpha(1) = a'$ $\beta(0) = b$, $\beta(1) = b'$. The Fréchet distance, δ_F, is symmetric and satisfies the necessary conditions as a distance metric. If $\delta_F(f,g) = 0$ then the two lines f, g are equal[27].

2.8.1 The Rossby Radius

To obtain a general definition for locality, the Rossby radius of deformation (RRD) is used. It is closely linked with the scale of eddies in ocean circulation. More precicely, the first baroclinic Rossby radius of deformation measures the length scale of where vortex stretching is more important than vorticity[28].The RRD, R_r, is given by

$$R_r = \frac{c_1}{|f(\theta)|}, \tag{50}$$

where $f(\theta)$ is the Coriolis parameter and is equal to

$$f(\theta) = 2\Omega \sin\theta. \tag{51}$$

Ω is the radial velocity of the Earth and is measured to $7.29 \cdot 10^{-5}$rad/s and θ is the latitude. The value of c_1 can be found by solving a Sturm-Liouville problem for the structure of the vertical component of the velocity. The value of c_1 varies with latitudes and ocean depth. Chelton et al (1998) presents a map of this geographical variability. For the Norwegian coastline $c_1 \approx 1$ and $R_r \approx 8$km. Note that eq. (50) only holds at latitudes higher than 5°[28].

3 Method

The work in this report builds on the method for identification of LCS in 2 dimensions described by Farazmand and Haller (2012)[7]. The method can be separated into two parts; First, lines satisfying conditions A, B and C (see eq. (16)) are found by calculating the eigenvalues and eigenvectors of the CG strain tensor, and then calculating the trajectories of eq. (21). Next a selection process is performed to extract the lines which also satisfy condition D. These are the LCSs of the system. It is this second part which is the focus of this paper. Farazmand and Haller (2012) suggested using a set of horizontal and vertical lines and then compare all of the strainlines within some distance ϵ from each intersection. This method is illustrated in figure 7 and works fairly well if some knowledge of the domain is available prior to the extraction. This domain knowledge could for example come from studying the FTLE-field, or using the Rossby radius to make an estimate based on the scale of the system. However, a poor choice of either the number of lines, or their position can lead to lines being ignored during the selection. In this project an alternative method is suggested for the identification and extraction of the LCSs. By using clustering, a form of unsupervised machine-learning, similar lines are grouped together, then the line which exhibits the strongest $\bar{\lambda}_2$ can be extracted from each cluster. This could potentially be a more robust way to identify which lines are classified as LCSs.

The program was implemented using python with the libraries *Numpy* and *Sci-kit-learn*[15, 24]. *Numpy* is a high performance linear algebra library and *Sci-kit-learn* is a high level machine learning library. The latter includes several algorithms for dimensional reduction and clustering. The double-gyre field was used for testing and calibration of the program. This flow field is given by eq. (52) and is commonly used and well suited for testing LCS theories[1]. The rest of this section describes the steps needed to calculate and identify the LCSs using the double gyre field to illustrate the effect of the dimensional reduction and clustering. Then the extracted set of LCSs is compared to the results of Farazmand and Haller (2012) using identical parameters for the flow field[7]. The results presented here are further discussed in section 4 and compared with the results from clustering on modelled ocean data.

3.1 Double Gyre

The double-gyre is the canonical flow field for testing LCS theories.[1] It is a time-periodic flow and is defined by the following stream function

$$\psi(x, y, t) = A \sin(\pi f(x, t)) \sin(\pi y). \tag{52}$$

Where the function $f(x, t)$ is given by

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

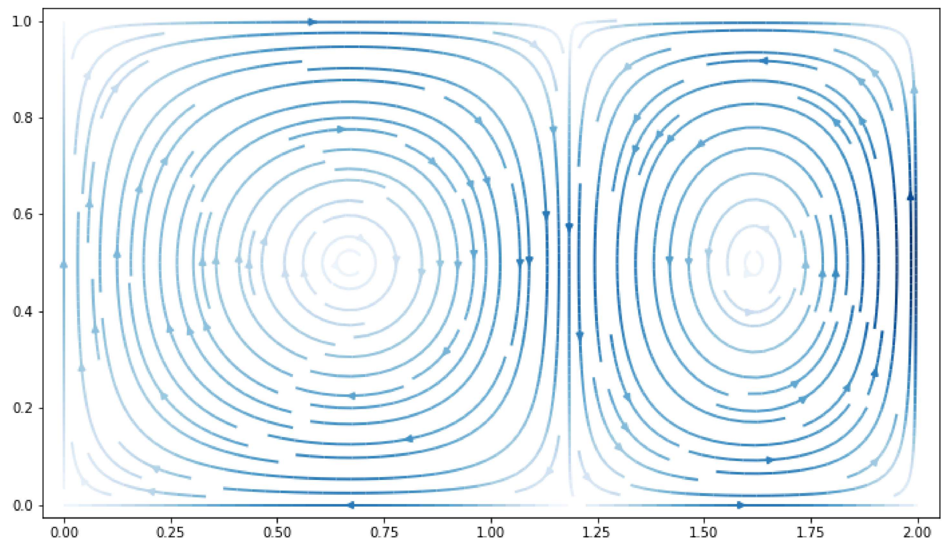


Figure 12: Streamplot of the velocity field given by eq. (53). The color represents the norm of the velocity, $|\mathbf{v}|$. A deeper blue represents a larger value for $|\mathbf{v}|$. The field is plotted at $t = 3$ for the parameters $A = 0.1, \omega = 2\pi/10$ and $\epsilon = 0.2$.

The velocity components $\mathbf{v} = [u, v]$ are the derivatives of the stream function,

$$\begin{aligned} u &= -\frac{\partial \psi}{\partial y} = -\pi A \sin(\pi f(x,t)) \cos(\pi y), \\ v &= \frac{\partial \psi}{\partial x} = \pi A \cos(\pi f(x,t)) \sin(\pi y) \frac{\partial f(x,t)}{\partial x}. \end{aligned} \tag{53}$$

If we restrict the domain to $x \in [0, 2]$ and $y \in [0, 1]$, the velocities will always be parallel to the boundaries. This means that no particles will leave the field. Furthermore, from the definition of the flow field, it can easily be proved that $\nabla \cdot \mathbf{v} = 0$,

$$\nabla \cdot \mathbf{v} = \left(-\frac{\partial}{\partial x}\frac{\partial}{\partial y} + \frac{\partial}{\partial y}\frac{\partial}{\partial x}\right)\psi = 0. \tag{54}$$

This equation states that the flow is incompressible[29]. The velocity field is presented in figure 12. The color denotes the norm of the velocity, $\mathbf{v}$. A stronger blue equals a larger $|\mathbf{v}|$. The two gyres oscillate back and forth as a function of time. The size of each gyre is also changing. The parameters used for the plots in this section is $A = 0.1, \omega = 2\pi/10$ and $\epsilon = 0.2$. The figure was made at time $t = 3$.

For fields where particles might leave the domain, extra care has to be taken regarding choice of time-interval and initial placement of particles. The extra care is necessary for fields using datasets with limited spatial size as the velocity is undefined and may for example be set to zero. This will in turn limit the amount of deformation experienced by the field resulting in erroneous values for the CG strain tensor. The figures presented in this section were made using a uniform grid $\mathcal{G}$ of 1000×500 particles covering a domain U spanning $x \in [-0.01, 2.01]$,

$y \in [-0.005, 1.005]$, where a small buffer has been added as suggested by Farazmand & Haller (2012), and an auxiliary grid around the uniform grid points, with a spacing of $\delta x = \delta y = 2 \cdot 10^{-5}$.

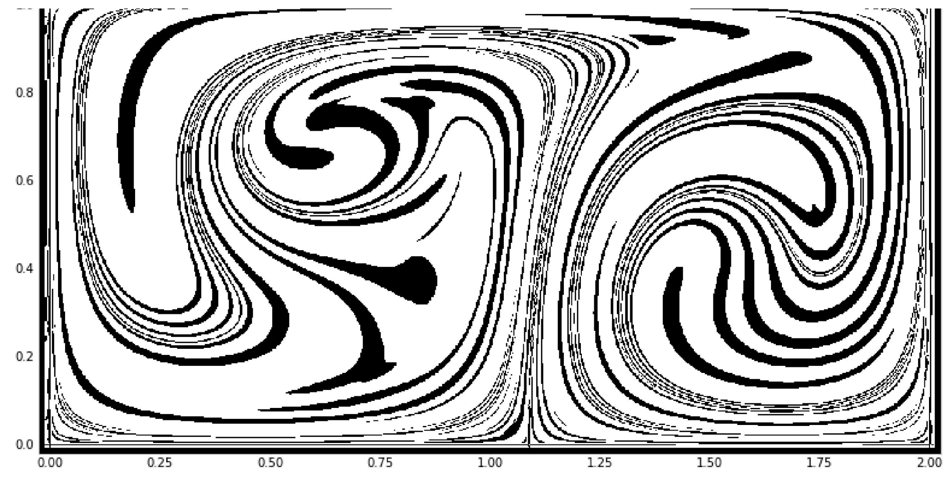
3.1.1 Strainlines

The strainlines are calculated using the eigenvalues, λ_i, and the corresponding eigenvectors, $\boldsymbol{\zeta}_i$, from the Cauchy-Green Strain Tensor (CG) (see section 2.3 for details). Figure 13 (a) shows the subset $\mathcal{G}_0 \in \mathcal{G}$, where $\mathcal{G}_0$ is all the points in $\mathcal{G}$ satisfying conditions A and B in eq. (16). This set can easily contain more than 10^5 points. The plot in figure 13 (a) shows a set of 131 661 points. Of all these points, some are bound to belong to the same strainline. Running the integration on all of them will uncover the same strainlines several times. To avoid redundant computation, a set of horizontal and vertical lines are used to extract points. Only the points satisfying the first two conditions of eq. (16) and which also coincide with one of the horizontal or vertical lines in the grid are used as initial points for the integration. This is shown in figure 13 (b). The reduced set of points is about two orders of magnitude smaller than $\mathcal{G}_0$.

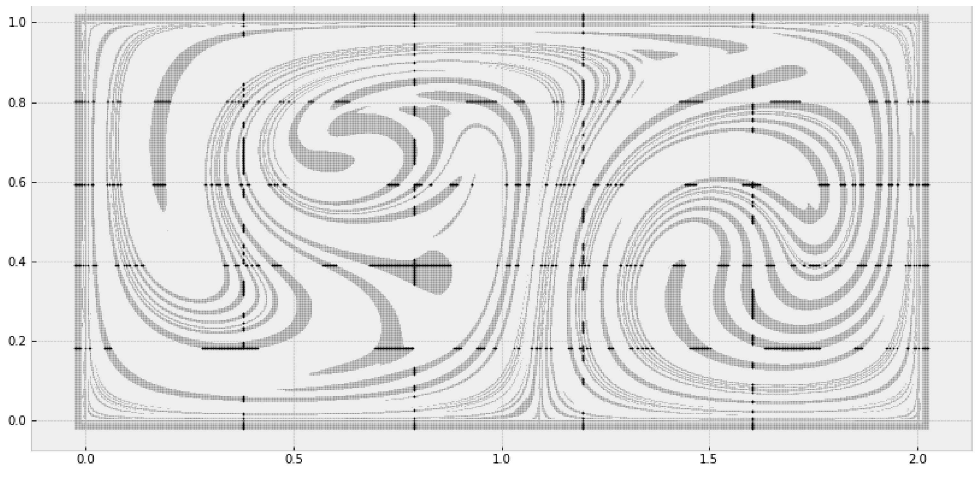
The strainlines are found by integrating eq. (21) in both directions using the points found in the reduced version of $\mathcal{G}_0$ as starting points for the integration. The integration is run until either a maximum length has been reached, the line reaches the border, or until conditions A and B has failed repeatedly. For each step of the integration, before doing any interpolation of the grid of eigenvectors, the orientation of the eigenvectors had to be checked and corrected. One gridpoint was selected as a reference, then the sign of the dot product of the reference vector with one of the other surrounding vectors would signal if the vectors had to be flipped or not. This procedure ensured that the surrounding points were rid of orientational discontinuities. Then the value of $\boldsymbol{\zeta}_1$ at the point $\mathbf{r}$ could be interpolated from the surrounding grid. Finally, the new change in the strainline had to be compared with the previous term in and flipped if necessary (See section 2.4 for details). To achieve long, smooth lines it was necessary to include a small buffer around the domain, so that the integration did not stop prematurely, which is why an extended domain has been used, $x \in [-0.01, 2 + 0.01]$, $y \in [-0.005, 1 + 0.005]$. Another reason for the integration to stop prematurely could be numerical noise in the calculation of the set $\mathcal{G}_0$. This could result in conditions A and B failing during integration, even though they should have been satisfied. Therefore a threshold length l_f should be chosen so that if the conditions fail over a length l_f the integration is stopped. In the case of the double gyre, this was chosen to be $l_f = 0.3$.

3.1.2 Identifying the LCSs

Figure 14 shows the strainlines of the system, i.e. those trajectories of eq. (21) that also pointwise satisfy conditions A and B (see sec. 3.1.1). Looking at the



(a)



(b)

Figure 13: Plot (a) shows the set of points $\mathcal{G}_0 \in \mathcal{G}$ satisfying conditions A and B in eq. (16). This set consists of more than $1.3{\cdot}10^5$ points. Running the integration for the entire set $\mathcal{G}_0$ is likely to take a long time and result in a large amount of redundant calculations. In (b) only the points which coincide with a set of horizontal and vertical lines are used, reducing the number of points to approximately $2 \cdot 10^3$, which is two orders of magnitude less.

figure one can see that there is a large number of strainlines covering the domain. Only some of these qualify as LCSs. In this figure, some pruning has already been performed as lines with length less than 1.0 have been discarded. This is because one can assume that short strainlines will not affect the trajectories at the same scale as the longer lines[7].

Condition D in eq. (16) states that for a curve, γ to be classified as a LCS, the mean eigenvalue $\bar{\lambda}_2$ of γ is maximal among all nearby curves satisfying $\gamma || \boldsymbol{\zeta}_1$. To apply condition D to the set of strainlines, a measure of similarity between the lines were needed. The Fréchet metric was used to measure the pairwise distance

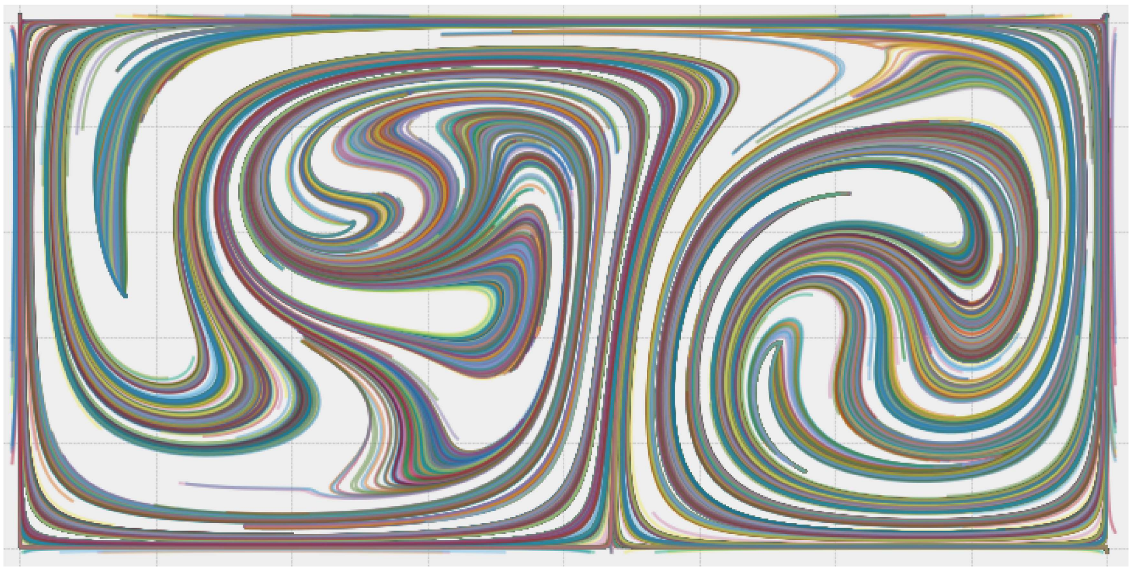


Figure 14: Strainlines (trajectories of eq. (21)), calculated using the points shown in figure 13 (b) as initial conditions. From each point, a trajectory was started in two directions and terminated when it either reached the boundary of the domain or failed to satisfy condition A and B (See eq. 16) repeatedly over some predefined distance l_f. A more detailed explanation of the procedure is given in sections 2.4 and 3.1.1.

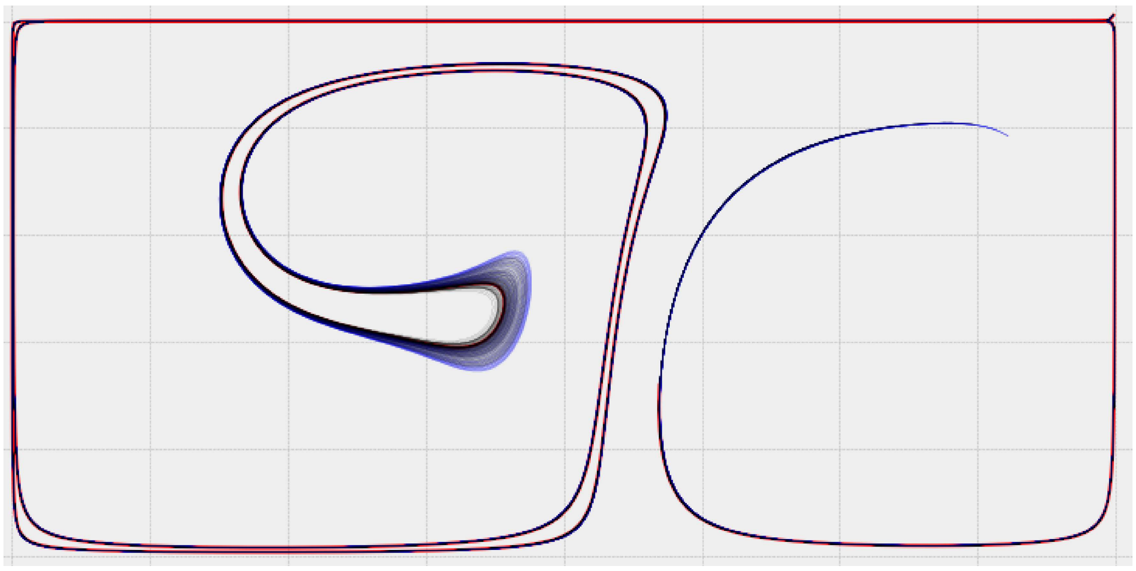


Figure 15: Plot of one chosen strainline in red and its similar neighbors in black and blue. The black lines are the lines which were most similar to the red line. The blue lines deviated a bit more from the red line, but are still a close match. The Fréchet distance was used as a measure of similarity. This was calculated using only 30 points from each line. The low resolution could introduce some errors.

between the strainlines. This was calculated between every pair of strainlines and collected in a similarity matrix, $\mathbf{S}$. Figure 15 shows one selected line in red, and the set of lines most like it. The black lines had a high degree of similarity to the red one, while the blue lines were less similar. A range of different approaches for identifying the LCSs using clustering were tested. Some of these approaches were applying clustering directly to the lines, reducing the dimensionality of the lines and the similarity matrix with different types of algorithms, and others applied clustering directly on the similarity matrix, $\mathbf{S}$.

Before going into more detail about how the dimensional reduction and clustering was applied, the overall method for identifying LCSs can be summarized by the following points:

- Calculate the Cauchy-Green strain tensor.
- Find the points in $\mathcal{G}$ which satisfy condition A and B.
- Develop strainlines, which by eq. (21) satisfy condition C.
- Perform the clustering.
- Select the most repelling strainline in each cluster as the LCSs of the system.

3.2 Dimensional Reduction & Clustering

Following the steps described in section 2.7, the first task when initiating a clustering problem was to prepare the data. The dimension of the vectors containing the strainlines were reparameterized so that all lines were represented by the same number of points. This made it possible to have the complete set of lines represented as a $2 \times N \times M$ matrix where N was the total number of lines, and M was the number of points describing each line. The factor 2 was needed because each point along a line were described by an x and y component. Furthermore, the clustering could be performed either on the lines themselves, or on a matrix containing the pairwise similarities between the lines. Additionally, in each of these two cases the clustering could be applied directly, or the dimensionality could be reduced prior to clustering using one of the procedures described in section 2.6. The similarity matrix was calculated using the Fréchet distance, which is given by eq. (49).

In the following subsection, dimensional reduction and clustering is applied on the strainlines. Some of the different algorithms are presented, such as the K-PCA and t-SNE dimensional reduction. The K-PCA was performed using a *polynomial*- and *cosine*-kernel. The t-SNE was tested using different values for the hyperparameter, i.e. the perplexity, Λ. Then the performance of three different clustering algorithms were evaluated for the double-gyre field. The subsection after that repeats the evaluation, but on the similarity matrix instead. A more thorough discussion on the different clustering algorithms and plots follow in section 4.

In order to make the clustering algorithms tackle different types of data, i.e. strainlines made using the periodic double gyre field, and strainlines made using a-periodic ocean data, some additional configuration was needed. For the DB-SCAN method this consisted of adopting an iterative scheme for finding a good value for ϵ. This value is related to the size of the clusters. The algorithm was set up with an initial value $\epsilon = \epsilon_0$. This initial value was set quite high, leading to only a small number of clusters being identified. If the number of clusters found was less than some threshold, then ϵ would be reduced. Then, if ϵ reached some minimum threshold, it was reinitialized to $\epsilon = \epsilon_0$, and then the cluster threshold wa reduced instead. This process was repeated until a satisfying number of clusters was found using a value of ϵ which was quite large. The idea was that a large ϵ would result in more generalized clustering. The Affinity propagation was initialized using a damping factor between 0.5 and 0.75. The damping factor reduces the risk of obtaining oscillating solutions. The preferences were set to the mean eigenvalues, after first centering and scaling them to unit variance. This ensured that a general scale for the different data was obtained. The scaled eigenvalues were then mapped through an inverse exponential function. An inverse exponential function was used because a strainline with preference closer to zero was more likely to be chosen as an exemplar. This resulted in the strainlines with large mean eigenvalues all having preferences close to zero. The last algorithm used was the Agglomerative-clustering. Average-linkage was used together with a k-nearest neighbor graph. The number of clusters the Agglomerative-clustering was set to find was seven for the double gyre field.

All of the following approaches for clustering the data seemed to return more LCSs than needed to describe the dynamics of the system. The pairwise Fréchet distances were therefore collected in a similarity matrix, $\mathbf{S}$, and used to prune the set of LCSs. This was done by iterating through the set of LCSs, and for each line, gather the set of similar lines in a list. Then the line with the largest mean eigenvalue from each list was extracted. This resulted in a less cluttered set of LCSs, which still seemed to describe the dynamics of the field well.

3.2.1 Clustering Strainlines

The strainlines are described by M points in $\mathbb{R}^2$. In this project $M = 2000$. Thus each line consisted of 4000 numbers. The high dimensionality of the strainlines made the clustering time-consuming and could degrade the quality of the results (see section 2.6).

Reducing the Dimensionality Using K-PCA.
The strainlines were reduced to points in ten-dimensional space using K-PCA. A fifth-order polynomial kernel was used for the transformation. Figure 16 shows four arbitrary projections of the data in a three-dimensional space. These four projections reveal that there are some indication of clusters present in the data. This indicates that the data can be clustered in a meaningful way. Data which

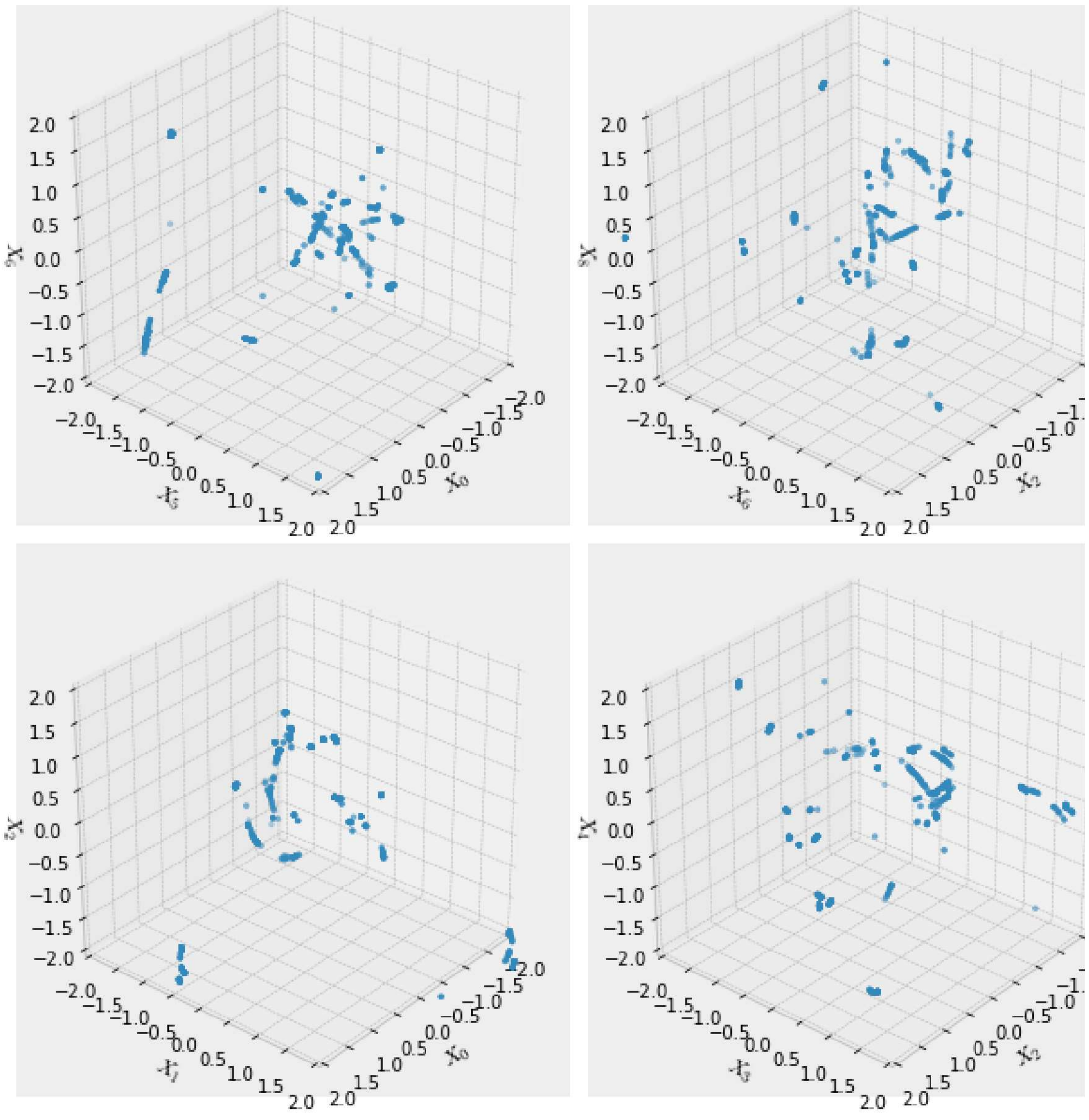


Figure 16: Scatter plot of the strainlines after being reduced to points in a ten-dimensional vector space. The dimensional reduction was performed with K-PCA using a fifth-order polynomial kernel. The plots show four random projections of the data in a three-dimensional space.

do not contain any clusters should naturally not be clustered, as most of the algorithms will find clusters anyway[8].

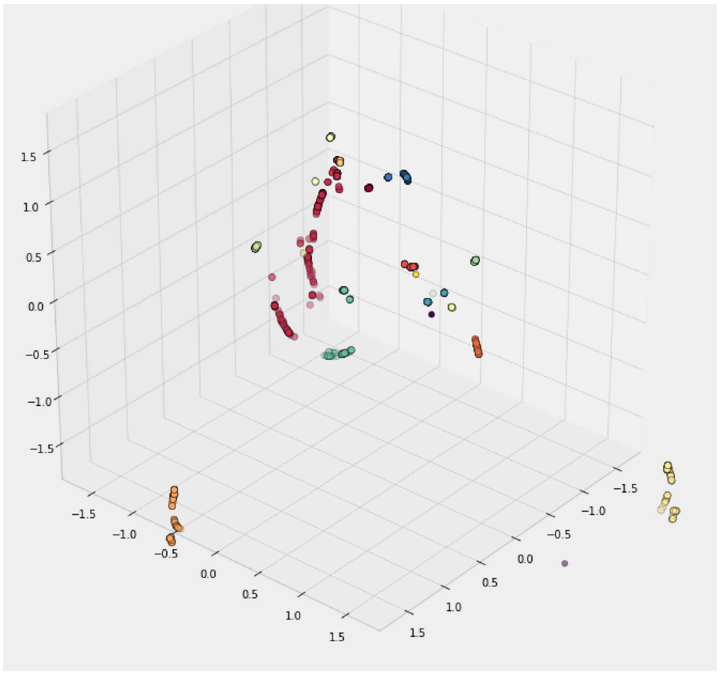
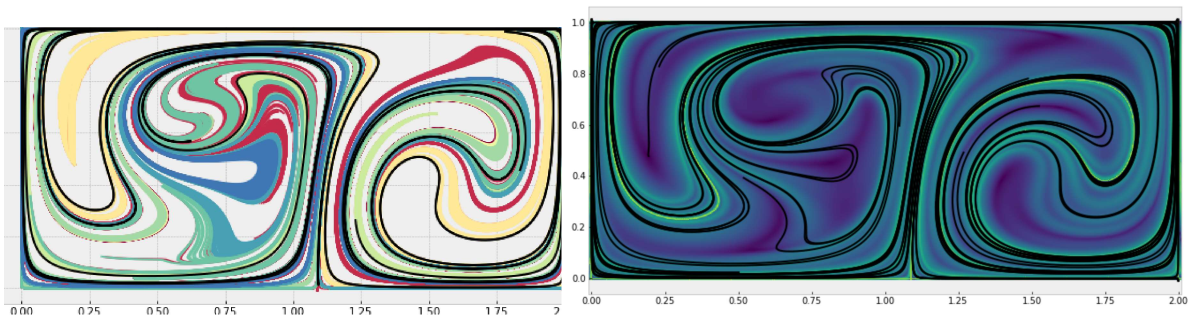


Figure 17: The resulting clusters formed by the DBSCAN algorithm. This was applied to the set of strainlines reduced to ten dimensions using K-PCA. The plot of the clustering in feature space can be a good indicator of whether the algorithm performed well on the data. In this plot we can see that most of the groups of points were identified. There are some outliers and clusters consisting of only a few points. As this data is ten-dimensional not all of the connections within clusters are visible.

The results of applying DBSCAN algorithm on the strainlines after they have been transformed by K-PCA is shown in figure 17 and 18. Figure 17 plots the resulting clusters in feature space, which in this case spans ten dimensions. Plots of the feature space can be used to judge the quality of the clustering algorithm, as it allows one to see how the clusters are formed. However, it can be hard to interpret how this corresponds to the strainlines, especially since one is often limited to projections from high dimensional space. In section 2.7 five steps to working with clustering is mentioned. The fourth and fifth step were data abstraction and evaluation. In order to make sense of the resulting clusters, the points are mapped to their corresponding strainlines. Figure 18 (a) shows the strainlines colored by group. The black lines represent noise in the clustering and are lines that do not belong to any cluster. This only applies to the DBSCAN algorithm, as all strainlines are assigned a cluster when using Affinity propagation or Agglomerative-clustering. Figure 18 (b) plots the LCSs that were classified for the system using DBSCAN. In this case, the algorithm returned 14 clusters, which results in up to 14 LCSs. Several of these lines are nearly identical, with some parts of the lines diverging from each other. To remove some of these, the lines can



(a) (b)

Figure 18: The plot in (a) shows the clustered strainlines by color. There are some lines that look as if they have been put in the wrong group. This could either be caused by the K-PCA not retaining enough of the original data, or some of the hyperparameters of the DBSCAN algorithm not being properly tuned. The black lines in (a) represents noise in the data, and are not assigned to a cluster. The plot in (b) are the LCSs of the system superimposed over the FTLE field. Looking at the set of LCSs, one can see that there are several lines which are identical. The similarity matrix could be used in order to remove some of the duplicated lines.

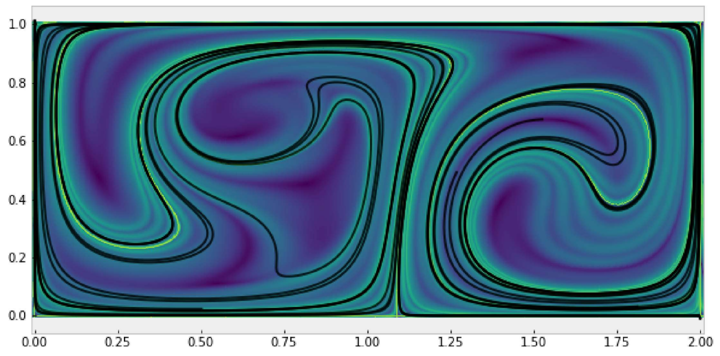
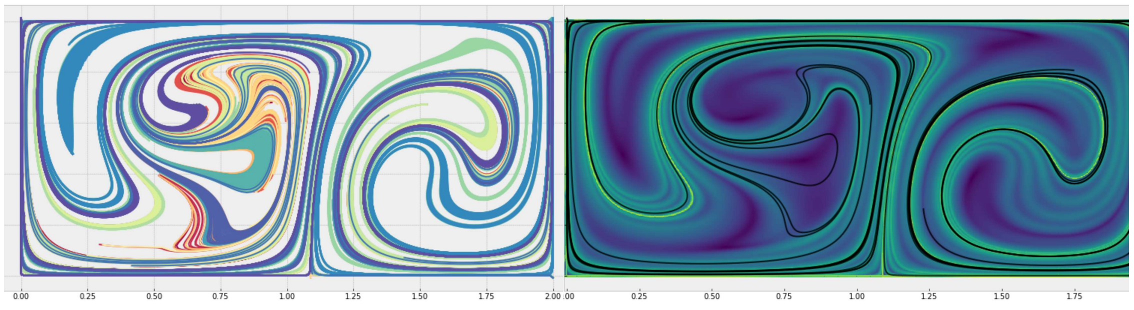


Figure 19: Plot of the set of LCSs identified using the DBSCAN algorithm. The algorithm was applied on the set of strainlines after they had been reduced to points in ten-dimensional space. In this plot, the excess LCSs have been removed using the similarity matrix, **S** (see section 3.1.1). Excess LCSs are lines which resemble other lines to a high degree, or are very close to other, stronger LCSs.

be compared using the similarity matrix, **S**. If two lines have a Fréchet distance less than some threshold, the one with the highest mean eigenvalue is kept, and the other discarded. The pruned set of LCSs is plotted in figure 19.

The resulting clusters formed by Affinity propagation is shown in figure 20 (a). This algorithm also produced a relatively high number of clusters. The plot in figure 20 (b) is the set of LCSs identified for the system after removing duplicate lines using the similarity matrix, **S**. The LCSs classified by both the DBSCAN



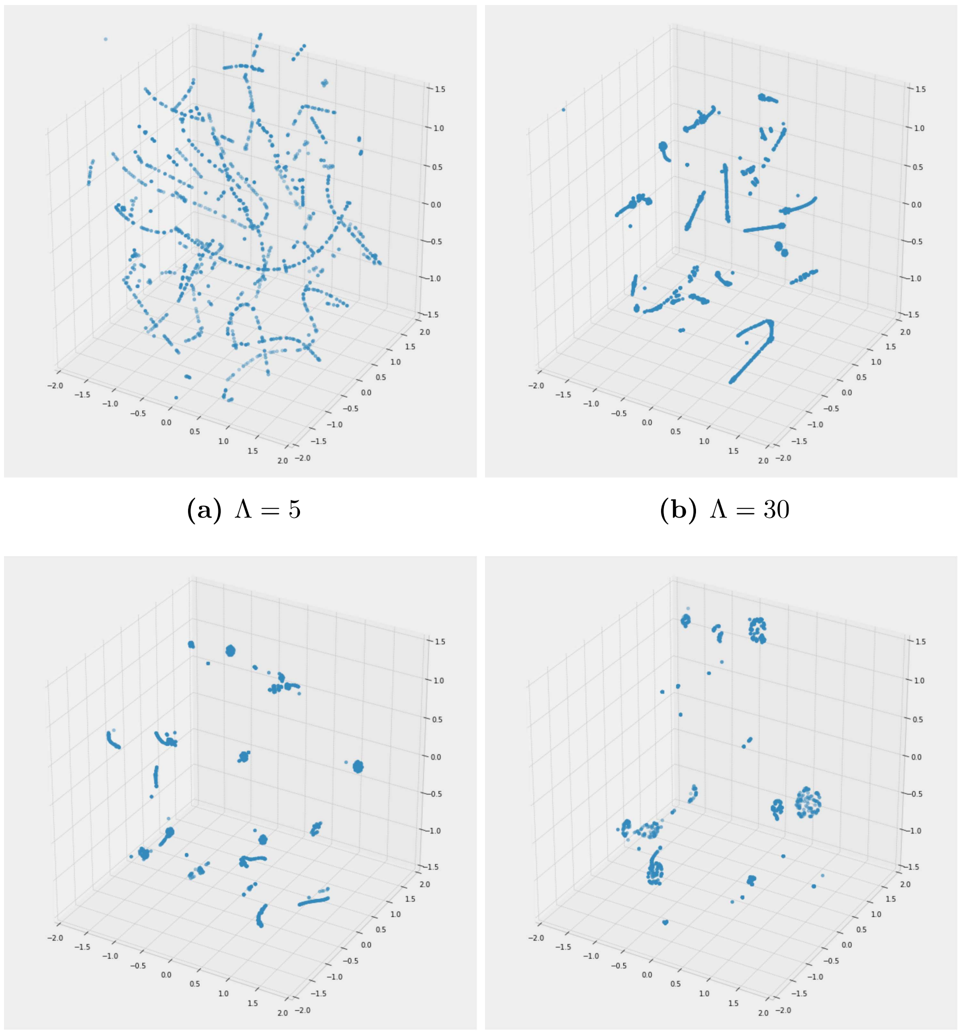
(a) **(b)**

Figure 20: The plot in (a) shows the strainlines clustered by color using the Affinity propagation algorithm. The algorithm uses a vector containing each line's respective mean eigenvalue as preference for which strainlines were to be chosen as an exemplar (See section 2.7). There are some lines that look as if they have been put in the wrong group. This could either be caused by the K-PCA not retaining enough of the original data, or some of the hyperparameters of the Affinity propagation algorithm not being properly tuned. The plot in (b) are the LCSs of the system superimposed over the FTLE field. The LCSs have been pruned using the similarity matrix in order to remove lines that closely resemble others.

and Affinity propagation algorithms are superimposed over the FTLE field. From figure 19 and 20 (b) one can see that the identified LCSs match the FTLE field quite well. There are still some lines in both plots which could probably have been removed. As mentioned above, the black lines in figure 18 (a) represent strainlines which are treated as noise, and are therefore not assigned to a cluster. The noise is related to the number of neighbors the algorithm requires for forming a cluster. A lower number of neighbors could therefore reduce the number of strainlines treated as noise. The DBSCAN and Affinity propagation do not require the number of clusters to be set beforehand, but discover this by analyzing the data. A lower number of points to form a cluster would be likely to result in more clusters to be returned by the DBSCAN algorithm. This would in turn result in a larger number of LCSs identified for the system. As both clustering algorithms already seem to return more LCSs than needed to describe the dynamics of the field, it might indicate that using K-PCA to reduce the dimensionality is not suited for this data.

Reducing the Dimensionality Using K-PCA and t-SNE.
The second approach for clustering was to use K-PCA to first reduce the dimensionality of the lines down to $d = 25$ and then to apply t-SNE to reduce it further to three. t-SNE is designed for projecting high-dimensional data down to two or three dimensions so that it can be visualized in a meaningful way. K-PCA has to be applied first because the runtime of the t-SNE scales badly with the number of input dimensions. There are some caveats when using t-SNE to analyze the data. First, the sizes and shapes of the clusters that appear after the transformation are not necessarily related to the original structure of the data. Distances between points get warped. However, the algorithm excels at separating different classes

of data, that is, uncovering the clusters. The algorithm is also sensitive to its initial hyperparameters such as perplexity and number of input dimensions. Van der Maaten (2008) states that the t-SNE algorithm should be fairly insensitive to perplexity values in the range $\Lambda \in [5, 50]$, but looking at the plots in figure 21, it is apparent that this is not necessarily the case. Figure 21 shows the t-SNE applied to the strainlines using four different values of Λ. The perplexity is related



(a) $\Lambda = 5$ **(b)** $\Lambda = 30$

(c) $\Lambda = 60$ **(d)** $\Lambda = 150$

Figure 21: Scatter plot of all the data after it has been reduced to three dimensions by t-SNE. The four plots represent different values for the chosen perplexity, Λ. A low value for Λ results in more spread in the data, resulting in a large number of small groups. Increasing the value gives larger clusters with some outliers. The perplexity is linked to the degree of which the algorithm considers local or global structure in the data. A low Λ will therefore result in the algorithm prioritizing local structure, and thus group only neighboring lines together, forming a myriad of small clusters. This can be related to the problem of overfitting, resulting in poor generalization when separating lines from each other. A larger value for Λ should therefore be used in order to keep the global structure of the data.

to the number of neighbors the algorithm considers. A low Λ results in a higher number of small clusters than for larger values. In a sense, Λ can be understood as a parameter varying the level of locality the algorithm considers. A low Λ will ensure that the algorithm prioritizes the local structure, while for a high Λ, the global structure will be prioritized. A parallel can be made to the concept of overfitting in machine-learning[21]. Overfitting is a common problem when working with machine learning and results in a low grade of generalization. In this case, the t-SNE groups only very similar lines together instead of finding groups based on the more general structure of the data. The parameters that seemed to give consistently good results for both the double gyre field, and for the ocean data in section 4 are given in table 1.

Table 1: The parameters used for the combination of K-PCA and t-SNE that worked well with both the strainlines found for the double gyre field and the strainlines found for the ocean data in section 4.

K-PCA		
	Kernel	*cosine*
	Target dimension	25
t-SNE		
	Λ	50
	Target dimension	3

The LCSs plotted in figure 22 and 23 were identified using DBSCAN and Affinity propagation. The LCSs found using DBSCAN on the set of lines reduced using t-SNE were identical to those identified using the K-PCA dimensional reduction. For the set of LCSs identified using Affinity propagation, the result was a bit better than for the K-PCA case. The lines in figure 23 seemed to match ridges in the FTLE-field better than the ones plotted in figure 20.

3.2.2 Clustering the Similarity Matrix

The complete set of strainlines consists of approximately $N \approx 10^3$ lines. The similarity matrix, $\mathbf{S}$, is an $N \times N$ symmetric matrix, where $\mathbf{S}_{ij}$ is the Fréchet distance between line i and j. The matrix is scaled by dividing it by the Rossby radius. For the double gyre, $r = 0.5$ was used as a proxy for the Rossby radius. The reason for scaling $\mathbf{S}$ is to obtain values for the distances which are on the same range for different sets of data. Using $\mathbf{S}$ instead of using the strainlines directly reduces the number of features from 4000 to approximately 1000. This is a good improvement with regards to complexity, but it is still a high dimensional feature space. In addition to the lower number of features, using a custom measure of similarity can improve the performance of the algorithms. For example, the DBSCAN algorithm can be performed using the *Euclidean*- or *Minkowski*-metric[24]. The pairwise distance between points are calculated from the features. This, however, does not account for the orientation of the lines, length, and the fact that the points are not uniformly spaced along each line. The Fréchet distance considers

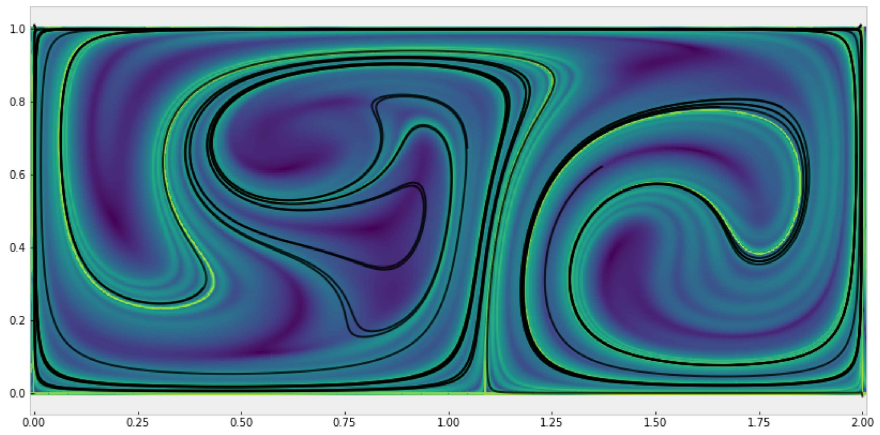


Figure 22: Plot of the resulting LCSs identified for the system using the DBSCAN algorithm. The strainlines were first reduced to points in three dimensional space using t-SNE prior to the clustering.

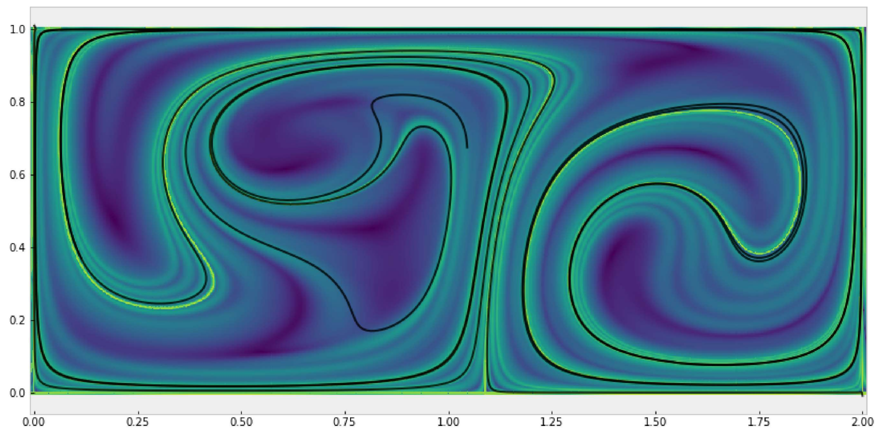


Figure 23: Plot of the resulting LCSs identified for the system using the Affinity propagation algorithm. The strainlines were first reduced to points in 3 dimensional space using t-SNE prior to the clustering.

both the location and ordering of the respective points (see section 2.8)[22]. For clustering on the similarity matrix, **S**, two approaches were tested; Reducing the dimensionality with K-PCA and t-SNE, and clustering **S** directly.

Reducing the Dimensionality Using K-PCA and t-SNE.
The K-PCA and t-SNE were applied using the parameters listed in table 1. This procedure gave a structure which exhibited several groups of points, each group representing a set of strainlines. The new representation is plotted in figure 24.

The clusters are of various shapes and sizes. By inspection of the data, all three algorithms should be able to find the clusters present.

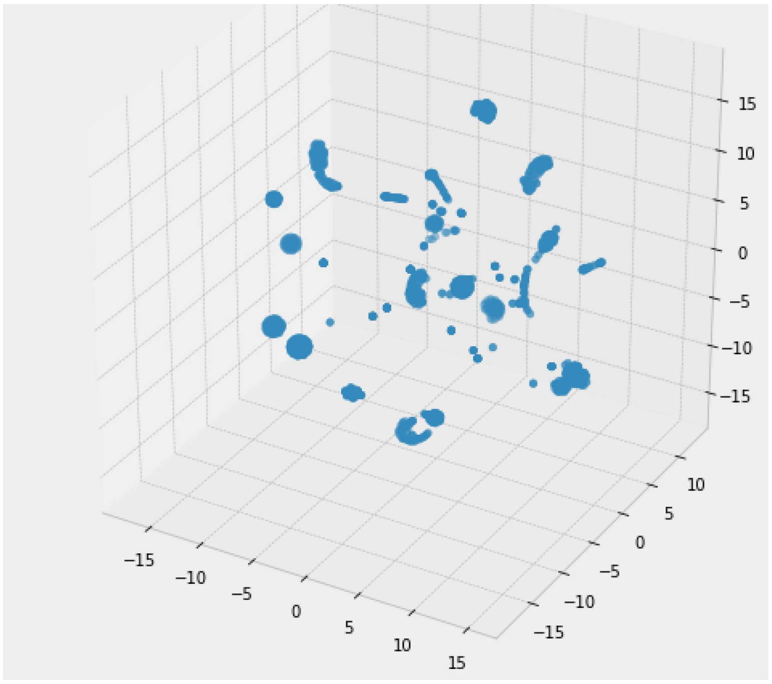
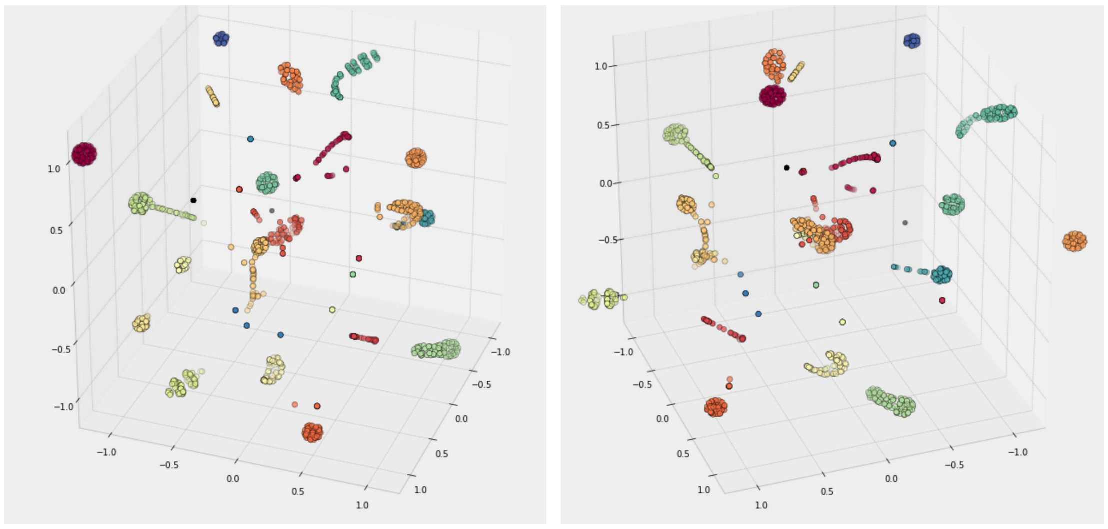


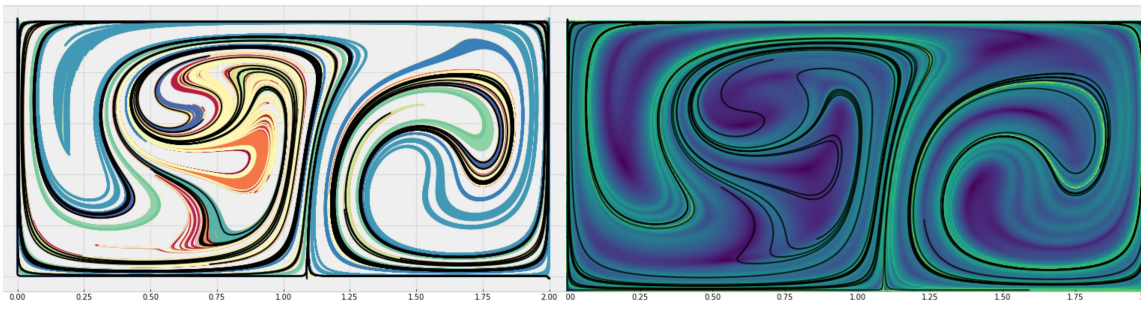
Figure 24: Plot of the similarity matrix, **S**, after being reduced to points in a three dimensional space using the t-SNE algorithm. It is apparent that there are clusters in the data, or at least in the low-dimensional representation of the data. The t-SNE algorithm excels at separating data, but the size and shapes of the clusters might be meaningless[11].

The clusters formed by the DBSCAN algorithm is plotted in figure 25. Visual inspection of the resulting clusters indicates that the DBSCAN algorithm did a good job with finding the clusters. The points colored black are not assigned to any of the clusters. The figure shows the clusters from two different angles, so that it is easier to see which points are assigned to each group. This representation indicates that the algorithm found, and clustered the points according to the underlying structures seen in figure 24. There are several outliers which have been assigned clusters. The clustering is then mapped back to the strainline representation. Figure 26 (a) plots the lines colored by group. Note that the colors of the clusters from figure 25 are not directly linked to the clusters in figure 26 (a). The plot in 26 (b) shows the LCSs of the system identified from the clusters in 26 (a). There are more strainlines identified as LCSs when clustering the reduced representation of **S**, than what was found when performing the clustering on the strainlines. The reason for more lines being identified as LCSs could be a too low value for Λ resulting in the t-SNE algorithm to focus on the local structure. This would result in a larger number of small clusters, which leads to more strainlines being chosen as LCSs. Figure 24 shows some outlier points as well as several clusters which look as if they only contain a small number of points. Figure 27 (a)



(a) **(b)**

Figure 25: Plot of the clustering performed by the DBSCAN algorithm. It groups dense regions, resulting in clusters of varying shapes and sizes. It looks as if the algorithm correctly identified all the groups present in the data. In order to evaluate whether the clustering is meaningful, it has to be mapped back to the strainline representation in $\mathbb{R}^2$.



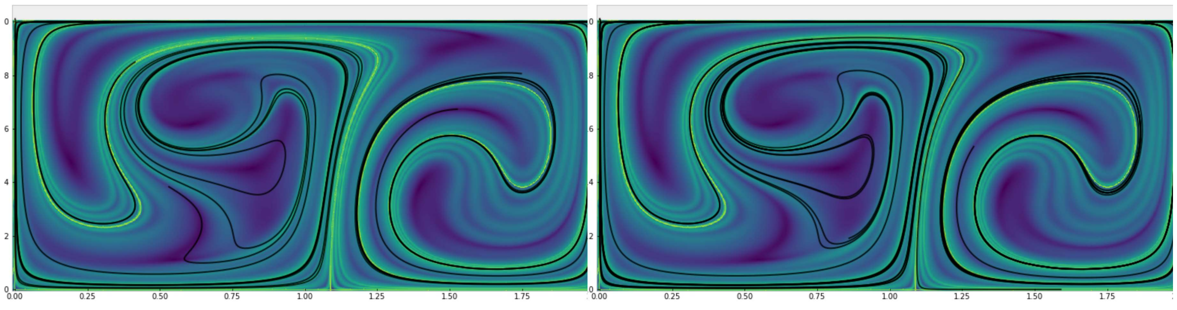
(a) **(b)**

Figure 26: In (a) the strainlines are colored by group. The black lines in (a) are treated as noise. By reducing the number of neighbors the DBSCAN algorithm considers, less strainlines will be treated as noise. The resulting LCSs of the system is presented in (b).

and (b) plots the set of LCSs extracted for (a) Agglomerative-clustering and (b) Affinity propagation. The Agglomerative clustering was set to find seven clusters in the data. Both clustering algorithms identified more lines as LCS than for the method based on clustering of the strainlines. This tendency of finding too many clusters in the data can be contributed to the t-SNE dimensional reduction. As mentioned above, a too low perplexity value might have resulted in the algorithm considering only local structures present in the data.

Clustering Directly on the Similarity Matrix.

The final approach was to apply the clustering algorithms directly on the similarity matrix, **S**. The dimensionality of the feature space makes it impractical to



(a) (b)

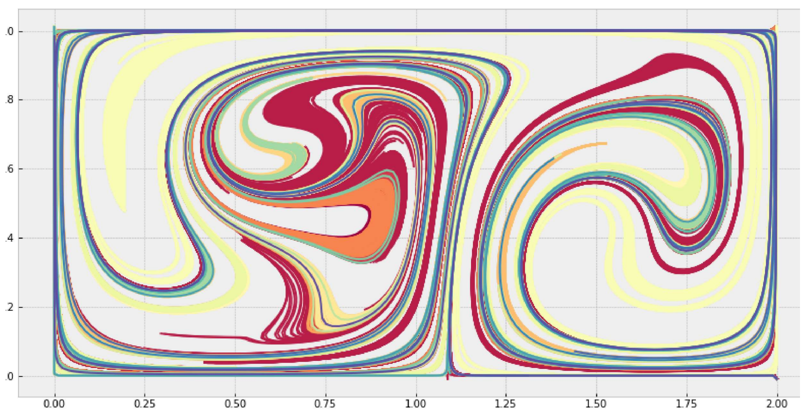
Figure 27: The figure shows the LCSs of the system located using the Agglomerative-clustering and Affinity propagation respectively. The Agglomerative-clustering was set to group the strainlines into seven clusters. The clustering was performed on the similarity matrix, **S**. **S** had first been reduced to three dimensions using the t-SNE dimensional reduction algorithm.

plot a projection down to two or three dimensions, and the resulting projections would be difficult to extract information from. It is likely that the plot will show a dense line with randomly clustered points. It is therefore more useful to study the plot of strainlines colored by their respective cluster. Figure 28 (a) to (c) shows the clusters obtained by the DBSCAN, Affinity propagation, and Agglomerative-clustering methods. The method which gave the most promising LCSs was the Affinity propagation algorithm and the resulting lines are plotted in figure 29.

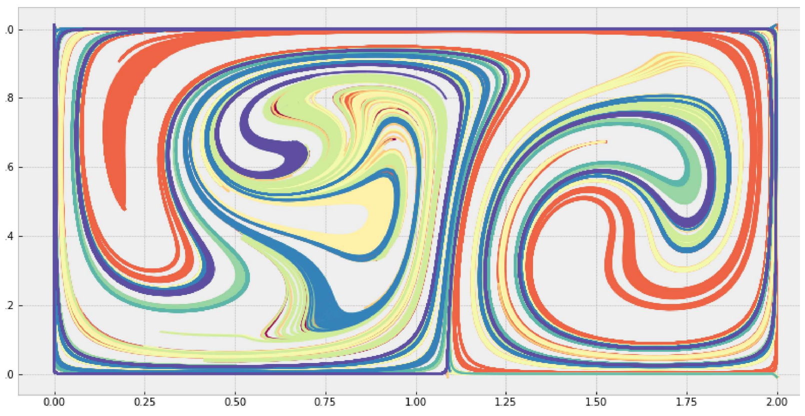
Comparing these results with a naive approach, for example, simply extracting the seven strainlines which have the largest mean eigenvalues and classifying them as the LCSs of the system, results in seven nearly identical LCSs. The resulting lines are plotted in figure 30 As mentioned in section 2.4, several of the points in $\mathcal{G}_0$ are bound to belong to the same strainline. It would seem that even if a grid of horizontal and vertical lines were used to extract a small set of points, a large number of these would still be part of similar lines. The classification using clustering and a similarity matrix results in a set of LCSs which seems to describe the dynamics of the field better.

3.3 Advecting the LCSs

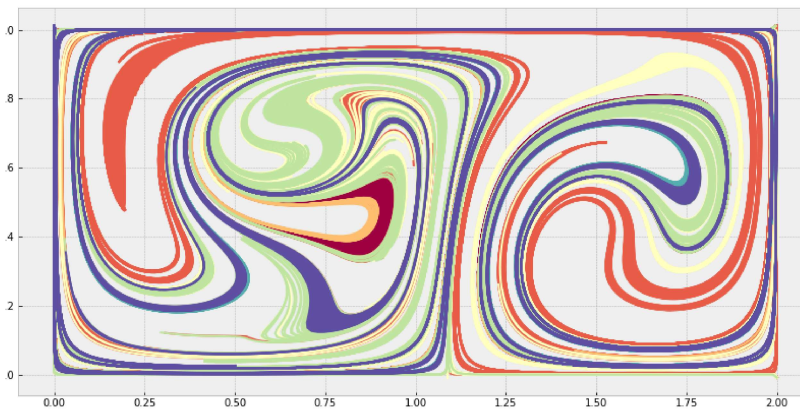
By utilizing eq. (22) to advect the LCSs found at time t_0, we can find the corresponding material lines at later times t. As stated in section 2.5, there is no guarantee that the material line remains an LCS when advected by the flow map[7]. The procedure is also prone to numerical errors when used on a repelling LCS. In figure 31 (a) the identified LCSs are plotted at $t_0 = 0$. The blue lines are attracting LCSs and the red are repelling LCSs. Figure 31 (b) and (c) shows the lines advected to $t = 3$ and $t = 6$. It becomes apparent from looking at the plots in figure 31 that the repelling lines shrink as time goes on. Some numerical instabilities are visible along the red lines around $(1.1, 0.1)$ in plot (b), and $(1.0, 0.8)$



(a)



(b)



(c)

Figure 28: The strainlines are colored by group. The three plots were created using the DBCAN, Affinity propagation, and Agglomerative-clustering respectively. The grouping performed by the Affinity propagation algorithm resulted in the set of LCSs which matched the FTLE field best. The clustering presented here was performed directly on the similarity matrix, **S**, containing the pairwise Fréchet distances between strainlines.

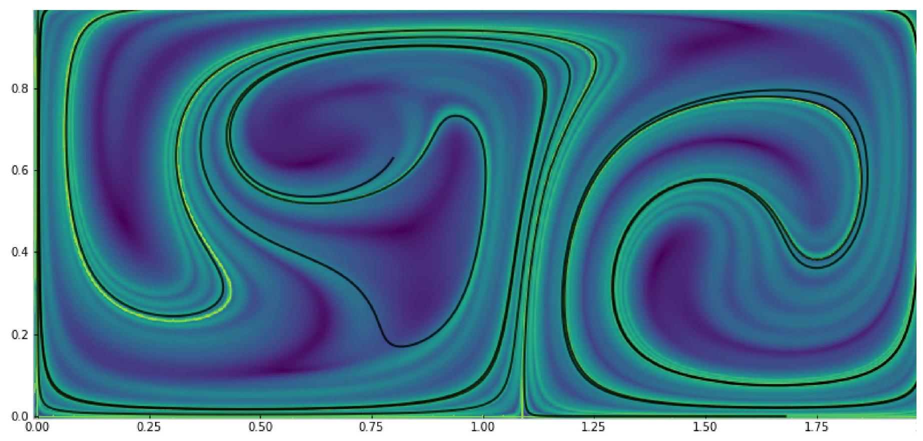


Figure 29: Plot of the set of LCSs extracted from the ensemble of strainlines. The clustering was performed on the similarity matrix, **S**, using the Affinity propagation algorithm. This was the method which achieved the results which seemed to match the FTLE field best, both for the double gyre data, and for the modelled ocean data presented in section 4.

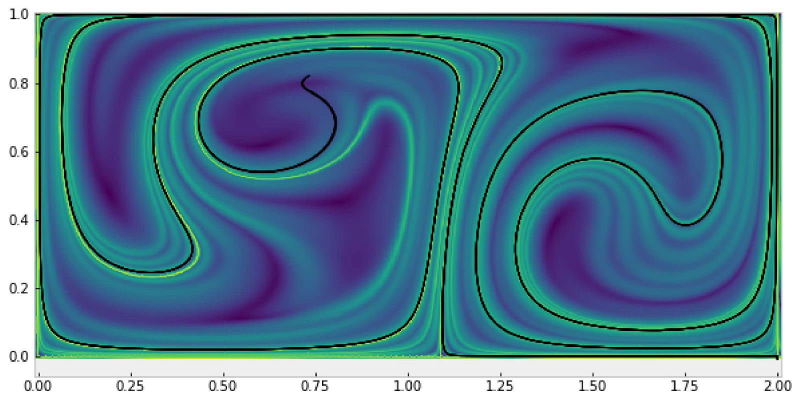
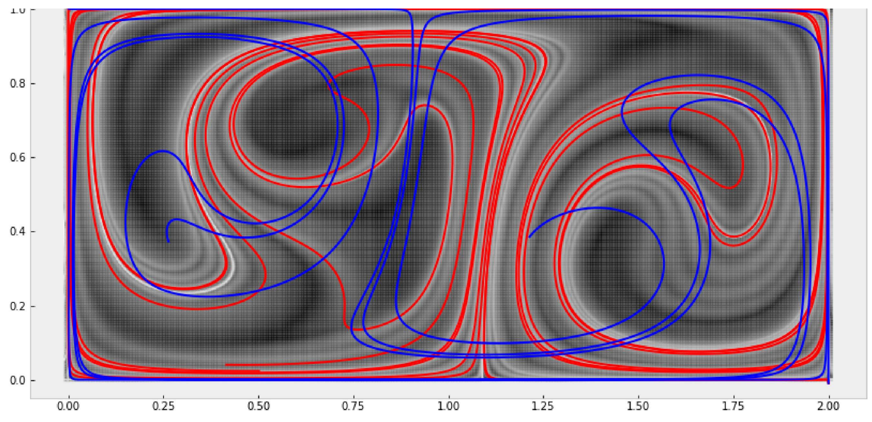
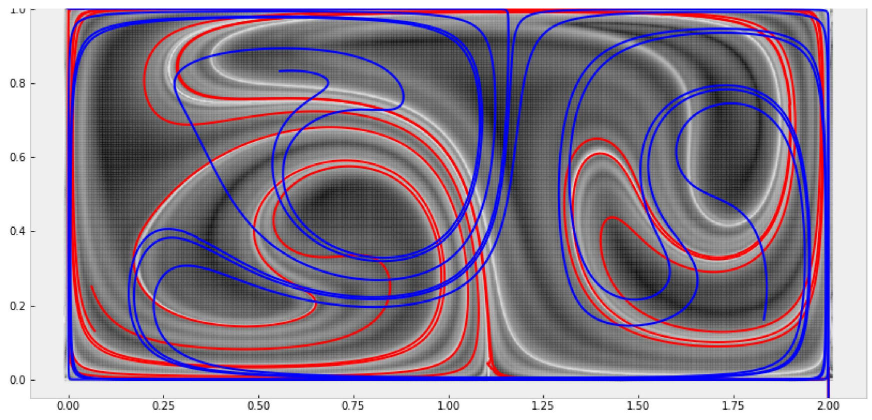


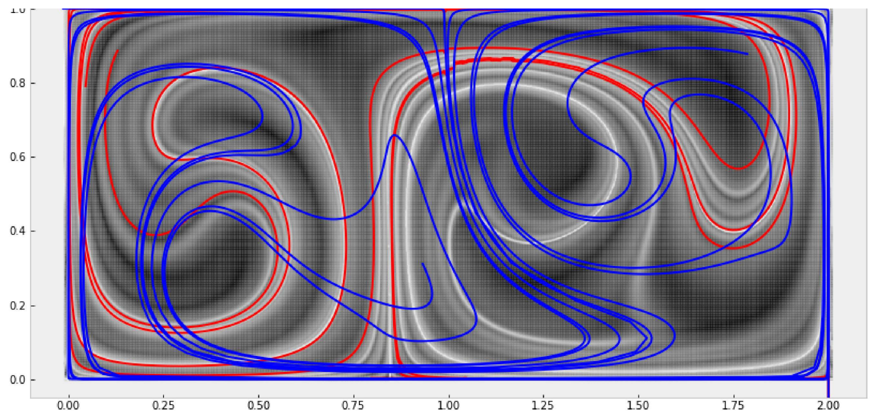
Figure 30: A naive approach to extracting LCSs from the set of strainlines. Here the seven strainlines with the largest values for $\bar{\lambda}_2$ were chosen as LCSs. This resulted in seven nearly identical LCSs. When comparing this with the LCSs extracted using clustering, i.e. the set of LCSs in figure 29, one can see that the clustering obtained a set which seemed to match the FTLE field better.



(a)



(b)



(c)

Figure 31: The set of attracting (blue) and repelling (red) LCSs are advected from $t = 0$ to $t = 6$ in the double gyre field. The LCSs are superimposed over the FTLE field which is calculated using the forward time interval. This results in the field matching the repelling LCSs. In plot (b) the field has been advected to $t = 3$. The repelling lines seem to match the FTLE field well, but there are some additional ridges in the field which indicate that other LCSs might also be present. As the lines are advected by the field, the repelling lines shrink, and the attracting lines stretches. In plot (c) at $t = 6$ one of the strongest ridges in the FTLE field is still covered by a repelling LCS, but it is apparent that there is more dynamics to the right side of the plot, not being explained by the LCSs. Looking at plot (c) one can also note that the attracting lines have become longer.

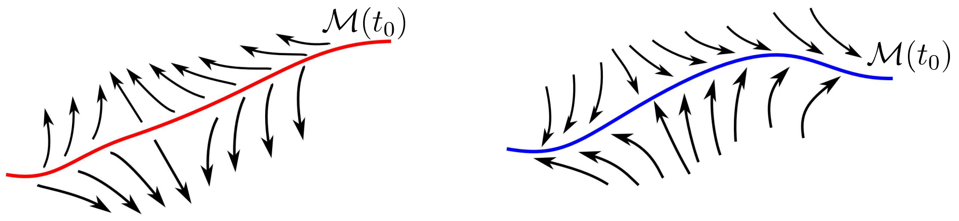


Figure 32: The direction of the flow of particles around an attracting material line (blue) and around a repelling material line (red). The flow stretches and shrinks the respective material lines. This means that as the field is advected, the repelling lines will gradually disappear.

in plot (c). The attracting lines however are stable and stretch with time. This behaviour is expected and can be understood by looking at how the flow acts on attracting and repelling lines, which is shown in figure 32. This results in the blue lines dominating the field as time goes by. The lines are superimposed over the FTLE field calculated for the forward time intervals. While the advected lines describe the dynamics fairly well for a period of time, the appearance of new ridges in the FTLE field indicates that new LCSs might have appeared.

3.4 Evaluating the Clustering

An easy way to test the validity of the results is to compare the resulting set of LCSs to the results presented in the paper by Farazmand and Haller (2012)[7]. By using the grid to extract the LCSs from the field, they managed to find a single LCS which covered the whole domain. The corresponding LCSs found using the approach presented in this section is plotted in figure 33. The selection was made using Affinity propagation directly on the similarity matrix, **S**, as this approach showed the most consistent results on the various types of data. It is apparent that this approach identifies a few more strainlines as LCSs. There can be several reasons for this. First, limitations in the resolution used for calculating the Fréchet distance between the lines could contribute to inaccuracies in the pairwise similarities in **S**. Another point to consider is that there were no strainlines that matched the single line identified in Farazmand and Hallers article. This could be because of differences in the implementation of the strainline calculations, or some tweaking of the algorithm that was left out of the article. Finally, it is not certain that the single LCS discovered there was the only LCS of the system. This project aims at finding a more objective way for extracting the LCSs from the strainlines. Farazmand and Hallers technique of searching for lines close to intersections of a seemingly arbitrary grid introduces the possibility that some LCS are left out.

The LCSs presented in figure 33 matches previous results in Farazmand and Haller (2012) well. The overlap of LCSs to the left of the figure is troubling. The reason for the overlapping lines is most likely caused by the different lengths of the lines making the values in **S** between them blow up. This causes similar lines to not

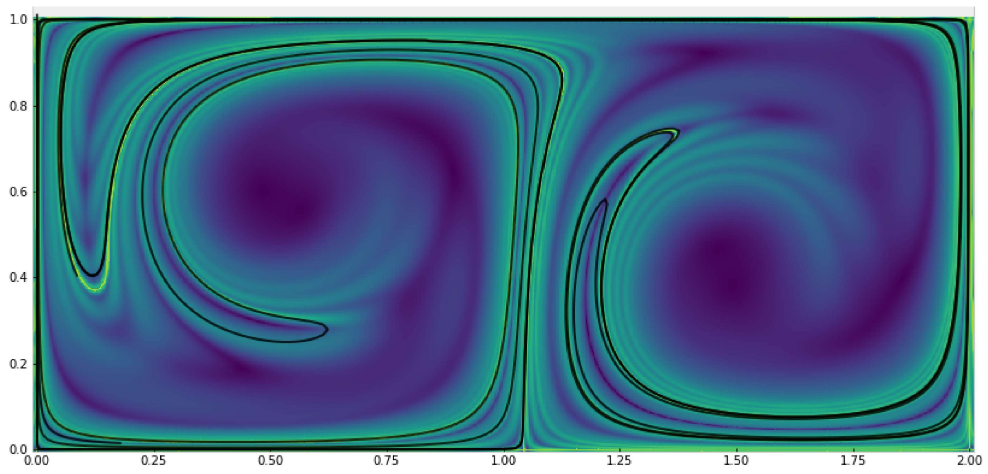


Figure 33: The LCSs extracted for the same system studied by Farazmand and Haller (2012)[7]. The LCSs found using Affinity propagation to cluster the similarity matrix gives a neat set of lines. While there are some differences to the LCS identified in Farazmand and Hallers paper, the overall structure is the same. There are some overlapping lines, especially at the left side of the figure. These lines probably deviate too much from each other at other areas of the flow so that the pruning applied after clustering does not remove them.

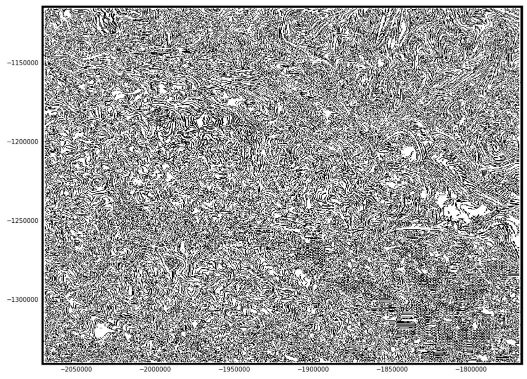
be removed after the clustering. This could likely be contributed to the lacking resolution used for calculating **S**. However, the LCSs still match the FTLE-field, capturing the dynamics of the flow.

4 LCSs in Modelled Ocean Data

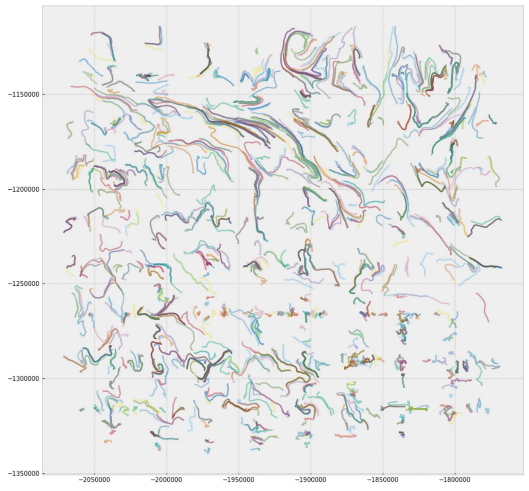
A set of ocean velocimetric data was retrieved from the Meteorological Institute of Norway[30]. The data is available with a temporal resolution of 1 h and a spatial resolution of either 4 km or 800 m. Figure 34 shows the set of points satisfying condition A and B (see eq. (16)) and the resulting strainlines for two different areas of the Norwegian coastline using the 800 m and 4 km data sets. The integration was performed for up to 10^4 steps, with a stepsize of 36 m. The set of strainlines uncovered using the 800 m data consist mostly of smaller lines, not revealing much of the overall dynamics of the field, while the 4 km data captures the more general skeleton of the flow. Therefore, the remaining plots presented in section 4 were made using the Nordic 4 km data between 10 a.m., 23.03-2018 to 10 a.m., 29.03-2018, spanning latitudes 63°to 76°, and longitudes ÷15°to 20°. In this project, only two dimensional LCSs were considered so that only the surface layer of the data was used. All of the figures in section 4.1 to 4.3 were made using the backwards time interval $[+72\text{ h}, 0\text{ h}]$. The clustering was also applied to the forward interval, but the plots were not included due to the large number of figures. The location and time for the set of data was selected by studying the evolution of the FTLE field, looking for areas that exhibited ridges in the FTLE field. Note that the scale of the eddies in the field is relatively small compared to the size of the domain, e.g. the domain spans about $100R_r$ compared with the $4R_r$ spanned in the double gyre case. Using a set of only four horizontal and vertical lines to reduce the set $\mathcal{G}_0$ is therefore likely to miss a lot of the eddies present in the flow. Instead eight lines were used along each axis. If too few lines were to be used for the flow being studied, the structure of the grid would reveal itself in the formation of the strainlines, which would mean that LCSs that dominate the regions between the lines would remain hidden. The strainlines calculated using the modelled data with 800 m resolution illustrates this problem in figure 34 (b).

4.1 Identifying LCSs in Modelled Ocean Data

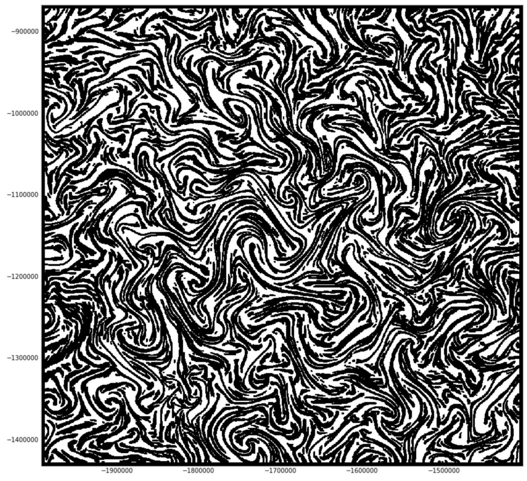
Unlike the double gyre field, modelled ocean velocity data is a-periodic and a lot more chaotic. Figure 34 (c) and (d) shows the set $\mathcal{G}_0$ of points satisfying condition A and B, and the corresponding strainlines, for a patch of ocean outside the Norwegian coastline. The field was computed using an interval of 72 h. In figure 35 (b), a total of eight horizontal and eight vertical lines were used for the reduction of $\mathcal{G}_0$. The points which coincided with the set of lines were used as initial values for the integration of strainlines. This meant that the lines were spaced approximately 50 km apart, or about $5R_r$. The larger set of horizontal and vertical lines results in a larger number of points which coincide with the lines. Therefore the development of the strainlines is more computationally intensive than it was for the double gyre field. If the grid pattern is visible when plotting the material lines, it means that not enough horizontal and vertical lines were used in the process and the calculation can miss the LCSs located in between the grid lines. The domain of points that satisfy condition A and B of eq. (16) is plotted in figure 35 (a). This plot consists of nearly $3.4 \cdot 10^5$ points. Black represents points were the conditions



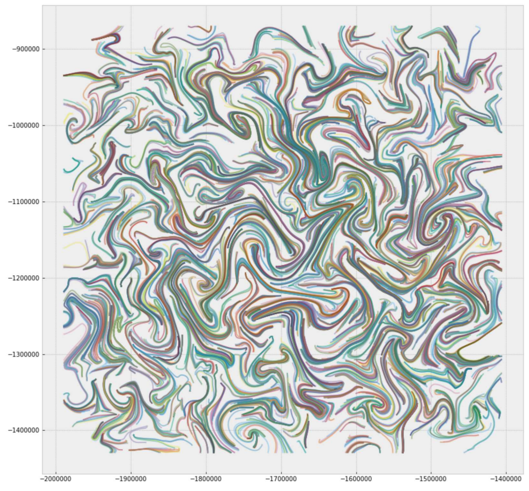
(a)



(b)



(c)



(d)

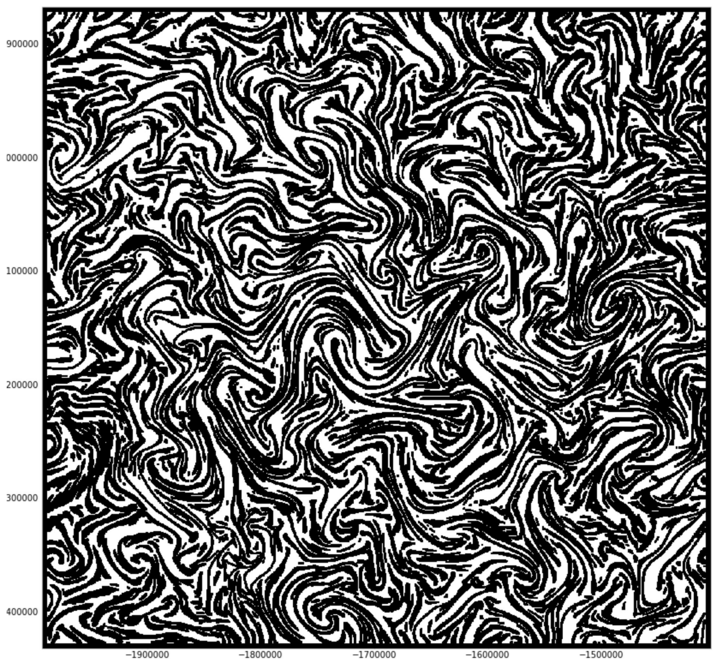
Figure 34: The plot in (a) and (c) presents the set of points $\mathcal{G}_0$ which satisfies conditions A and B in eq. (16). The plot in (a) was made using modelled ocean data with a spatial resolution of 800 m, while the plot in (c) was made using data with a spatial resolution of 4 km. The resulting strainlines are presented in (b) and (d). The strainlines obtained from the 800 m dataset are much shorter and seem to describe more local features. The set of horizontal and vertical lines is clearly visible through the pattern of strainlines (see section 2.3). The strainlines computed from the 4 km data are much longer and presumably more representative of the global dynamics of the field.

are satisfied. The reduced set of points which coincide with the horizontal and vertical lines consists of about $5.5 \cdot 10^3$ and is plotted in figure 35 (b). This is a reduction of nearly two orders of magnitude.

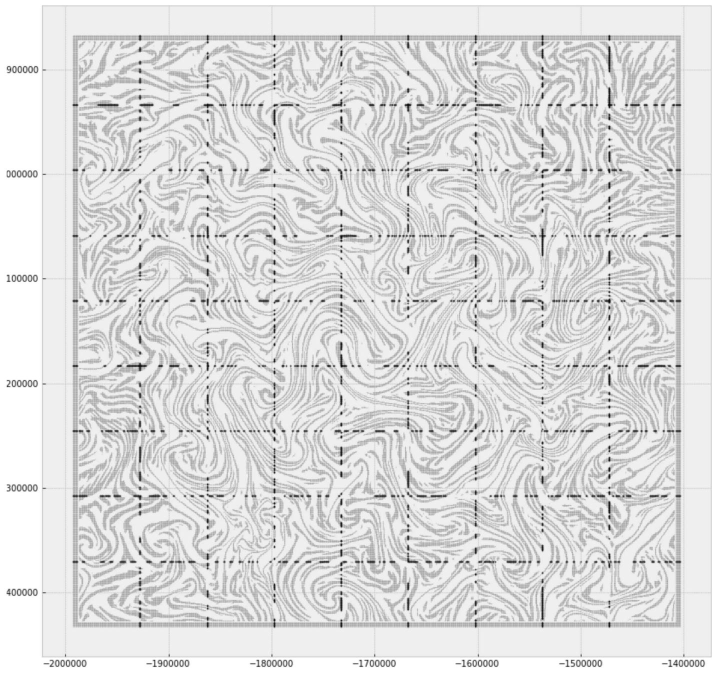
An important feature which requires extra attention is the size of the domain. When the particles are advected through the field, there is a risk of some leaving the domain. For the double gyre, the domain was restricted so that no particles could leave, but for modelled ocean data this is not the case. Particles leaving the modelled field also poses a larger problem than if particles left the analytic field. Outside of the domain in the case of modelled data, no information regarding the velocity field is known. Extrapolating the values is likely to give poor results. Therefore it is important to add a large enough frame around the field of interest, so that not too many particles will leave the domain before the advection has finished. If a large number of particles leave the domain during the advection, the resulting deformation will depend on the domain size. This will give erroneous results for the CG strain tensor.

The distance between lines consisting of points satisfying the two conditions ranges from 5 km to 15 km. Small numerical errors might result in the strainlines being integrated up slightly outside regions in $\mathcal{G}_0$ where conditions A and B are met. Therefore the integration was allowed to fail consecutively over a length of up to $l_f = 5$ km. The set of strainlines calculated for the ocean data is presented in figure 36 for both the forward- and backwards-time integration. A clear difference between the task of clustering the strainlines found for the double gyre and the strainlines presented in figure 36 is the number of clusters. The relative size of the eddies compared to the size of the domain implies that a higher number of LCSs are needed to cover the field. The number of clusters to group the strainlines into must be provided for the Agglomerative-clustering, while the DBSCAN and Affinity propagation should be able to discern the number of clusters from the data. The Rossby radius is used as a scale of the system. For the Norwegian coastline, the Rossby radius is approximately $R_r \approx 10$ km. This distance was used for scaling the distance matrix and for pruning the set of LCSs identified by each algorithm. This should ensure that the resulting sets of LCSs were consistent with the type of domain.

The following subsections match those of section 3.2. The results from applying dimensional reduction and clustering on the ocean data is presented. The next section clusters using the strainlines, preprocessed by either K-PCA, or K-PCA and t-SNE to reduce the dimensionality. Afterwards follows the clustering applied on the similarity matrix, $\mathbf{S}$, of the system. The Fréchet distance is used as a measure of similarity. The values of $\mathbf{S}$ are then given by the pairwise Fréchet distances between all of the strainlines. The clustering is performed either on $\mathbf{S}$ directly, or on a more compact representation of $\mathbf{S}$, where the dimensionality has been reduced using t-SNE first. The results are compared against their counterparts in section 3.2.

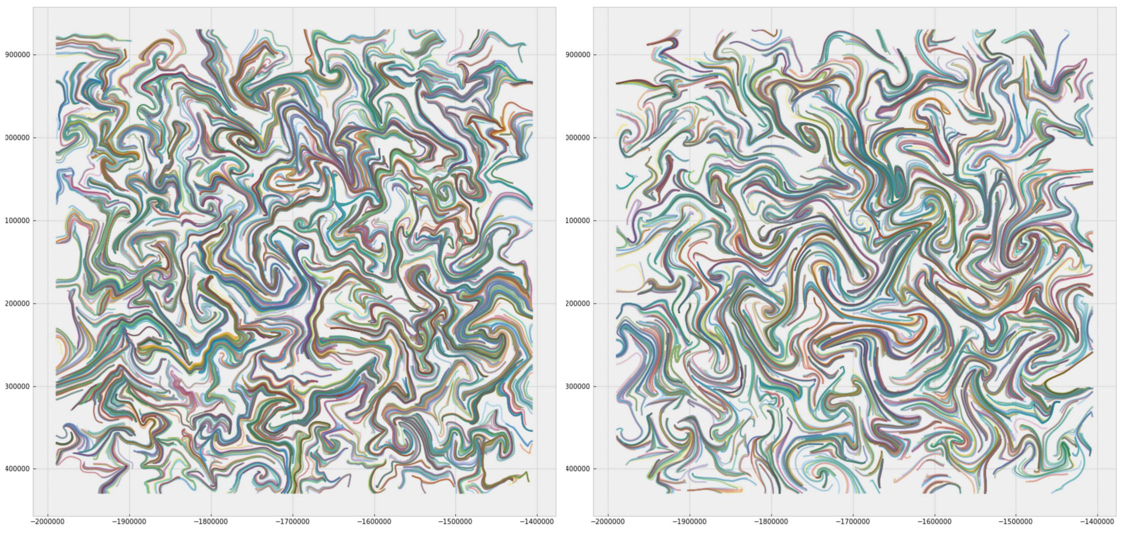


(a)



(b)

Figure 35: The set of points satisfying conditions A and B of eq. (16) for the selected area in the 4 km dataset. The plot in (a) presents the entire set, $\mathcal{G}_0$, where black represents the points where the conditions are satisfied. Performing the integration (see section 2.4) on this set will calculate the same strainlines multiple times. Therefore a subset is extracted by only using the points which coincide with a set of lines. This is illustrated in (b). The number of points coinciding with these lines is about two orders of magnitude less than the total number of points in $\mathcal{G}_0$.



(a) **(b)**

Figure 36: The two plots presents the set of strainlines calculated for the forward and backward time intervals respectively. The maximum distance the integration was allowed to fail satisfying conditions A and B (See eq. (16)) was set to $l_f = 5$ km. A total of 16 lines were used to extract the initial points for the integration.

4.2 Clustering the Strainlines

All strainlines were resampled to vectors comprised of 2000 points in $\mathbb{R}^2$. The high dimensionality could be problematic as it would increase the complexity of the clustering task, and it might degrade the results. Therefore, finding a method to reduce the dimensionality, without losing much of the information hidden in the data, could prove worthwhile.

Reducing the Dimensionality Using K-PCA

The K-PCA dimensional reduction gave promising results when applied to the strainlines in the double gyre field. The strainlines calculated for the ocean data were reduced to points in ten-dimensional space by K-PCA, using the same parameters as for the double gyre case. The kernel used was a fifth-order polynomial kernel. Figure 37 shows the ten-dimensional data projected onto four different three-dimensional spaces. The type of structures in the data that were visible for the double gyre field are missing for the modelled ocean data. The distribution of points in figure 37 reveals no obvious clusters, but rather resembles random noise. It might therefore look like the K-PCA method works better for certain types of data. The strainlines found in the double gyre case were all somewhat resembling each other. The length of the lines were more uniform, and they were smoother. The lines uncovered in the modelled ocean data were more erratic. They were of a lower relative size compared with the domain, the orientations of the lines were more random, and there was a much larger number of clusters present.

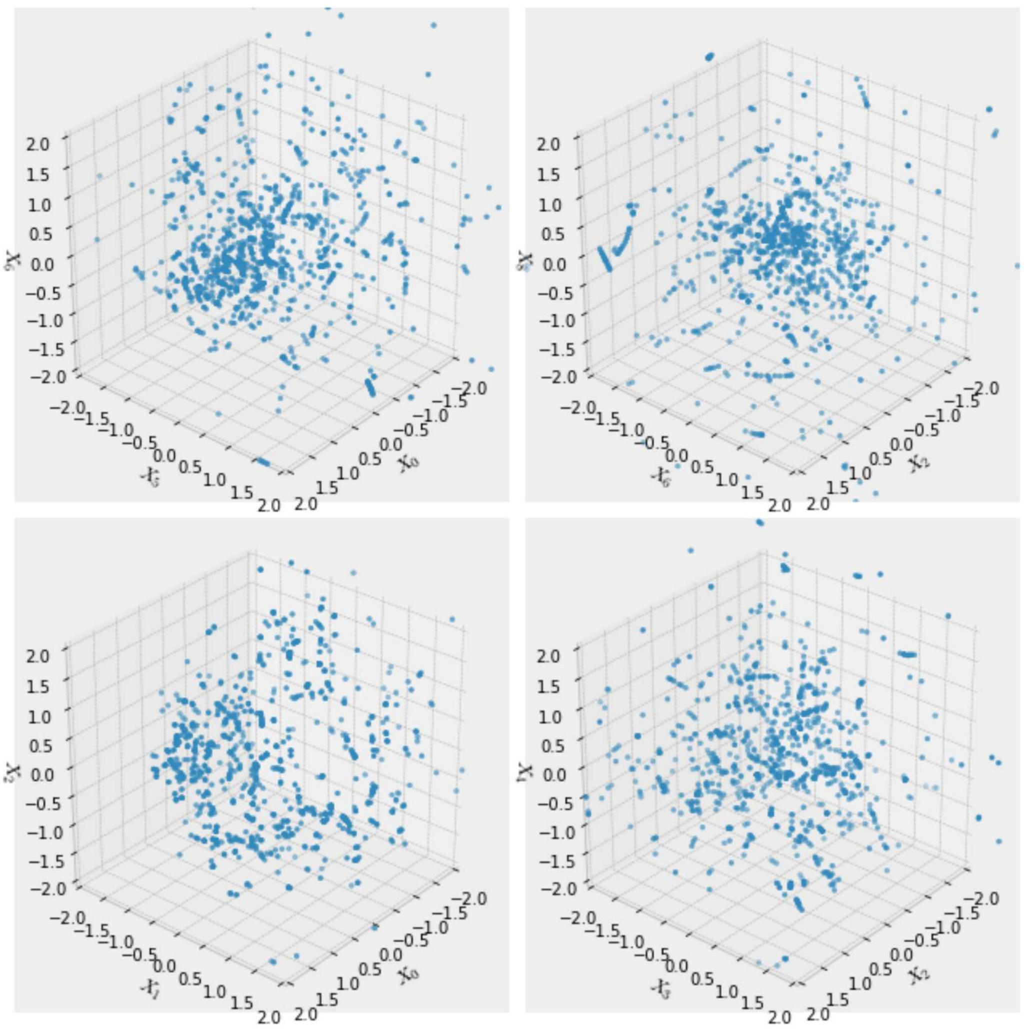


Figure 37: K-PCA was applied to the data, reducing the high dimensionality of the strainlines to points in ten dimensional space. The plot presents four random projections of this ten dimensional data. Comparing with the corresponding plot in figure 16 it can be seen that there is less structure for the ocean data. The plots mostly resemble noise in this case, while there was some structure visible for the double gyre. This could indicate that the dimensional reduction removes too much of the information hidden in the data for it to be useful on the modelled ocean data.

From the plots in figure 37, the clustering is expected to give poor results, as data resembles random noise. Therefore no informative grouping of points is expected. The clustering achieved using DBSCAN is presented in figure 38 (a). Only a few clusters were identified and most of the lines were put into a single cluster, while the other clusters only consist of two or three lines. The Affinity propagation algorithm fared better than the DBSCAN. A total of 47 clusters were identified. The set of LCSs discovered using the clustering presented in figure 38 (b) consisted of approximately 30 lines.

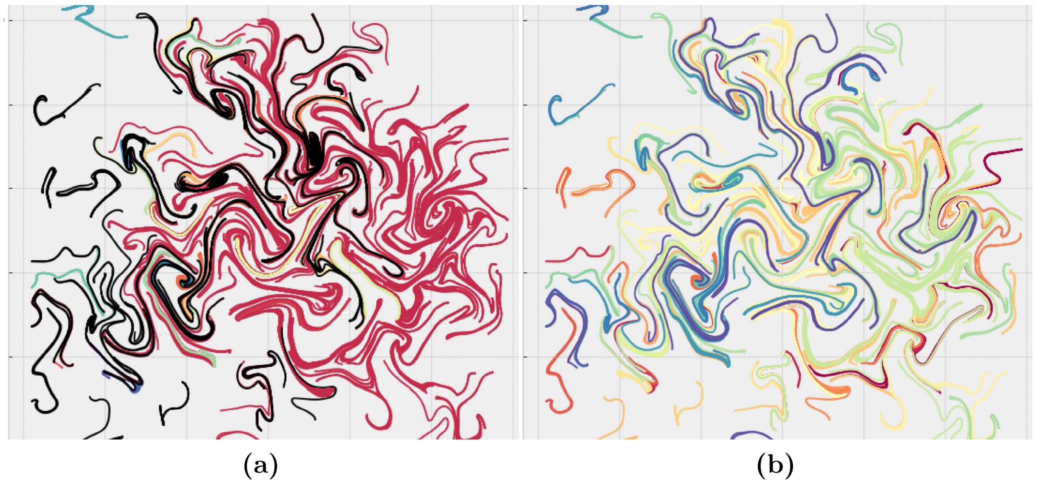


Figure 38: The strainlines have been assigned a group, represented by a color. The clustering was performed using the DBSCAN and Affinity propagation algorithms respectively. It was performed on the reduced representation of the strainlines using K-PCA. The data is plotted in figure 37 and shows no apparent clusters. It was therefore expected that the performance of the clustering algorithms would be poor. The DBSCAN grouped most of the strainlines into a single cluster, while the Affinity propagation grouped the lines seemingly at random. Neither managed to find a good set of LCSs.

Reducing the Dimensionality Using K-PCA and t-SNE

The K-PCA procedure may remove some of the information that the structure of the data holds. To retain as much of the information as possible, the K-PCA was combined with t-SNE to reduce the dimensionality. The same procedure was tested for the double gyre field and is presented in section 3.2.1. The parameters are given in table 1. First, the dimensionality was reduced to 25 using K-PCA, then t-SNE was applied to further reduce it to three dimensions. The new projection of the strainlines as points in three dimensional space is presented in figure 39. This figure was made with a perplexity, $\Lambda = 50$. Comparing this with figure 21 (c), the points are clearly much less densly packed than for the double gyre data. As mentioned in the K-PCA case, this is likely caused by the more random structure of the set of strainlines. However, where the K-PCA procedure returned data which seems to be mostly noise, the t-SNE approach returned a representation where there are some regions of varying density. Seeing that there are some visible grouping in the data, this representation should therefore be a better basis for applying clustering.

Affinity propagation and DBSCAN both gave good results when applied to this data. Figure 40 (a) and (b) plots the LCSs of the system superimposed over the FTLE field for the two algorithms. There are several ridges in the FTLE field where no LCSs have been detected. This could be explained by the fact that up till the clustering, no extra information is given to the program regarding the scale of the system. For the clustering on the strainlines, the Rossby Radius is only used in relation to the pruning that is applied afterwards. The resulting lines

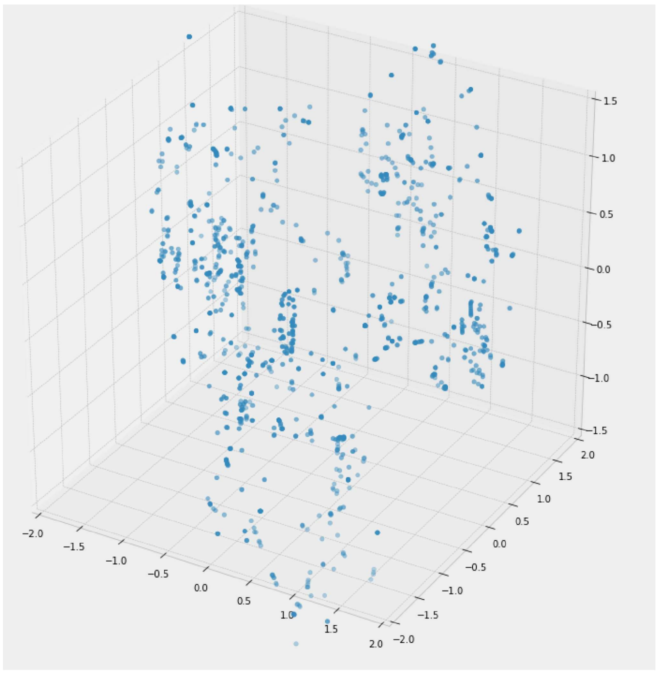


Figure 39: Plot of the strainlines having first been processed by K-PCA, reducing them to points in 25 dimensional space. Then the dimensionality was further reduced using t-SNE. The new representation of the strainlines as points in three dimensional space shows areas of varying density. While the groups are loosely tied together, there are a lot more interesting features in this data than for using only K-PCA to reduce the dimension.

extracted using Affinity propagation could most likely be improved by tuning the input preferences for this exact problem. As for the DBSCAN results, inspecting the clusters that were formed, plotted in figure 40 (c), shows that a majority of strainlines were not assigned to any clusters. The number of lines belonging to clusters are closely related to the number of lines required to form a cluster core, i.e. the number of points in feature space required within a dense area to form a cluster. For the plot in figure 40, the number of points was set to ten. By reducing it to only require five strainlines in an area for a cluster to form, nearly all of the strainlines are assigned to clusters. It thus seems the value for this parameter should be set low in the case of clustering strainlines in ocean data. Consider the case of a strainline with a large $\bar{\lambda}_2$ which finds itself far removed from the rest of the strainlines. If the number of lines needed is set too high, such lines will simply be ignored. Even if there are no groups of lines in an area, there may still be strainlines there qualifying as LCSs. Therefore a low value for this parameter is justified.

Figure 41 plots the clustering and set of LCSs identified setting the minimum number of neighbors to five. There are still ridges in the FTLE field where no LCSs were extracted, but the DBSCAN algorithm seems to capture the LCSs which describe the dynamics of the field. The validity of the uncovered lines are

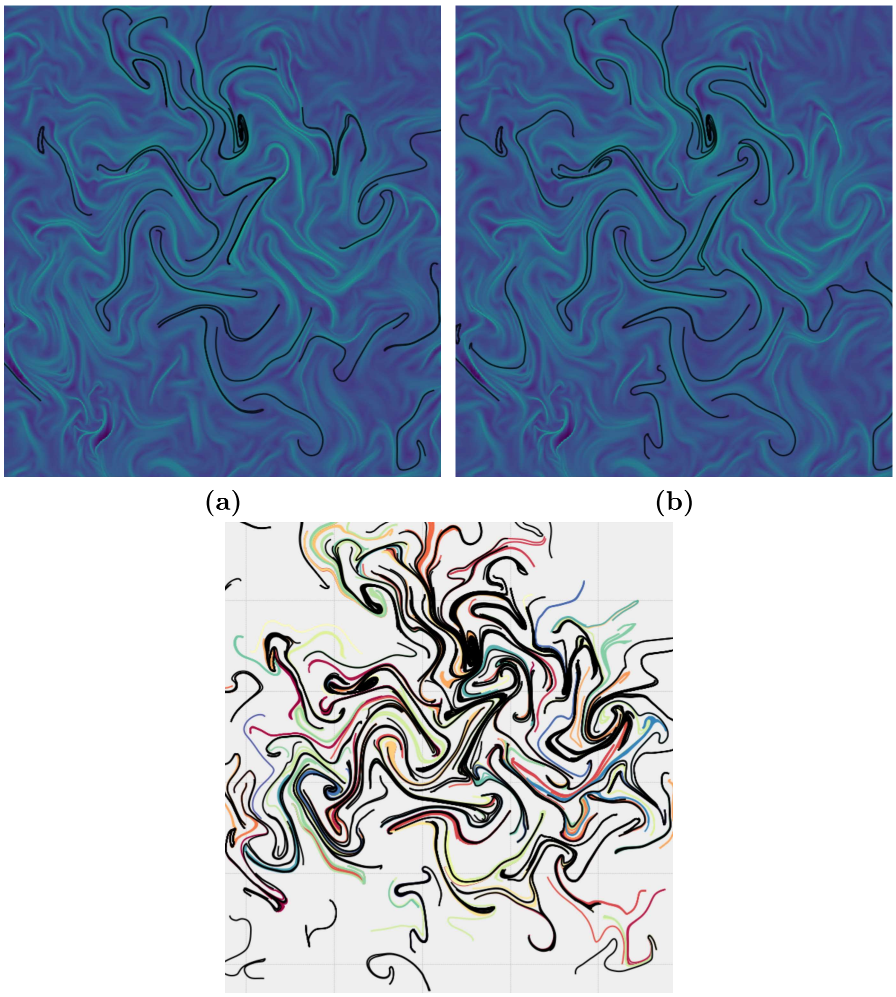


Figure 40: The extracted LCSs using (a) Affinity propagation and (b) DBSCAN to cluster the strainlines. The strainlines were first reduced to points in three dimensional space using K-PCA and t-SNE. The plot in (c) shows the strainlines assigned to various clusters using the DBSCAN algorithm, represented by color. The black lines are not assigned to any cluster. This is caused by a high value for how many neighbors the DBSCAN algorithm considers. In this figure, the parameter was set to ten. Looking at figure 39, the clusters are quite spread out. Therefore, a lower value for the number of neighbors seem warranted.

later tested by advecting an ensemble of particles together with the LCSs to see if the lines govern the motion. The remaining ridges present in the field might simply not contain strainlines with high $\bar{\lambda}_2$ values.

4.3 Clustering the Similarity Matrix

Applying clustering algorithms to the similarity matrix, **S**, gave good results for the double gyre field. Following the procedure presented in section 3, the shortest strainlines found for the modelled ocean data have been removed from the set,

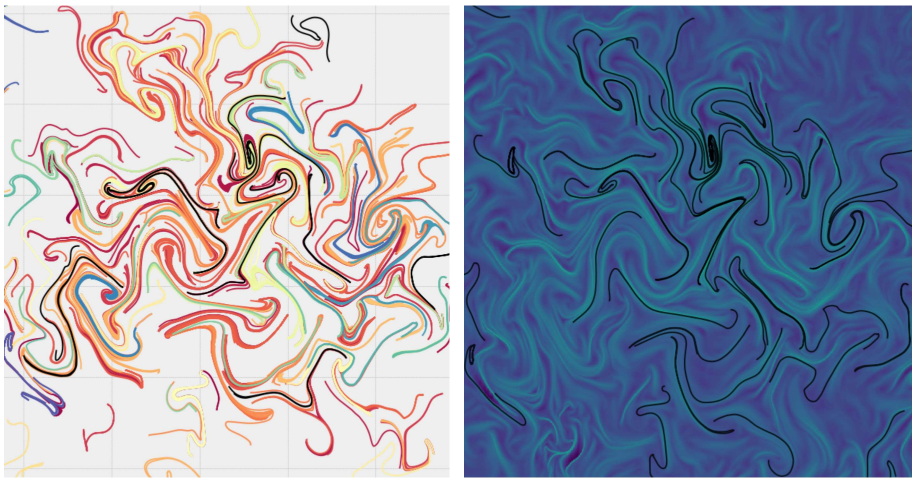


Figure 41: The DBSCAN algorithm was applied to the strainlines after they had been reduced to points in three dimensional space. The dimensional reduction was performed using K-PCA and t-SNE. The plot in (a) shows the strainlines assigned to various clusters. Each cluster is represented by a color. In this figure, the number of neighbors for the DBSCAN algorithm to consider was set to five. Comparing with the case considering a minimum of ten neighbors in figure 40 (c), it is apparent that a lower parameter for the minimum number of neighbors is needed to correctly group most of the strainlines.

reducing it to approximately $N \approx 10^3$ strainlines. This was done to make the computation less demanding, and should not affect the resulting LCSs by a large degree, as short lines will only have a negligible effect on the overall dynamics[7]. The matrix, $\mathbf{S}$, is scaled by the inverse of the Rossby radius, R_r. As before, $\mathbf{S}$ consists of all the pairwise Fréchet distances between the strainlines. The scaling is performed in order to have values in $\mathbf{S}$ which are approximately of the same order independent of the domain.

Dimensional Reduction Using K-PCA and t-SNE.
The plots in figure 42 show the data after being processed by K-PCA and t-SNE. The structure resembles a cloud, with orbs of denser regions loosely tied together. The same configuration for the K-PCA and t-SNE as for the strainlines was used for dimensional reduction on $\mathbf{S}$ as well. The geometry of $\mathbf{S}$ in feature space is not the same as for the strainlines. Each feature describes how closely one line resembles another, while for the strainlines, the features represents coordinates in $\mathbb{R}^2$. There is therefore no other good reason for using the same set of parameters here except for the sake of simplicity. The parameters that were chosen for the K-PCA and t-SNE dimensional reduction were found through testing on $\mathbf{S}$. The tests were performed on both the forward- and backward-time intervals for the modelled data and the double gyre. For each of these sets, the *Cosine*, *Polynomial*, and *RBF* kernels were tested with a varying number of dimensions to first reduce by using K-PCA and different values of perplexity for the t-SNE. The resulting plots of the data reduced to three dimensions were then studied. The values in table 1 were

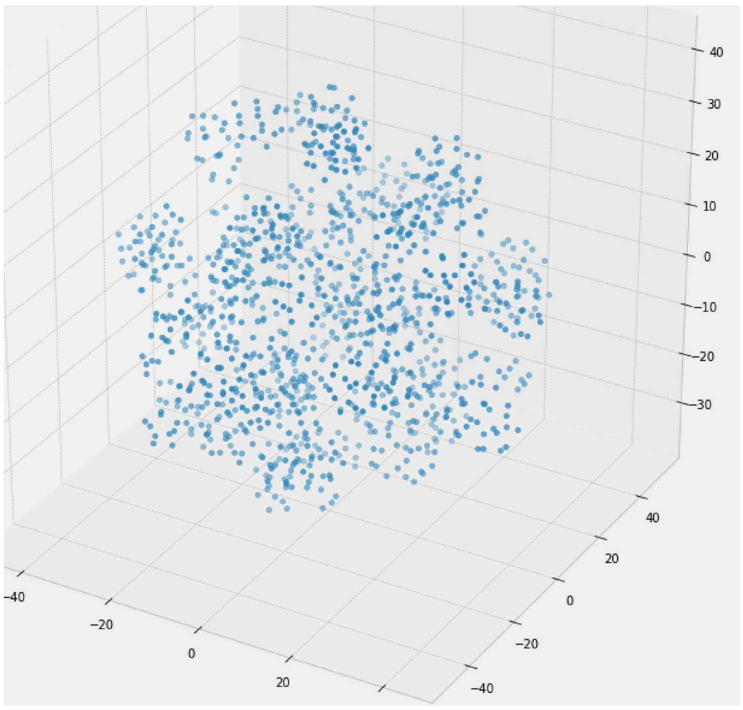


Figure 42: Plot of the similarity matrix, **S**, after it had been reduced to three dimensions using K-PCA and t-SNE. The resulting structure resembles an ensemble of clouds with slightly different sizes and variance.

chosen because they gave visible clusters for both the double gyre and the ocean velocity fields. These parameters could probably be tuned further and a separate set of parameters which fit the strainlines better could also be found. This could therefore be a subject for further work.

The results of the Affinity propagation, Agglomerative-clustering and DBSCAN clustering algorithms applied to the data presented in figure 42 are shown in figure 43 (a) to (f) respectively. Of these plots, the Agglomerative-clustering and DBSCAN gave the best results. The Affinity propagation gave good results for the double gyre data, but it did not deliver for the more chaotic, modelled data. In figure 43 (a) one can see that the algorithm found too few clusters and judged that many of the strainlines colored in orange, located towards the middle of the plot belong to the same cluster. It was harder to separate the performance of the two remaining algorithms. The Agglomerative-clustering was told to find 60 clusters. The number was chosen by inspecting the domain. It was assumed that the number of LCSs should be around the same as the number of intersections found in the grid comprised of horizontal and vertical lines. This assumption seemed reasonable as the lines were spaced at a distance of about $5R_r$ along both axes. This gave a grid which captured most of the details from the set of points satisfying conditions A and B (see figure 35). The set of LCSs plotted in figure 43 (d) covers the ridges in the FTLE field well. In figure 43 (e) and (f) the resulting clusters and LCSs using the DBSCAN algorithm are presented. The algorithm was forced to find at least two strainlines within some length, ϵ, in order to form a cluster. This is why there are some strainlines that are not assigned a cluster, represented by the black lines in figure 43 (e). However, most of the lines have been assigned a cluster, and judging from the figure, it looks as if the algorithm managed to group them fairly well. There are not as many LCSs extracted as for the Agglomerative-clustering. The set of LCSs uncovered with the DBSCAN algorithm forms a neater picture and might have found a smaller set where less important lines have been discarded. One thing to note is that only a few of the approaches locates any LCSs in the bottom left and the top right regions. This is most likely because there are no strongly repelling or attracting LCSs in those regions. The lack of ridges in the FTLE-field also supports this assumption.

Comparing the performance with the case using the double gyre field, the DBSCAN might be the method which gave the most consistent results. The Agglomerative-clustering algorithm was set to cluster the lines into seven groups for the analytic field, while it was 60 groups for the modelled data. The Affinity propagation did a good job on the double gyre, but gave poor results here. More testing to find better parameters for the dimensional reduction could probably improve the performance.

Clustering the Similarity Matrix Directly.
The clustering performed directly on **S** is shown in figure 44. The order in which the algorithms are presented are Affinity propagation, Agglomerative-clustering

and then DBSCAN. All three algorithms fared well with the double gyre data, but Affinity-propagation was a slightly better match with the FTLE-field as seen in figure 29. For the modelled ocean data, the resulting clustering by the Affinity propagation algorithm also gave the best results. From figure 44 (a) one can see that it did well with grouping similar lines together, and that the resulting LCSs match the ridges in the FTLE-field. The Agglomerative-clustering also did a good job at clustering **S**. The Agglomerative algorithm was set to locate 60 clusters, same as for the approach using dimensional reduction on **S**. The resulting LCSs identified are presented in figure 44 (d). Comparing these to the LCSs identified in (b), it is apparent that both of these approaches returned a set of LCSs which matched the FTLE field well. The FTLE field is as mentioned not completely accurate, i.e. a ridge in the FTLE field might not mean that there should be an LCS there[5]. However, the Affinity propagation gave a better set of LCSs for the double gyre field. The set of LCSs uncovered by the Affinity propagation was also slightly smaller, but still covering most of the same ridges. It might therefore seem like Affinity propagation gives more informative clusters resulting in a better picture of the dynamics of the field. The DBSCAN algorithm found the least number of clusters. This could be caused by a poor choice of parameters for the algorithm. The resulting clustering is presented in figure 44 (e) and shows that two of the clusters are very large, with one of them spanning across a huge area. Because of this, a low number of LCSs were identified, which are presented in figure 44 (f).

4.4 Advecting the LCSs

The plots in figure 45 show the set of LCSs uncovered by using Affinity propagation on the similarity matrix, **S**. The red lines represent repelling LCSs, and the blue lines are attracting LCSs. The plots show the LCSs at four different times $t_0, t_0 + 24h, t_0 + 40h, t_0 + 70h$, spanning an interval of 70 hours. An ensemble of particles is plotted together with the lines. By advecting the particles together with the LCSs, one can see that the lines govern the flow of particles. As the lines are advected over a longer interval, they become less reliable. There are some areas where the particles seem to be affected by LCSs which are not included in the plot. This could either be caused by a line not being identified as an LCS or because a new LCS has appeared at a later time $t > t_0$. The latter is expected as the different LCSs discovered may not be present for other intervals[7]. In the double gyre case the period of the system was given by ω (see eq. (52)). The modelled ocean data however, is a-periodic, and the more chaotic nature can make some LCSs short lived, while other can be more enduring.

The agreement between the particle movement and the evolution of the LCSs indicates that the set of LCSs extracted helps explain the dynamics of the system well. As expected, the repelling lines are intrinsically unstable and shrinks with the flow. However, the long-term movement of the particles can be predicted using mostly the attracting lines. As the flow evolves, the particles stick to the blue lines. These are also more numerically stable and are stretched by the flow. By the fourth plot, the field has been advected for 70 hours. The red lines have nearly disappeared, and the particles cling to the blue lines.

4.5 Evaluating the Clustering Methods

Several of the methods used for clustering gave satisfactory results both on the double gyre, and for the modelled data. Many of the plots show LCSs covering the ridges of the FTLE field, but there are slight differences in which lines get identified as LCSs. The reason for this is that both the number of clusters, and which lines gets sorted into certain clusters vary. Because of the large number of figures, not all of the test-cases have been presented in this thesis.

To use the DBSCAN method, the number of strainlines needed to form a cluster, and the distance, ϵ, between the lines in feature space, had to be set prior to clustering[25]. As mentioned earlier, the algorithm was set up to find a large number of clusters, using a relatively large initial value for ϵ_0. Then ϵ and the minimum number of clusters was reduced for each iteration until enough clusters were found using the largest value for ϵ as possible. The thought behind this was that a larger ϵ would force the algorithm to make more general clusters. This method worked reasonably well, and gave satisfactory results for the varying structures of data. The drawback was that the iterative process could be quite slow if the initial value for ϵ was too large. One area where this method could be improved is by using the *sample_weight* input parameter. By feeding the function with the mean eigenvalues, the lines with a large $\bar{\lambda}_2$ will be prioritized as cluster cores. Another point where this could be improved is by using the scale of the input data and the Rossby radius as a basis for setting ϵ. In the paper by Ester et al (1996) another approach for finding an appropriate value for ϵ is presented[25]. The procedure is comprised of calculating a k-nearest-neighbor graph and estimating the percentage of noise in the data. The threshold value for ϵ will then be given by the highest k-distance in the thinnest cluster in the data. Finding a value for ϵ which gives the wanted number of clusters is important in order to achieve an objective identification of LCSs.

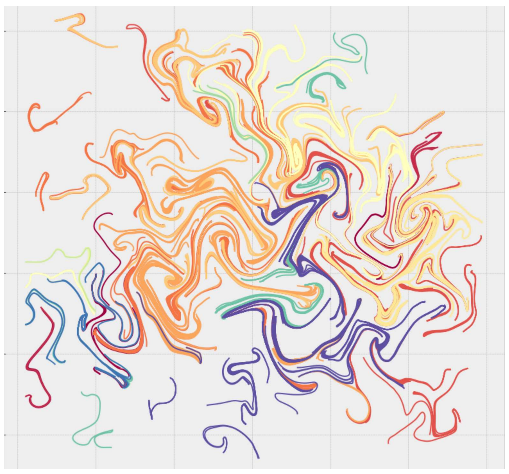
There are a vast number of different configurations for the Agglomerative-clustering which is implemented in *sci-kit learn*. There are three different linkage methods, different distance metrics, and multiple connectivity graphs, each with its own set of configurations. The ones presented in this project used the Average-linkage with a k-nearest-neighbors graph. This graph was a list of size N, where N is the number of strainlines, with either a boolean denoting which of the strainlines are the k nearest neighbors, or the distances to the k nearest neighbors. The rest of the list is set to zero. Because the Agglomerative-clustering needed the number of clusters as input, the other clustering algorithms seemed more promising for handling various flow fields. However, with more optimization, the Agglomerative-clustering could give good results. A starting point for improving the use of this algorithm could be to link the number of clusters to the Rossby radius of the system. The algorithm shows a lot of potential given the results presented in section 3 and 4.

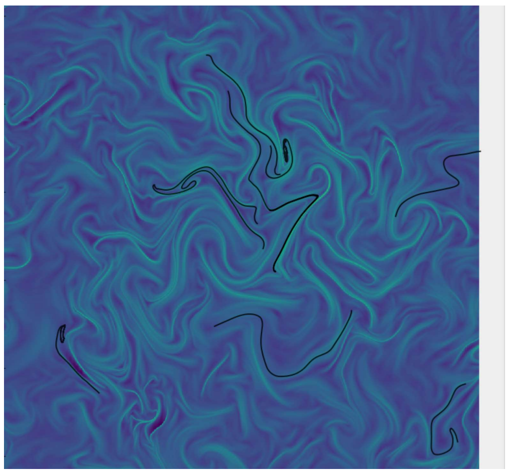
The method which performed best was the Affinity propagation applied to the similarity matrix **S**. This approach gave an ensemble of LCSs that matched the FTLE-field well. The Affinity propagation was initialized with a damping factor between 0.5 and 0.75, and the preferences were set to an array consisting of the mean eigenvalues. The damping factor reduces the tendency for solutions to oscillate when performing the propagation[26]. The eigenvalues were centered and scaled to unit variance using the function *StandardScaler* from the *sci-kit learn* library[24]. The scaled eigenvalues were then mapped through an inverse exponential function so that the the largest eigenvalues would result in a low preference. The points which have preferences close to zero would have the highest probability to be chosen as an exemplar. Other functions could probably have been used, but the exponential function produced good results.

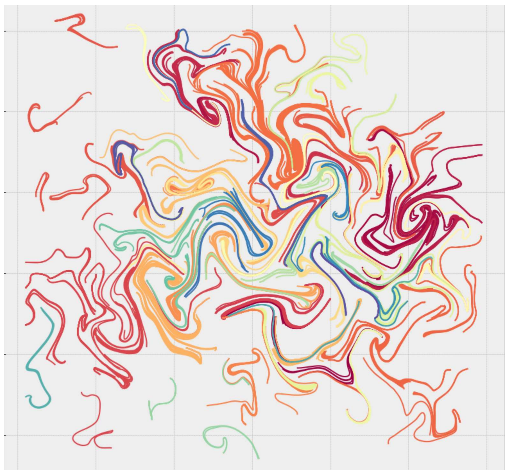
Clustering on the strainlines having only used K-PCA to reduce the dimensionality gave poor results. It is likely that the algorithm did not manage to retain enough of the information in the data. Different parameters were tested, but none seemed to generalize well for both sets of data. The structures that were captured in the reduction of the double gyre data were absent in the ocean data. The addition of the t-SNE as a final step in the dimensional reduction helped to some degree, but the methods gave inferior results compared with clustering directly on the high dimensional similarity matrix using Affinity propagation. Other algorithms for dimensional reduction might retain more of the information hidden within the data, but for the methods tested in this paper, too much of the original structure was lost. The t-SNE is still an excellent tool for visualizing the data and checking if there are clusters present, but the clustering algorithms should be applied to the high dimensional representation.

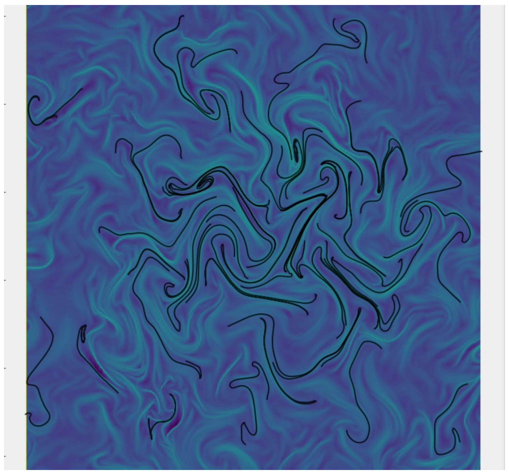
The t-SNE method has a non-convex cost function which is given by eq. (44). This non-convexity can result in the gradient descent not finding the global minimum. There are several parameters than can be used to optimize the solution found by the gradient descent. Hinton and van der Maaten (2008) argues that the quality of the optima does not vary much with different parameters[11]. By taking a look at figure 21 one can see that this is not necessarily the case. Van der Maaten (2008) also state that the algorithm may exhibit varying performance when applied to data with a high intrinsic dimension. This is caused by the assumption of local linear relation between neighboring pairs.

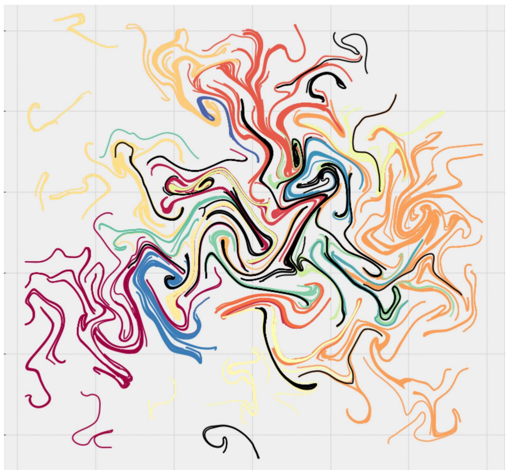
In section 4.4 the effect the LCSs has on surrounding particles is illustrated. The red lines are repelling LCSs, and the blue lines are attracting LCSs. The movement of the particles agrees with the extracted lines.


(a)


(b)


(c)


(d)


(e)

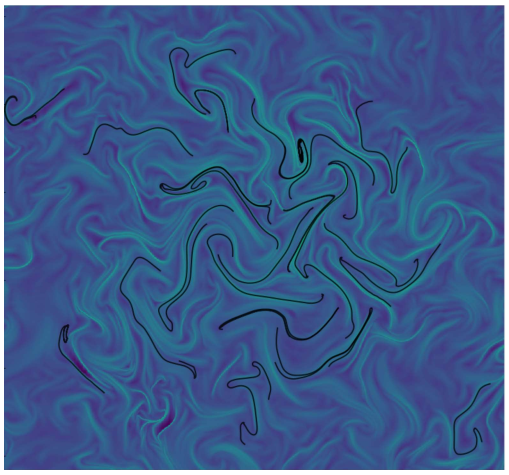
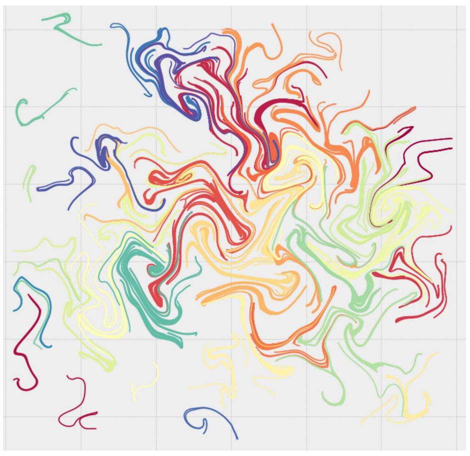
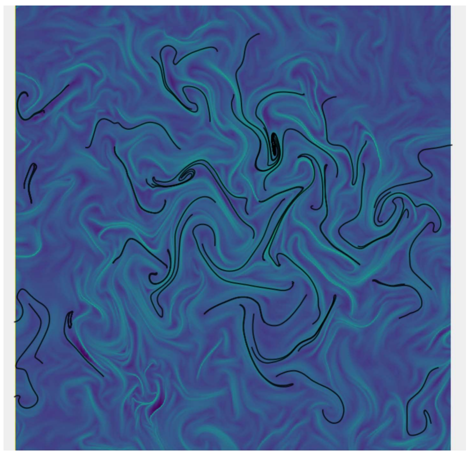
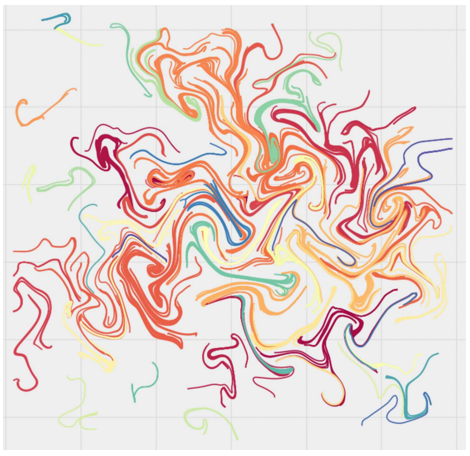
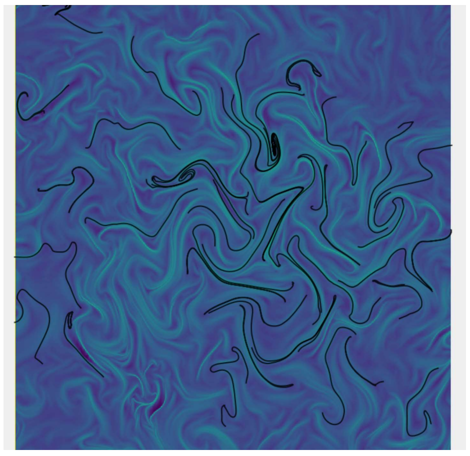

(f)

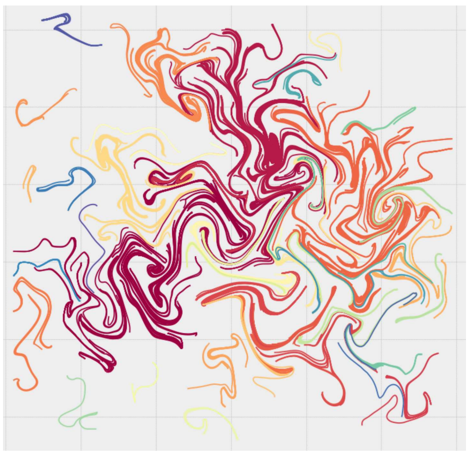
Figure 43: The plots in (a) to (f) presents the clusters and the resulting LCSs for the Affinity propagation, Agglomerative-clustering, and the DBSCAN algorithms respectively. The clustering was performed on the similarity matrix, **S**. Prior to clustering, **S** had been reduced to three dimensions using K-PCA and t-SNE. The resulting data is presented in figure 42.


(a)


(b)


(c)


(d)


(e)

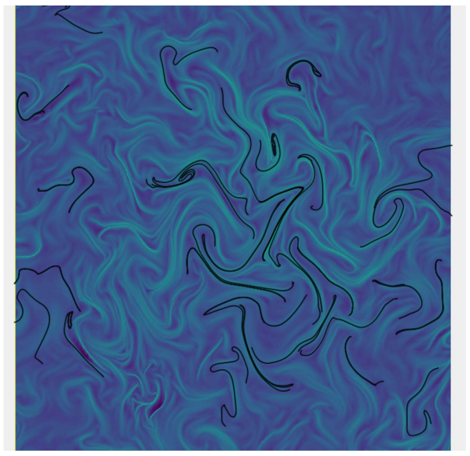

(f)

Figure 44: Plots of the clusters and corresponding sets of LCSs identified when performing the clustering directly on the similarity matrix, **S**. Plot (a) and (b) are the results using Affinity propagation, (c) and (d) are from the Agglomerative clustering, (e) and (f) are from the DBSCAN algorithm. Both the Agglomerative-clustering and Affinity propagation gave satisfactory results. There are some slight variations in the set of LCSs. However, the Affinity propagation performed better than the Agglomerative-clustering when it was used on the double gyre field.

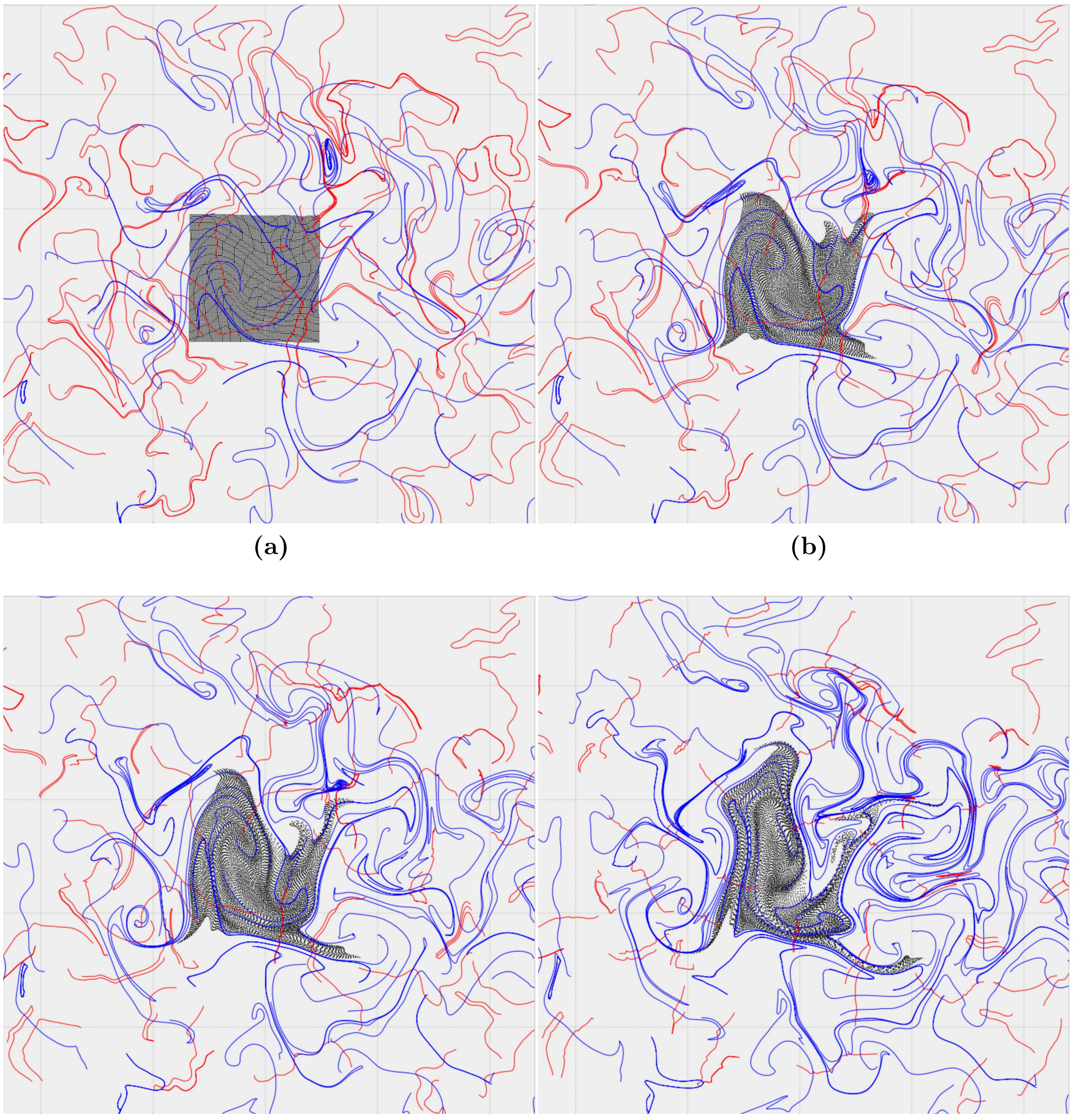


Figure 45: The LCSs of the system identified using Affinity propagation directly on the similarity matrix, **S**. The blue lines represents attracting LCSs, and the red lines repelling LCSs. The LCSs in (a) are advected to $t = t_0 + 24$ h, $t = t_0 + 40$ h and $t = t_0 + 70$ h in (b), (c) and (d) respectively. The initial black square in plot (a) represents the initial position of the ensemble of particles at $t = t_0$. As time goes on, the square gets deformed by the LCSs. Because the repelling lines shrink and the attracting lines stretch, the blue lines eventually take over the plot if advected long enough. From the figure one can see that the trajectories are governed by the nearby LCSs.

5 Conclusion

Using clustering and dimensional reduction as an alternative method for extracting LCSs out of sets of strainlines shows promising results. Out of the DBSCAN, Agglomerative-clustering and Affinity-propagation, it was the latter which managed to identify the most general set of LCSs. The aim of this project was to find a method for identifying LCSs which could be a more robust alternative to the procedure described in Farazmand and Haller (2012), which was largely based on using a seemingly arbitrary grid[7]. The clustering presented in this thesis eliminates the need for using such a grid, but the algorithms introduced a whole new set of parameters. Many of these parameters had to be tuned in order to achieve satisfactory results. However, most of these parameters could either be found by iteration (see section 3.2) or by for example using the mean eigenvalues as weights in the clustering, as noted in section 3.2 and 4.5. The dimensional reduction procedure also showed potential, but there was some loss of information in the reduced representation, which caused important strainlines to be left out of the final set of LCSs. This could possibly be caused by the t-SNE approach converging to sub-optimal solutions[11]. A solution to this, which has not been tested here, could be to perform the reduction many times, using different input parameters, and then to use the solution which has the lowest minimum for the cost function (see section 2.6.3).

The set of LCSs identified for the double gyre flow field using clustering, were a good match with the LCS identified in Farazmand and Haller (2012). However, the implementation used for this thesis did not produce the single strainline presented there, but rather several disconnected pieces which formed approximately the same line. This is most likely caused by numerical errors in the integration or some differences in the implementation.

To check if the LCSs uncovered in the modelled ocean data would describe the dynamics of the flow, the LCSs were advected together with an ensemble of particles. Then the LCSs effect on the nearby trajectories were studied. The particle trajectories were shown to be in good accordance with the identified set of LCSs. Especially for shorter intervals the dynamics of the field seem well explained by the LCSs. After an interval of 70 hours, the repelling lines were reduced to a negligible length and most of the particles were found hugging nearby attracting lines.

Even though the results were promising, there is still work remaining with regards to tuning the clustering and testing other approaches to dimensional reduction. An example is the input parameters for the DBSCAN algorithm which could be calculated using the procedure suggested by Ester et al (1996)[25]. Further testing of the usability of this approach on other sets of data is also needed in order to decide if clustering can replace the previous methods for extracting LCSs. A good way to test the validity of the LCSs uncovered by this method would be to apply it to the Deepwater Horizon case to test its reliability in predicting the spread of the oil spill.

References

[1] Thomas Peacock and George Haller. Lagrangian coherent structures: The hidden skeleton of fluid flows. *Physics Today*, 66(2):41–47, February 2013.

[2] María J. Olascoaga and George Haller. Forecasting sudden changes in environmental pollution patterns. *Proceedings of the National Academy of Sciences*, 109(13), March 2012.

[3] National Public Radio for bp cleanup, 2013 meant 4.6 million pounds of oily gunk. `https://www.npr.org/2013/12/21/255843362/for-bp-cleanup-2013-meant-4-6-million-pounds-of-gulf-coast-oil`. Accessed: 2018-06-11.

[4] Tampa Bay Times oil from bp spill pushed onto shelf off tampa bay by underwater currents. `http://www.tampabay.com/news/environment/water/oil-from-bp-spill-was-pushed-onto-shelf-off-tampa-bay-by-underwater/2137406`. Accessed: 2018-06-11.

[5] George Haller. Lagrangian coherent structures. *Annual Review of Fluid Mechanics*, 47(1):137–162, 13 2015.

[6] George Haller. A variational theory of hyperbolic lagrangian coherent structures. *Physica D: Nonlinear Phenomena*, 240(7):574 – 598, 2011.

[7] Mohammad Farazmand and George Haller. Computing lagrangian coherent structures from their variational theory. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(1):013128, 2012.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[9] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall series in artificial intelligence. Pearson Education, Boston, 3rd edition. edition, 2016.

[10] A. K. Jain, R. P. W. Duin, and Jianchang Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, Jan 2000.

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[12] Ernst Hairer, Syvert P Nørsett, and Gerhard Wanner. *Solving ordinary differential equations I - Nonstiff problems.* Springer, 1993.

[13] Steven H Strogatz. *Nonlinear dynamics and chaos : with applications to physics, biology, chemistry, and engineering.* Studies in nonlinearity. Addison-Wesley, Reading, Mass, 1994.

[14] Shawn C. Shadden, Francois Lekien, and Jerrold E. Marsden. Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3):271–304, 2005.

[15] Travis E. Oliphant. *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015.

[16] Ko-Foa Tchon, Julien Dompierre, Marie-Gabrielle Vallet, François Guibault, and Ricardo Camarero. Two-dimensional metric tensor visualization using pseudo-meshes. *Engineering with Computers*, 22(2):121–131, September 2006.

[17] Charles Bouveyron and Camille Brunet-Saumard. Model-based clustering of high-dimensional data: A review. *Computational Statistics & Data Analysis*, 71:52 – 78, 2014.

[18] S. Y. (Sun Yuan) Kung. *Kernel methods and machine learning*. Cambridge University Press, 2014.

[19] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1327:583–588, 1997.

[20] Geoffrey E Hinton and Sam T. Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 857–864. MIT Press, 2003.

[21] DM Hawkins. The problem of overfitting. *Journal Of Chemical Information And Computer Sciences*, 44(1):1–12, 2004.

[22] Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 05(01n02):75–91, 1995.

[23] Oded Maimon. *Data Mining and Knowledge Discovery Handbook*. Springer series in solid-state sciences Magnetic bubble technology. Springer, 2nd ed. edition, 2010.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[26] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *Science (New York, N.Y.)*, 315(5814), February 2007.

[27] Thomas Eiter and Heikki Mannila. Computing discrete fréchet distance. Technical report, 1994.

[28] D Chelton, R Deszoeke, M Schlax, K El Naggar, and N Siwertz. Geographical variability of the first baroclinic rossby radius of deformation. *Journal of Physical Oceanography*, 28(3):433–460, March 1998.

[29] Frank M White. *Fluid mechanics*. McGraw-Hill series in mechanical engineering. McGraw-Hill, New York, 7th ed. edition, 2011.

[30] MET Norway. Data from The Meteorological Institute of Norway. `https://thredds.met.no/thredds/catalog.html`. [Online; accessed 2018-06-11].