THOMAS WILDE

FLOW MAP PROCESSING

DISSERTATION

Flow Map Processing

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik der
Otto-von-Guericke Universität Magdeburg

von Thomas Wilde geb. Seidl
geb. am 18. Februar 1984 in Annaberg-Buchholz

Gutachter

Prof. Dr. Holger Theisel
Prof. Dr. Gerik Scheuermann
Prof. Dr. Markus Hadwiger

Magdeburg, den 26. März 2021

Thomas Wilde: *Flow Map Processing,*
Dissertation, 26. März 2021

ABSTRACT

This thesis presents new techniques to the field of Flow Visualization that are based on the flow map.

Water flowing down a river, the aerodynamics of a car, the air we breathe – flows, i.e., liquids or gases in motion, play an essential role in our everyday life. Understanding flows is essential in various fields. Therefore, they are of substantial research interest. The discipline of Flow Visualization aims for a meaningful visualization of information contained in flows. The techniques we can apply heavily depend on how the data is represented. One way that has gained research interest in the last years is the *flow map*. Flow maps are Lagrangian flow representations, i.e., they describe the movement of massless particles in the flow. They have particular properties that make them complex data structures. This thesis shows different ways to work with flow maps that can be classified into two parts.

In the first part of this thesis, we consider the flow map structure and directly address its data. Based on a sound introduction, we develop a concept to modify flow maps. Each modification happens in a local area defined in space-time and entails a global adaption of different flow map parts. We present efficient techniques that handle this process while maintaining inherent flow map properties. Furthermore, we present *drift fields*, a novel technique to present flow data from the Lagrangian perspective. Drift fields implement advantages of flow maps but are easier to handle. We show their relation to flow maps and how to compute them.

The second part of this thesis is related to flow features that we derive from the flow map. The extraction of ridge structures from finite-time Lyapunov exponent fields is a widely used approach for the computation of Lagrangian coherent structures. We present an approach that uses intermediate time steps and extracts highly resolved ridge geometries. Furthermore, we consider the phenomenon of recirculation in unsteady 3D flows. We show that particles with recirculating behavior form 2-manifolds and present *recirculation surfaces* – the first flow feature that incorporates the full 5D flow map for 3D unsteady flows.

ZUSAMMENFASSUNG

Diese Arbeit stellt neue Techniken im Bereich der Strömungsvisualisierung vor, die auf der Flow Map basieren.

Wasser, das einen Fluss hinunterfließt, die Aerodynamik eines Autos, die Luft, die wir atmen - Strömungen, d.h. Flüssigkeiten oder Gase in Bewegung, spielen eine wesentliche Rolle in unserem täglichen Leben. Das Verständnis von Strömungen ist in unterschiedlichen Bereichen essenziell. Strömungen sind daher von erheblichem Forschungsinteresse. Das Forschungsfeld der Strömungsvisualisierung zielt auf eine sinnvolle Visualisierung der in Strömungen enthaltenen Informationen ab. Die anwendbaren Techniken, hängen stark davon ab, wie Strömungsdaten dargestellt werden. Eine Variante, die in den letzten Jahren an Forschungsinteresse gewonnen hat, sind *Flow Maps*. Flow Maps sind Lagrangesche Darstellungen, d.h. sie beschreiben die Bewegung von masselosen Teilchen in der Strömung. Sie haben besondere Eigenschaften, die sie zu komplexen Datenstrukturen machen. In dieser Arbeit werden verschiedene Möglichkeiten aufgezeigt, mit Flow Maps zu arbeiten.

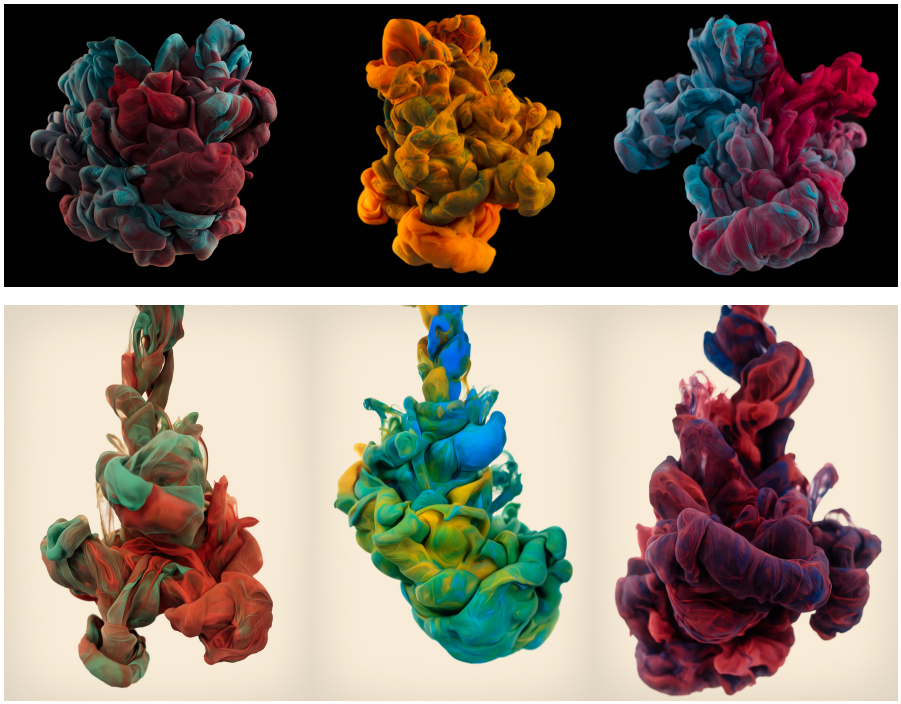
Im ersten Teil dieser Arbeit betrachten wir die Flow Map Struktur und befassen uns direkt mit den enthaltenen Daten. Basierend auf einer fundierten Einführung entwickeln wir ein Konzept zur Modifikation von Flow Maps. Jede Modifikation geschieht in einem lokalen Bereich, der in der Raum-Zeit definiert ist und zieht eine globale Anpassung verschiedener Flow Map Teile nach sich. Wir stellen effiziente Techniken vor, die diesen Prozess unter Beibehaltung der inhärenten Flow Map Eigenschaften ermöglichen. Darüber hinaus stellen wir *Drift Felder* vor, eine neuartige Technik zur Darstellung von Strömungsdaten aus der Lagrangeschen Perspektive. Drift Felder implementieren einige Vorteile von Flow Maps, sind aber einfacher zu verwalten. Wir zeigen ihre Beziehung zu Flow Maps und wie man sie berechnet.

Der zweite Teil dieser Arbeit befasst sich mit Strömungsmerkmalen, die wir aus der Flow Map ableiten. Die Extraktion von Ridge Strukturen aus FTLE (finite-time Lyapunov exponent) Feldern ist ein weit verbreiteter Ansatz zur Berechnung von Lagrangesche-kohärenten Strukturen. Wir stellen einen Ansatz vor, der mehrere Zeitschritte einbezieht und extrahieren hochaufgelöste Ridge Geometrien. Weiterhin betrachten wir das Phänomen der Rezirkulation in zeitabhängigen 3D-Strömungen. Wir zeigen, dass Partikel mit rezirkulierendem Verhalten 2-Mannigfaltigkeiten bilden und präsentieren *recirculation surfaces* - das erste Strömungsmerkmal, das die vollständige 5D-Flow Map für zeitabhängige 3D-Strömungen verwendet.

CONTENTS

1

INTRODUCTION



BLACKGROUND & THE BLACK TRAP IN MUNICH. Beautiful patterns emerge when ink or varnish drops into water. (Images courtesy Alberto Seveso [143])

'Blackground' and *'The Black Trap in Munich'* are the titles of two photo series by the artist and photographer *Alberto Seveso*. He made them by dropping varnish and ink into a water tank. With the proper equipment, light conditions, view angle, patience, and experience, these impressive images were created. They are stunning examples of the complexity and beauty of liquids or gases that are in motion.

Especially when two different media interact with each other, we become aware of flows. Wind blowing snow or dust over the ground, smoke rising from a blown-out candle, pouring milk into coffee, or the perfume smell of a person walking by are more examples of situations where we realize flows. Even though we do not recognize it most of the time, flows impact our everyday life. They can be found almost everywhere and in all scales, from microscopic to global size. Cytoplasmic streams influence the transport of nutritions in cells. The blood flow in our body and the air we breathe in and out keep us alive. The wind passing our skin cools us down. The flying skills of birds depend on their ability to

control the air around their wings. Vertical airflow next to doors keeps the heat inside buildings. Managing the drainage of rain avoids damage to buildings and streets. The airflow around a car influences its driving stability and fuel consumption.

Scientists put considerable effort into global climate models that describe air movement in the atmosphere and water flows in the oceans. They predict the weather for the next days or the pollution transport after a volcanic eruption or an oil spill in the ocean. Biology, medicine, engineering, meteorology, oceanography – flows are essential in many fields. Researchers try to understand, explain, and control them by searching for models representing flow phenomena and interactions.

Flow Visualization, as a subfield of Scientific Visualization, is an essential tool during this process. It has gained importance during the last years and is an active research field. Flow Visualization aims to find and visualize relevant information contained in flows. New algorithms and methods for analyzing and describing flow data get developed continuously. Flow data grows in size and complexity, especially when dealing with time-dependent data sets. With the increasing computing power and storage capacity of modern computers, researchers can handle this data.

A large number of data sets stem from simulations and contain a variety of information. Examples for such data range from scalar values like temperature, pressure, or salinity to more complex data described by tensors, like stress or diffusion. The data we consider in this thesis is the velocity data. Velocity data expresses flow direction and flow speed. Typically, velocity data is represented by time-dependent vector fields. A common way to investigate flows is to analyze these vector fields directly. Direct techniques evaluate the vector field usually pointwise at single locations – they use the *Eulerian* point of view. Another group of approaches uses the *Lagrangian* point of view. For this, one observes and analyzes the behavior of particles advected by the vector field. Massless particles are virtually placed in the vector field. A numerical integration determines the trajectory a particle takes – its pathline. This thesis's primary focus is a powerful tool to describe such pathlines – the *flow map*.

The flow map directly encodes particle trajectories. It is a complex data structure in different ways. To obtain a flow map, one has to perform a significant number of numerical integrations, which is computationally expensive. Flow maps are high-dimensional, which leads to a considerable amount of memory to store them. They have a complex and highly connected inner structure. Furthermore, they can show *'strange'* behavior for long integration times, e.g., they develop huge gradients. To handle flow maps is challenging but rewarding. Due to increased computing power, flow maps have gained research interest over the last years. Several visualization techniques utilize them already. Though,

most of these techniques use only a small part of it. In this thesis, we aim to go a step further and focus on the *full flow map*.

1.1 CONTRIBUTIONS

We separate the contributions of this thesis into two parts. Both of them deal in different ways with the flow map. The first part is concerned with the inherent structure of the flow map itself, a particular representation form, and its processing. In the second part, we present two flow features relying on large parts, respectively, the full flow map.

Flow Map Processing

We give compact theoretical background about flow maps, explain what they are, and their relation to vector fields. Based on a solid mathematical foundation, we characterize the general intrinsic properties. For vector fields, well-established approaches exist to model, construct, deform, simplify, and compress them. In comparison, there is a lack of such processing techniques for flow maps. The main reason for this is their complex inner structure, making it hard to modify them. To tackle this problem, we present a technique for flow map modification. Based on a space-time deformation, we locally change flow map entries. We propagate these local adaptions to the overall global structure. This way, we explicitly *'repair'* the flow map and keep the inherent properties.

After that, we introduce *drift fields* – a new form of Lagrangian flow representation situated between velocity fields and flow maps. We give a concrete theoretical definition and present particular properties. Moreover, we show how to modify drift fields without losing their characteristics and present an efficient way to convert them to flow maps. We see the presented methods as the first steps towards further flow map processing techniques.

Flow Features from Flow Maps

We present two approaches to flow feature extraction that relies on large parts of the flow map. The first one deals with detecting *Lagrangian coherent structures* (LCS) based on *ridge* detection in *finite-time Lyapunov exponent* (FTLE) fields. LCS are robust structures that describe global flow properties. Uncovering these flow structures is a promising way to gain a simplified understanding of global flow behavior. A widely used tool to compute LCS is ridge detection in FTLE fields. With ongoing integration time, FTLE ridges tend to become sharp and thin. It is challenging to detect them once they have gained this shape. Then again, ridges emerge slowly over time and are easier to detect for short integration times. Based on this insight, we present an

approach that weakens the *finite-time* in the FTLE computation. By considering multiple time steps during ridge detection, we utilize a more significant part of the flow map than standard techniques. Furthermore, we suggest a statistical consideration of extracted ridge geometry for flow characterization. Though we vary the integration time for this technique, we are still bound to a fixed starting time and use only a part of the flow map.

The second approach we present goes to the final step we want to achieve in this thesis. We introduce an entirely new flow feature that takes the full flow map into account – *recirculation surfaces*. Recirculation is a phenomenon that influences many effects in 3D time-dependent flows. An intuitive understanding of the term *recirculation* refers to a moving particle that reaches its starting point again. Different communities observed this phenomenon already, ranging from biology, chemistry, and physics to Flow Visualization. Though, there seems to be no unique standard description of the concept of recirculation. We address this problem by giving a concrete definition. Based on this, we define recirculation surfaces as 2-manifolds embedded in the 5D space of the flow map. They represent the first comprehensive flow feature described in the definition space of the full flow map. We study their properties, provide an algorithm for their computation, and demonstrate their extraction on different data sets. Furthermore, we show that the finding of isolated closed orbits in steady vector fields occurs as a particular case of recirculation surfaces.

1.2 THESIS STRUCTURE

This thesis is divided into four parts.

Part I gives the essential background information that we see as a prerequisite to understanding the presented work.

- Chapter 2 gives a short introduction to the field of Flow Visualization. It covers concepts used throughout the thesis like scalar fields, vector fields, and integral curves. We introduce the flow map, give a theoretical foundation, discuss its properties and its relation to velocity fields. Furthermore, we give a brief overview of the FTLE.
- Chapter 3 is dedicated to a data set – the Double Gyre. It is a synthetic velocity field that we use as a benchmark for all presented approaches. It plays a notable role in our research and took a decisive influence on this thesis.

Part II is related to the flow map itself and contains our contributions to flow map processing.

- Chapter 4 presents a flow map modification approach by space-time deformation.
- Chapter 5 introduces drift fields as a new form of Lagrangian flow representation. We give a formal definition, discuss their properties, and their relation to velocity fields and flow maps. We show how to convert them and illustrate their deformation.

Part III presents our contributions to flow feature extraction utilizing the flow map.

- Chapter 6 shows an approach for the computation of FTLE ridge lines for long integration times by an adaptive grid refinement.
- Chapter 7 defines the phenomenon of recirculation and presents recirculation surfaces.

Part IV finally concludes this thesis by giving a summary in Chapter 8 and showing possible future research in Chapter 9.

1.3 LIST OF PUBLICATIONS

The following articles have been published in peer-reviewed international conferences as a result for the research of this thesis:

- T. Wilde, C. Rössl, H. Theisel
  FTLE Ridge Lines for Long Integration Times
  IEEE Scientific Visualization Conference, 2018, Short Paper

- T. Wilde, C. Rössl, H. Theisel
  Recirculation Surfaces for Flow Visualization
  IEEE Transactions on Visualization & Computer Graphics, 2019

- S. Wolligandt, T. Wilde, C. Rössl, H. Theisel
  Static Visualization of Unsteady Flows by Flow Steadification
  Proceedings of Vision, Modeling, and Visualization, 2020

- S. Wolligandt, J. Zimmermann, T. Wilde, M. Motejat, H. Theisel
  Lagrangian Q-criterion and Transport of Salt and Temperature
  IEEE Scientific Visualization Contest, 2020

- T. Wilde, C. Rössl, H. Theisel
  Flow Map Processing by Space-Time Deformation
  Advances in Visual Computing – International Symposium, 2020

- S. Wolligandt, T. Wilde, C. Rössl, H. Theisel
  A Modified Double Gyre with Ground Truth Hyperbolic Trajectories for Flow Visualization
  Computer Graphics Forum, 2020

- T. Wilde, S. Wolligandt, C. Rössl, H. Theisel
  Drift Fields for Flow Map Processing
  submitted to EuroVis Conference 2021

1.4 NOTATION

The following table lists the mathematical notation we use in this thesis. We introduce more specialized notations in the respective section, if necessary.

General Notation

x, t, τ	Scalar value
$f(...)$	Scalar-valued function
$\mathbf{x}$	Vector in nD-space (column-wise)
x_i	Component of a vector
$\mathbf{f}(...)$	Vector-valued function
D, T	Spatial or temporal domain
$\mathbf{M}$	Matrix
$\mathbf{x}^{\mathrm{T}}, \mathbf{M}^{\mathrm{T}}$	Tranpose of a vector or matrix
$\mathbf{M}^{-1}$	Inverse of a matrix
$\mathbf{x}^{\mathrm{T}}\mathbf{x}$	Dot product in matrix notation
$\mathbf{M} \cdot \mathbf{x}$	Matrix-vector product
$\mathbf{M} \cdot \mathbf{N}$	Matrix-matrix product
$\det(\mathbf{M})$	Determinant of a matrix
$\mathrm{d}f \,/\, \mathrm{d}t$	Total derivative of a (multivariate) function
$\partial f / \partial t$	Partial derivative of a multivariate function
$\mathcal{M}$	Set
$t_0, \mathbf{x}_1, T_\tau$	Particular instance of an element
$\widetilde{\mathbf{v}}, \widetilde{\phi}$	Modified or extended version of an element

Common Symbols

$\mathbf{0}$	Zero vector (column-wise)
ϵ	Small quantity
$\mathbf{I}$	Identity matrix
t	Time
τ	Integration time
$\mathbf{v}(\mathbf{x})$	Steady (time-independent) vector field
$\mathbf{v}(\mathbf{x}, t)$	Unsteady (time-dependent) vector field
$\phi(\mathbf{x}, t, \tau) = \phi_t^\tau(\mathbf{x})$	Flow map
∇	Nabla operator
$\mathbf{J} = (\nabla \mathbf{v})^{\mathrm{T}}$	Jacobian matrix (first-order partial derivatives)
λ	Eigenvalue of a matrix
$\mathbf{e}$	Eigenvector of a matrix

Part I

BACKGROUND

2

INTRODUCTION TO FLOW VISUALIZATION

"The Purpose of Computing Is Insight, Not Numbers" is the motto of the book *Numerical Methods for Scientists and Engineers* by Richard Hamming, published in 1962 [69]. The author himself states out that many people misinterpret it, especially when dealing with computers. In 1973 he clarified that this statement not only refers to the results achieved with increasing computing power. It covers the whole process of encountering a problem. Defining the problem, selecting appropriate formulas to describe it, implementing the algorithms, extracting the results, choosing the methods to present them, and many more. One can take each of these steps in different ways, and every path will lead to further insights into the whole process. When we interpret the quote this way, it perfectly fits the research field of *Scientific Visualization*.

Scientific Visualization is concerned with techniques to extract and visualize knowledge from scientific data [10]. It emerged as a discipline in the 1980s and became of interest with increasing computing power. McCormick et al. [104] shaped the term in a publication in 1987. In their work, they listed short and long-term goals and pointed out principal conclusions and recommendations. The report also specified the term *scientific data* as data that results from simulations or computations. Scientific Visualization combines aspects from several different areas, like computer graphics, image processing, signal processing, and user-interface methodology. During the following years, many activities all over the globe advanced Scientific Visualization research in different fields like physics, meteorology, biology, medicine, chemistry, and engineering [17]. Specialized workgroups, with the knowledge and primarily powerful computational architecture, appeared and visualized scientific data. In recent years the nature of Scientific Visualization has changed again. The necessary computational power is ubiquitous. Easy-to-use software, online tools, cloud computing, and professional software packages filled with complex techniques for high-dimensional data are readily available not only for experts. Even though Richard Hamming's quote is still valid today, we have to see the process of scientific visualization as a whole and make thoughtful decisions at each step. Otherwise, *"we run into the risk of using the computer to make clever rubbish" (Helen Wright)* [181].

We assign this work to the field of *Flow Visualization*, which is a part of Scientific Visualization. The following sections give a brief introduction to this research field and provide the necessary background for this thesis. We will cover the essential data types – scalar fields, vector

fields, and flow maps. We give a formal definition for each and introduce established mathematical concepts and visualization techniques. We explain what pathlines, streamlines, streaklines, and timelines are and how they connect vector fields and flow maps. In the latter part of this chapter, we introduce the finite-time Lyapunov exponent. We use it on different occasions throughout the thesis. Furthermore, we explain how we handle most data types – discretized on a regular sampling grid. However, we only focus on the concepts relevant to this thesis. For additional material, we refer to the literature and the sources cited throughout the work.

2.1 SCALAR FIELDS

This section introduces scalar fields, one of the essential data formats used in Scientific Visualization. We begin with a formal definition and discuss basic visualization techniques afterward.

2.1.1 *Definition*

definition scalar field

We define the scalar field $f(\mathbf{x})$ as a scalar-valued function over the spatial domain D that assigns a scalar value $s \in \mathbb{R}$ to each position $\mathbf{x} = (x_1, x_2, ..., x_n)^{\mathrm{T}}$

$$f : D \to \mathbb{R}\,,$$
$$f(\mathbf{x}) = s\,,$$

with $\mathbf{x} \in D$. The domain D is a subset of the vector space $\mathbb{R}^n$, i.e., $D \subseteq \mathbb{R}^n$. Usually, n has the value of $1, 2$, or 3. We may refer to the spatial dimensions as x, (x, y) or (x, y, z) in these cases. In this thesis, most often, $n = 2$. In many scientific applications, time plays an important role. For example, in simulations, a state's change is computed for a particular period. We may save the results in multiple scalar fields, one for each simulated time step. Alternatively, we can save the data in a single scalar field that changes over time. For this, we extend the definition of scalar fields by an additional parameter $t \in T$ that refers to the current time

$$f : D \times T \to \mathbb{R}\,,$$
$$f(\mathbf{x}, t) = s\,.$$

T depicts the temporal dimension, which is usually an interval of $\mathbb{R}$.

gradient & ∇-operator

For this thesis, we assume the scalar fields are differentiable. Thus we can compute the *spatial gradient* of the scalar field for each time step $t \in T$ at each position $\mathbf{x} \in D$. The gradient combines the partial derivative in each spatial dimension to the gradient vector $\mathbf{j} = (j_1, j_2, ..., j_n)^{\mathrm{T}} \in \mathbb{R}^n$.

It gives the direction with the steepest slope of scalar values. Formally we define the spatial gradient for the scalar field as follows:

$$\nabla f : D \times T \to \mathbb{R}^n \,,$$

$$\nabla f(\mathbf{x}, t) = \left(\frac{\partial f(\mathbf{x}, t)}{\partial x_1}, \frac{\partial f(\mathbf{x}, t)}{\partial x_2}, ..., \frac{\partial f(\mathbf{x}, t)}{\partial x_n} \right)^{\mathrm{T}} = \mathbf{j} \,.$$

The symbol ∇ denotes the *Nabla-operator*, a vector combining the partial derivatives for the spatial dimensions. We use it to simplify the notation of several differential quantities. It is defined as:

$$\nabla = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, ..., \frac{\partial}{\partial x_n} \right)^{\mathrm{T}} .$$

Hessian matrix

For scalar fields also the *Hessian* matrix is of interest. The Hessian is an $(n \times n)$-matrix that combines the second-order partial derivatives of the scalar field. It is used to classify *critical points*, and it is defined as:

$$\mathbf{H}f = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x},t)}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x},t)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x},t)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x},t)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x},t)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\mathbf{x},t)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x},t)}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x},t)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x},t)}{\partial x_n^2} \end{bmatrix} .$$

Examples of scalar fields are charts representing the value of stock markets (1D), a map showing the temperature of a weather forecast (2D), or a Computer Tomography scan (3D).

2.1.2 *Visualization*

Depending on the data encoded and the particular used application, several approaches exist to display scalar fields. We pick up the classification for these techniques given by Oster [113]. We focus on methods relevant to this thesis and divide techniques into *image-based* methods and *geometry-based* methods.

Image-Based Methods

Image-based methods try to give a dense visualization of the scalar field's data. Therefore, we map each position's scalar value to a specific property used for the visualization.

line plot

A common approach is to map the scalar value to a spatial position. If we have a 1D scalar field, we can picture the scalar field as a function graph or line plot. Today's office applications implement this technique. Therefore it is well known and understood even for non-experts. Figure 2.1 shows an example – the intra-day development of a stock market index.

Figure 2.1: Line Plot. A visualization of the value development of a stock market index. Such graphs are an example of a widely used visualization technique of 1D scalar fields. They are so common that the google search engine generates them automatically for some queries. The shown example is a screenshot of the top result after performing a search for *"DAX performance index."* (Image adapted from www.google.com [5])

height-mapping

For 2D scalar fields, the according technique exists; it is called *height-mapping*. We interpret the scalar values as intensity levels and map them to particular heights. Thus the 2D field gets transformed into a 3D surface. Figure 2.2 (left) shows an artificial scalar field with the corresponding surface obtained by height-mapping.

color-mapping

A technique we regularly use in this thesis is *color-mapping*. This method maps scalar values to colors from a predefined color map. Thereby, the scalar field gets transformed into an image or a texture. Images are displayed directly; textures are mapped to a surface. Color-mapping is a simple yet powerful technique that is easy to implement and quickly gives insights into data. Physical maps that can be found in atlases are a well-known example. The height of a landmass is mapped to a color ranging from green to brown, sea levels are shown as blue. Figure 2.2 (right) shows a picture of an Augmented Reality Sandbox [126] that uses a similar blue-green-brown map to color a scanned landscape. The selection of the color map plays an essential role in this technique and needs special attention. Research in color-theory and cartography shows a strong influence of the selected color map on the data's intuitive interpretation. We refer to the literature for more information on this, especially the works from the research groups around Brewer [15, 16], Kindlmann [81], Bujack [20, 110, 136], and Crameri [29] are a good start.

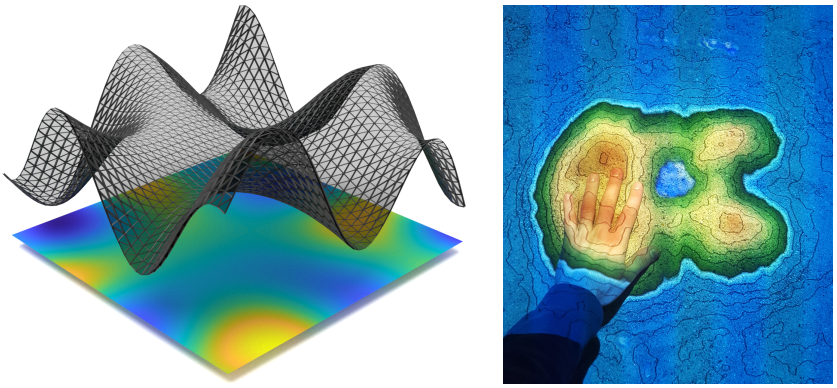


Figure 2.2: HEIGHT-MAPPING AND COLOR-MAPPING. Two examples of scalar field visualization. Left: A combination of height-mapping and color-mapping. The height of the surface depicts the underlying scalar values. Right: A picture of an Augmented Reality Sandbox. The measured landscape is colored based on its height and resembles physical maps known from atlases. (Image source for left picture: Gerrits [57])

Geometry-Based Methods

Geometry-based methods do not aim for the visualization of the data itself. Instead, they try to extract meaningful features from the scalar fields and display them. We want to introduce the following features relevant to this thesis: *isocontours*, *critical points*, and *ridge lines*.

isocontours & isolines

The term *iso* has an ancient Greek origin and means *equal*. All locations of a scalar field with the same scalar value form a set of *isocontours*, i.e.,

$$\mathcal{C}(s) = \{\mathbf{x} \in D \mid f(\mathbf{x}) = s\} \ . \tag{2.1}$$

For isocontours in 2D scalar fields, we also use the term *isolines*. In the previous section, we explained color-mapping that assigns colors to particular scalar values. Usually, the color maps show a smooth transition between the different color values. Isolines break this smooth transition between colors. We visualize isolines with a color that is distinct from its surrounding; therefore, they become visible. Figure 2.2 (right) shows an example, the dark lines are isolines. When the underlying scalar field is smooth and continuous, isolines always form closed curves or end in the domain border. Note, Equation (2.1) and the described properties are valid for any dimension, i.e., isocontours form closed manifolds in higher dimensions. Isocontours are a standard technique in cartography and meteorology. They are well researched [17, 35, 156, 166], and efficient algorithms for their computation exist [98].

critical point

Critical points are locations where the scalar field shows a local extremum. At these positions, the gradient $\nabla f(\mathbf{x}, t)$ maps to the zero vector $\mathbf{0}$. These locations have to be isolated, meaning there is no other location in their neighborhood showing the same behavior. Otherwise,

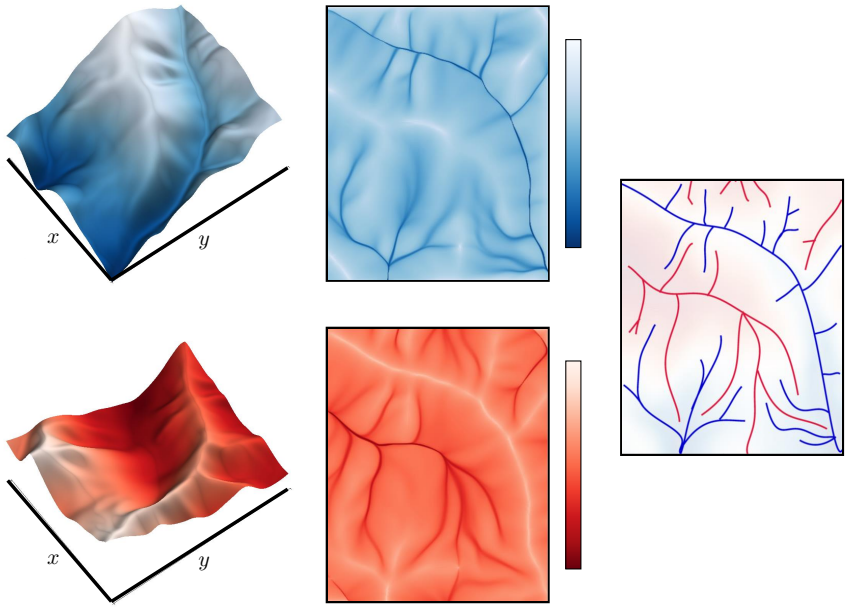


Figure 2.3: RIDGE AND VALLEY LINES. Different visualizations for a scalar field. Left: Height field visualizations of a scalar field and the inverted field. Center: Color-mapping of the same fields. Right: Ridge and valley networks with prominent ridge lines (red) and valley lines (blue). (Image adapted from Anand et al. [2])

it will no longer be a point, which implies furthermore critical points have a dimensionality of 0. We distinguish between local maxima, local minima, and saddles. To determine the type, we have to perform an eigenanalysis of the Hessian matrix at the point's location. If the Hessian's eigenvalues are all negative, it is a local maximum; if all are positive, it is a local minimum. If it has positive and negative eigenvalues, we classify the point as a saddle.

ridge & valley lines

Unlike critical points, *ridge lines* are geometric features that have a dimensionality of 1. One can imagine them as curves that show local maximum behavior. Depending on the use case, there exist several formal definitions for ridges [120, 141]. An elegant and often used one is the definition as height ridges given for arbitrary dimensions by Eberly [35]. There exist different techniques to extract feature lines that fulfill the height ridge definition. Depending on the particular implementation, additional filtering steps are necessary to remove false positives or less significant ridge lines [117]. *Valley lines* are the opposite of ridge lines, i.e., curves with local minimum behavior. If we are interested in valley lines, we can search for height ridges in the inverted scalar field $-f(\mathbf{x}, t)$. Figure 2.3 shows height-mappings, color-mappings, and extracted ridge and valley lines of a scalar field.

2.1.3 *Relevance for this Thesis*

We use scalar fields in different parts of this thesis. In many cases, we use color-mapping to visualize them. Chapter 5 presents drift fields. In essence, a drift field is a combination of two scalar fields with special relations to each other. Furthermore, we extract isolines from drift fields to describe particle movement. In Chapter 6, we compute FTLE fields as scalar fields and extract ridge lines. Chapter 7 introduces recirculation surfaces. For their computation, we derive a distance function that can be interpreted as a scalar field. We then search for all locations where this field becomes 0.

2.2 VECTOR FIELDS

Vector fields are similar to scalar fields, but they map to vectors rather than scalars. We typically use them to describe a movement or a force; examples are velocity fields, magnetic fields, or gravity fields. The standard way to represent flows is to use vector fields. Therefore, they are the primarily used data format for Flow Visualization. When a vector field describes a flow, it usually contains the velocity at discrete locations, i.e., it uses the Eulerian point of view. We also use the terms *(unsteady) flow field* or *velocity field* as synonyms.

2.2.1 *Definition*

definition vector field

We distinguish between steady and unsteady vector fields. A steady vector field is time-independent, i.e., it does not change over time. We define the steady vector field $\mathbf{v}(\mathbf{x})$ as a vector-valued function over the spatial domain D that assigns a vector $\mathbf{u} = (u_1, u_2, ..., u_n)^{\mathrm{T}}$ to each position $\mathbf{x} = (x_1, x_2, ..., x_n)^{\mathrm{T}}$

$$\mathbf{v} : D \to \mathbb{R}^n \,,$$
$$\mathbf{v}(\mathbf{x}) = \mathbf{u} \,,$$

with $\mathbf{x} \in D$, $D \subseteq \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^n$. If the vector field is unsteady, it varies over time. We extend the definition by an additional parameter $t \in T$ that refers to the time. Hence we define a time-dependent vector field as:

$$\mathbf{v} : D \times T \to \mathbb{R}^n \,, \tag{2.2}$$
$$\mathbf{v}(\mathbf{x}, t) = \mathbf{u} \,. \tag{2.3}$$

The temporal domain T is an interval in $\mathbb{R}$. Projecting an unsteady vector field $\mathbf{v}(\mathbf{x}, t)$ into $(n+1)$-dimensional space allows us to interpret it as a steady vector field $\widetilde{\mathbf{v}}(\mathbf{x}, t)$ in space-time

$$\widetilde{\mathbf{v}} : D \times T \to \mathbb{R}^{n+1} ,$$

$$\widetilde{\mathbf{v}}(\mathbf{x}, t) = \begin{pmatrix} \mathbf{v}(\mathbf{x}, t) \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{u} \\ 1 \end{pmatrix} .$$

We can define vector fields for arbitrary dimensions. In this thesis, we consider flows. Hence in most cases, we set $n = 2$ for 2D flows or $n = 3$ for 3D flows. For 2D and 3D vector fields, we may refer to the spatial dimensions as (x, y) or (x, y, z).

Just like scalar fields, we assume that vector fields are differentiable. Therefore, we can compute the partial derivatives of a time-dependent vector field

$$\mathbf{v}_i = \frac{\partial \mathbf{v}(\mathbf{x}, t)}{\partial x_i} , \quad \mathbf{v}_t = \frac{\partial \mathbf{v}(\mathbf{x}, t)}{\partial t} ,$$

with $i \in \{1, ..., n\}$. The partial derivative with respect to t is only relevant to unsteady vector fields. A combination of the spatial derivatives to a single matrix yields the *Jacobian* matrix

Jacobian matrix

$$\nabla \mathbf{v} : D \times T \to \mathbb{R}^{n \times n} ,$$
$$\mathbf{J} = (\nabla \mathbf{v})^{\mathrm{T}} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n) .$$

The Jacobian matrix is an $(n \times n)$-matrix containing first-order information about the current location's flow behavior. We use an eigenanalysis of the Jacobian matrix to classify first-order critical points. For more information on this, we refer to the literature [74, 75, 171].

2.2.2 *Visualization*

There exist various methods to visualize vector fields. A widely used categorization divides the methods into *direct*, *texture-based*, *topology- and feature-based*, and *geometry-based* [14, 19, 123]. We adopt this classification, give a short overview, and focus again on the techniques relevant to this thesis.

Direct Methods

Direct methods are the most straightforward techniques to visualize vector fields. They aim for a direct mapping of vectorial data and derived quantities to a visual representation. There is no complicated extraction or conversion step in between. Derived data is, e.g., velocity, curl, divergence, or other characteristic field properties. If the derived information is scalar data, we can apply color-mapping.

arrow plot

Otherwise, we map the data to a set of simple visual objects referred to as *glyphs* [11]. Therefore, we evaluate the vector field at several positions and place the corresponding glyph. *Arrow plots* are an intuitive and straightforward example of this technique [83, 142]. The used arrow glyphs encode the underlying vector. The arrow points in a particular direction, and its length depicts the magnitude of the velocity. Depending on the given data, the arrows need an appropriate scaling to result in a meaningful picture. Hence the relation between different arrow glyphs is essential. Due to occlusion, the number of glyphs that can be used is limited. For 3D arrow plots, the occlusion problem increases. Besides, the arrow's position and orientation can be challenging to understand due to the perspective. Figure 3.1 shows an example of an arrow plot.

Texture-Based Methods

Texture-based methods aim for a dense visualization by generating or modifying a noise texture based on local flow attributes. The resulting texture resembles the vector field behavior. These methods are primarily of interest to 2D vector fields or vector fields on 2D manifolds. For 3D, the occlusion problem arises.

LIC

One of the most popular techniques covering the vector field with dense visualization cues is *Line Integral Convolution* (LIC) [21]. We align a random noise texture and a steady vector field on a Cartesian grid. To determine the color for a position, we compute the streamline through it. The color values of the noise texture along this streamline are combined as a weighted integral. The combined value is used as the color value for the particular position. The result is a smoothed image with similar colors for locations along streamlines. One can also imagine this process as *'smearing'* out the noise texture along the vector field. In the past decades, much research was dedicated to improving LIC [111], e.g., adding additional visual cues [167], extending LIC to unsteady flows [70] and 3D [38], or decreasing the computation time [73, 151]. Figure 2.4 shows an example of a LIC texture visualization. For more information on texture-based methods, we refer to the literature [19, 90, 91].

Topology- and Feature-Based Methods

Topology-based methods aim for a sparse representation of the vector field by reducing the data to essential features. The first step for these techniques is the extraction of meaningful flow features. Based on these, we compute a topological skeleton. The skeleton separates the flow into regions with similar behavior. Another option is to visualize the extracted features themselves. Due to this, feature-based methods also appear as a distinct class of techniques in the literature [19, 123]. In the following, we introduce some of the most relevant features: *critical points* and *vortices*.

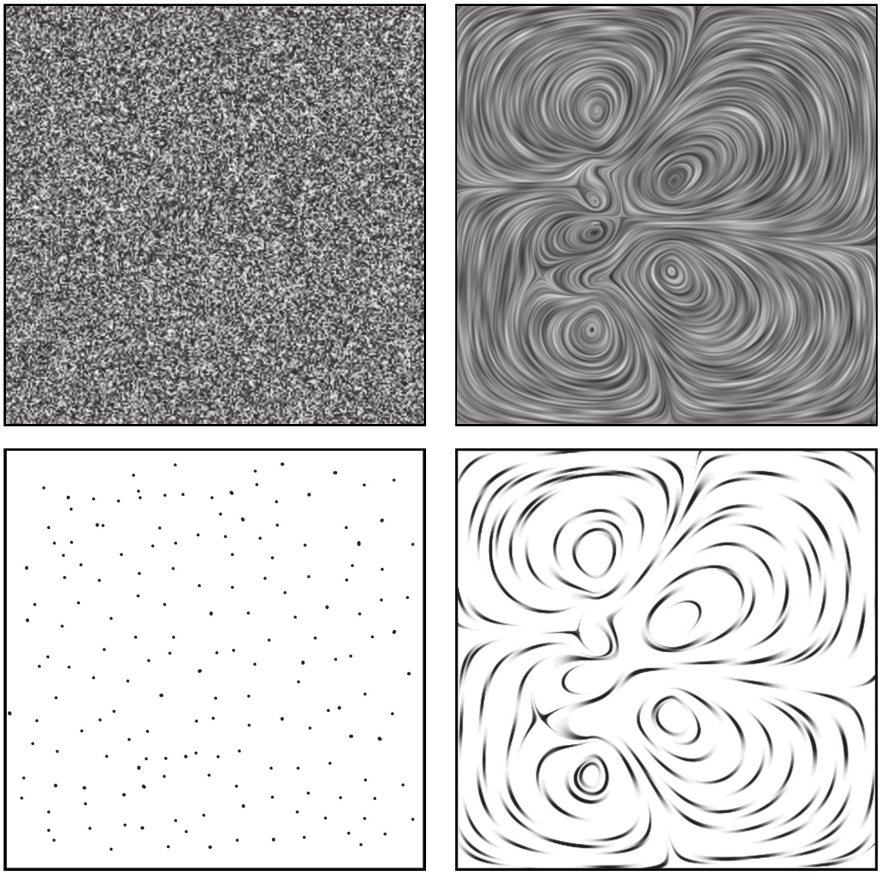


Figure 2.4: Line Integral Convolution. Examples of different noise textures (left) and the resulting LIC images (right). The vector field blurs the noise texture along its streamlines. Top: Dense white noise texture. Bottom: Sparse noise texture with black spots. (Image adapted from Netzel and Weiskopf [111])

critical points

A vector field $\mathbf{v}$ has a *critical point* at location $\mathbf{x}$ when it maps to the zero vector $\mathbf{0}$. The critical point is isolated when $\mathbf{v}$ is different from the zero vector in the vicinity of $\mathbf{x}$

$$\begin{aligned} \mathbf{v}(\mathbf{x}, t) &= \mathbf{0}\,, \\ \mathbf{v}(\mathbf{x} + \epsilon\mathbf{r}, t) &\neq \mathbf{0}\,. \end{aligned}$$

We classify critical points by flow behavior in their neighborhood described by the Jacobian matrix $\mathbf{J}(\mathbf{x})$. If its determinant is non-zero, then $\mathbf{x}$ is a first-order critical point, i.e., $\det(\mathbf{J}(\mathbf{x})) \neq 0$. An eigenanalysis of $\mathbf{J}(\mathbf{x})$ helps to classify the critical point. If the real parts of the eigenvalues are:

- all positive, $\mathbf{x}$ is a source with outflow behavior;
- all negative, $\mathbf{x}$ is a sink with inflow behavior;
- positive and negative, $\mathbf{x}$ is a saddle.

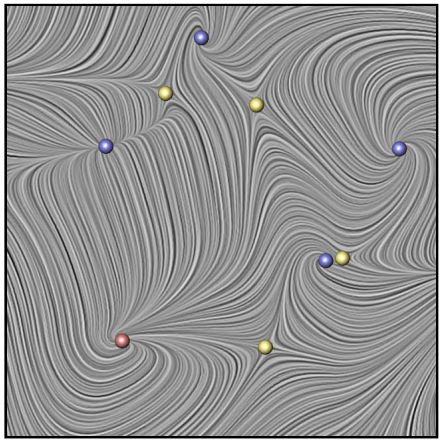


Figure 2.5: CRITICAL POINTS. LIC image of a vector field with critical points; sources (red), sinks (blue), and saddles (yellow). (Image source: Weinkauf [171])

If the eigenvalues have an imaginary part, the neighborhood of the location $\mathbf{x}$ shows swirling behavior. Figure 2.5 shows a LIC visualization combined with markers that indicate critical points. For more detailed information about critical points and their meaning for vector field topology, we refer to the literature by Helman and Hesselink [74, 75], Weinkauf [171], and Scheuermann et al. [138, 139].

vortex

A *vortex* is a structure that shows swirling flow behavior around a center. A real-world example that everyone knows is a tornado. Vortices emerge and persist; therefore, they are a defining feature for many flows. They are of practical relevance in many applications, e.g., we initiate them to stimulate fluid mixing or try to prevent them around aircraft. Vortices are one of the most discussed flow features, and many different methods for their extraction exist. Because they are not further relevant to this thesis, we refer to the literature for more information, e.g., Günther and Theisel recently presented an extensive survey regarding vortex extraction [62].

Geometry-Based Methods

particle seeding

Geometry-based methods visualize the flow with geometric primitives that are generated or transformed by the underlying flow data. Most of the techniques we sort into this class rely on observing massless particles in the flow [123]. We denote particles' placement in the flow as *seeding*. Geometry-based methods are classified further based on the seeding object's dimensionality. For example, we have 0D objects if we place individual particles as points, 1D if multiple particles form a line segment or curve, and 2D if they create a planar object. Based on the seeding structure's advection, we can compute geometric objects that present the flow behavior. Many methods provide different strategies for seeding particles or a robust construction for the geometric primitives.

McLoughlin et al. [105], as well as Edmunds et al. [36], give useful surveys on different methods.

2.2.3 *Relevance for this Thesis*

Vector fields appear throughout this whole thesis. Primarily we utilize them to describe flow data sets. We use synthetic flows that we define as analytic vector fields and simulated flows stored as discrete sampled vector fields. The vector fields appear in steady and unsteady form in 2D or 3D. Chapter 3 introduces the Double Gyre, an analytic unsteady vector field we use as a benchmark for all developed approaches. Although vector fields are not the final result for most of our approaches, we utilize them as a computation tool and derive meaningful information. In Chapter 4, we perform a deformation and use a vector field to control its strength. We describe drift fields in Chapter 5 by combining two scalar fields, which results in a vector field representation. The spatial gradient of this representation plays an essential role in the drift field definition. In Chapter 7, we utilize a derived vector field to find locations that belong to recirculation surfaces.

This thesis' primary focus is the flow map that encodes the trajectories of particles in a flow. We describe these trajectories as curves and compute them by numerical integration in vector fields. We will cover this topic in the next section.

2.3 PARTICLE INTEGRATION & INTEGRAL CURVES

When we inspect a massless particle in a flow, we are usually interested in the particle's trajectory. Therefore we place the particle in a flow, the underlying vector field then advects it, and we observe the path.

When we seed a massless particle at location $\mathbf{x}_0$ in a steady flow $\mathbf{v}(\mathbf{x})$, we describe the resulting path as a curve $\mathbf{c}(\tau)$. This curve yields the location of the particle after the integration time τ. The integration time τ denotes the period that has passed between the seeding and the current observation. The location is the solution of the autonomous *ordinary differential equation* (ODE)

ODE

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\,\mathbf{c}(\tau) = \mathbf{v}\left(\mathbf{c}(\tau)\right) \quad \text{with} \quad \mathbf{c}(0) = \mathbf{x}_0\ . \tag{2.4}$$

If we observe a particle in an unsteady flow $\mathbf{v}(\mathbf{x}, t)$, we additionally have to specify the seeding time t_0. The following ODE then describes the particle's location:

$$\frac{\mathrm{d}}{\mathrm{d}t}\,\mathbf{c}(t) = \mathbf{v}\left(\mathbf{c}(t), t\right) \quad \text{with} \quad \mathbf{c}(t_0) = \mathbf{x}_0\ . \tag{2.5}$$

If we augment the unsteady vector field $\mathbf{v}(\mathbf{x}, t)$ by one dimension and treat time t as an explicit state variable, we can rewrite Equation (2.5) as an autonomous ODE

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\begin{pmatrix}\mathbf{c}\\ t\end{pmatrix}(\tau) = \begin{pmatrix}\mathbf{v}(\mathbf{c}, t)\\ 1\end{pmatrix} \quad \text{with} \quad \begin{pmatrix}\mathbf{c}\\ t\end{pmatrix}(0) = \begin{pmatrix}\mathbf{x}_0\\ t_0\end{pmatrix} . \tag{2.6}$$

In general, we cannot obtain an analytic solution for the ODE. To solve them, we have to perform numerical integration. The seeding position and seeding time $(\mathbf{x}_0, t_0)$ are our initial conditions. Numerical integrators try to estimate the next location if the flow advects the particle for a short period. The particle moves a small step in this direction, and the time is advanced accordingly. We repeat this process until we reach the desired integration time.

There exist many different methods for numerical integration. To estimate particle trajectories in flows, we need an accurate prediction of the next particle location. The *fourth-order Runge-Kutta* method is suitable for this task [13]. It presents a fair tradeoff between computational costs and accuracy. For the results presented in this thesis, we used an implementation with an adaptive step size that follows Stalling and Hege's description [151].

tangent curves

Equation (2.4) puts a curve's derivative with respect to τ in relation to the vector field. The first derivative at a location of $\mathbf{c}(\tau)$ describes its tangent. If a curve's tangent coincides in every point with the underlying vector field, we call it a *tangent curve*. When we integrate a massless particle in a vector field $\mathbf{v}(\mathbf{x})$, we get its path through the corresponding flow as a curve $\mathbf{c}(\tau)$ – this is simultaneously a tangent curve.

integral curves

Depending on the seeding strategy for the particles and the method for path evaluation, we can derive different kinds of characteristic curves that describe the flow. We compute these curves by numerical integration; hence, we call them *integral curves*. For unsteady flows, we distinguish between four types of integral curves: *streamlines*, *pathlines*, *streaklines*, and *timelines*. Figure 2.6 illustrates the behavior in space-time and gives examples of the different types of integral curves. With the correct setup of the autonomous ODE in Equation (2.5), we can describe all of these curves as tangent curves of derived vector fields [169, 170].

2.3.1 *Streamlines*

stream-lines

We consider a steady vector field $\mathbf{v}(\mathbf{x})$ and place a massless particle in the flow. We can compute the particle's path by solving the ODE formulated in Equation (2.4) and describe it as a curve. This curve is tangent to the vector field at every point, and we call it *streamline*. Streamlines do not intersect. That implies, for every location exists exactly one streamline that passes through.

For unsteady vector fields $\mathbf{v}(\mathbf{x}, t)$, we have to extend the streamline definition. A streamline is a curve tangent to the vector field in its *current state*. We can compute it by integrating a particle in the vector field for a fixed time t_c. This is equivalent to increasing the dimension of the vector field and integrating the particle trajectory in the new derived vector field

$$\mathbf{s}(\mathbf{x}, t) = \begin{pmatrix} \mathbf{v}(\mathbf{x}, t) \\ 0 \end{pmatrix} .$$

Streamlines show a vector field's current flow direction and are a useful visualization tool for steady vector fields. Their suitability for unsteady vector fields is limited. It is impossible to estimate a particle's path from a single streamline visualization because it does not incorporate temporal information. For this case, pathlines, streaklines, and timelines are the better choice.

2.3.2 *Pathlines*

pathlines

We consider an unsteady vector field $\mathbf{v}(\mathbf{x}, t)$ and place a massless particle in the flow. The path this particle follows is called a *pathline*. We can compute the pathline by solving the ODE from Equation (2.5). Alternatively, we can increase the dimension of $\mathbf{v}(\mathbf{x}, t)$ and estimate the pathline as a tangent curve from the steady vector field

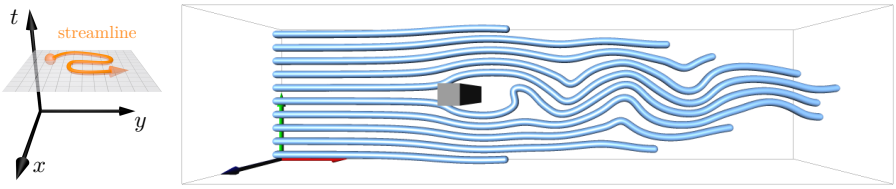
$$\mathbf{p}(\mathbf{x}, t) = \begin{pmatrix} \mathbf{v}(\mathbf{x}, t) \\ 1 \end{pmatrix} .$$

Pathlines visualize the behavior of particles in an unsteady flow over time. They can intersect in the spatial domain. In fact, an infinite number of pathlines passes through every location in space. Imagine two particles placed at the same starting position $\mathbf{x}_0$ but at two slightly different starting times in an unsteady flow. Usually, they will take different paths through the flow, but their pathlines intersect in $\mathbf{x}_0$. We keep this visual image of two different particles at the same start position in mind. We extend it and use it to explain streaklines.

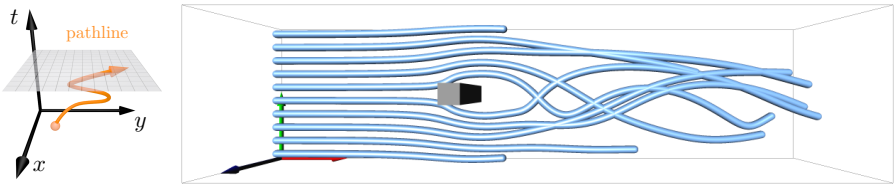
2.3.3 *Streaklines*

streaklines

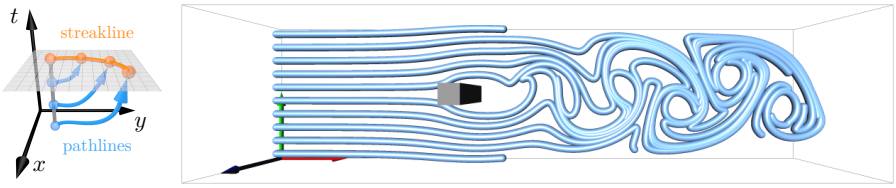
Again, we consider an unsteady vector field $\mathbf{v}(\mathbf{x}, t)$ that changes over time. We pick a fixed location in the flow and continuously seed particles at this position with ongoing time. The flow advects each of these particles along a different path. If we connect the advected positions of these particles to a curve, the result is a *streakline*. The smoke trail that forms from a blown-out candle is a real-world example. Weinkauf and Theisel [170] show that streaklines can be computed as tangent curves of a derived vector field in higher dimensions.



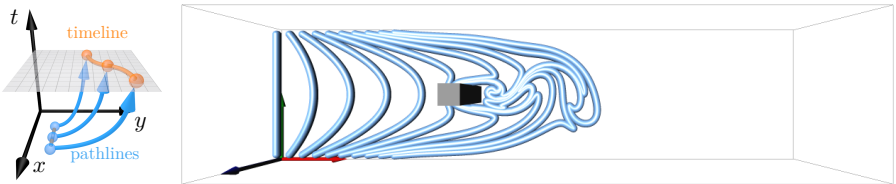
Streamlines: Particle trajectories for constant time t_c.



Pathlines: Particle trajectories for advancing time.



Streaklines: Same seeding position results in a trail.



Timelines: Particles form an advancing front.

Figure 2.6: INTEGRAL CURVES. Left column: Space-time illustrations. Right column: Examples of different types of integral curves. (Image adapted from Günther [60])

2.3.4 *Timelines*

timelines

Streaklines are computed by seeding multiple particles in an unsteady vector field $\mathbf{v}(\mathbf{x}, t)$ at the same location but at varying time steps. We compute *timelines* also by tracing multiple particles in an unsteady flow. However, to compute a timeline, we use the same seeding time but distribute the particles' starting positions along a seeding curve; the advected curve is a timeline. A thread floating on a river is a real-world example. Like streaklines, Weinkauf et al. [169] formulate timelines as tangent curves of particular vector fields in higher dimensions.

2.3.5 *Relevance for this Thesis*

Integral curves are the basic building block for many methods for the visualization of vector fields. For this thesis, we are particularly interested in pathlines. We use the Lagrangian point of view and must

compute them for our approaches. Usually, we do this by numerical integration. Chapter 5 presents a new form for flow representation; we use it to compute pathlines by local operations. This section explained that we describe them as tangent curves. Another way to describe pathlines is the flow map.

2.4 FLOW MAPS

The knowledge about massless particles' movement in unsteady flows is an essential tool for Flow Visualization. As we have seen in the previous section, one way to determine this movement is to compute pathlines in vector fields by numerical integration. Because we are mainly concerned about flows, we will use the term *velocity field* instead of vector field more often. When we consider a single pathline, we can evaluate it for varying τ to get different particle locations on the pathline. The current location where a particle is situated ultimately depends on its seeding position $\mathbf{x}$, its seeding time t, and the integration time τ. However, for a single pathline, seeding position and seeding time are fixed. Now, we want to extend the information from a single pathline to a *full set of pathlines*, i.e., we want to gather a construct that holds pathline information for all possible $(\mathbf{x}, t, \tau)$-combinations. This concept is expressed by the *flow map*. In the following, we give a formal definition, introduce the defining properties, and show the relations to velocity fields.

2.4.1 *Definition*

velocity field

We recapitulate the definition of unsteady velocity fields from Section 2.2, but this time we are more specific about the temporal domain. We define an unsteady velocity field as

$$\mathbf{v} : D \times T_t \to \mathbb{R}^n \,,$$
$$\mathbf{v}(\mathbf{x}, t) \,,$$

with $\mathbf{x} \in D$ and $t \in T_t$. The temporal domain has an additional subscript, which explicitly refers to the current temporal state t of the velocity field. It is part of the finite time interval $T_t = [t_s, t_e]$.

definition flow map

The flow map $\phi(\mathbf{x}, t, \tau)$ describing the same flow is a vector-valued function that contains the location of a particle seeded at $(\mathbf{x}, t)$ and advected by the velocity field for a period τ. We define the flow map as:

$$\phi : D \times T_t \times T_\tau \to D \,,$$
$$\phi(\mathbf{x}, t, \tau) \,.$$

For better readability, we also use the following abbreviations equivalently $\phi(\mathbf{x}, t, \tau) = \phi_t^\tau(\mathbf{x}) = \phi$, whenever seeding position, seeding time, or integration time are negligible or should be clear from the context.

spatial & temporal domain

D is the spatial domain containing all seeding and destination locations. T_t is the temporal range for possible seeding times. T_τ denotes a different interval containing valid integration times, with $T_\tau = [\tau_s, \tau_e]$. An integration time is valid if

$$(t + \tau) \in T_t \,,$$

i.e., we get a defined seeding time after the integration.

dimensionality

A flow map $\phi_t^\tau(\mathbf{x})$ is a map $\mathbb{R}^{n+2} \to \mathbb{R}^n$, where n denotes the number of spatial dimensions. The remaining two dimensions are the starting time t and integration time τ. In the remainder of this thesis $n \in \{1, 2, 3\}$, i.e., we consider 1D, 2D, and 3D flows. We denote the spatial dimensions as x, (x, y) or (x, y, z). We consider the following flow map types:

$$\begin{aligned}
&\text{nD flow} \quad \phi : \mathbb{R}^{n+2} \to \mathbb{R}^n \\
&\text{3D flow} \quad \phi : \mathbb{R}^5 \quad\;\; \to \mathbb{R}^3 \\
&\text{2D flow} \quad \phi : \mathbb{R}^4 \quad\;\; \to \mathbb{R}^2 \\
&\text{1D flow} \quad \phi : \mathbb{R}^3 \quad\;\; \to \mathbb{R} \,.
\end{aligned}$$

In practice, the use of 2D or 3D unsteady flows is common. In this thesis, we also use the 1D case to clarify concepts. However, all presented flow map properties and concepts hold for arbitrary dimensions.

The flow map encodes all possible locations for each pathline. To compute a complete pathline starting at $(\mathbf{x}_0, t_0)$ we perform consecutive evaluations of $\phi(\mathbf{x}_0, t_0, \tau)$ over a time interval $\tau \in [\tau_0, \tau_1] \subseteq T_\tau$.

For theoretical considerations, we assume that particles stay inside the flow, i.e., they neither leave the spatial nor the temporal domain. This constraint is necessary for some definitions we introduce in this thesis. Though, for the practical use case, this would be problematic. We would exclude many simulated flow data sets where particles enter or leave the domain. If a flow contains such parts, they need a particular treatment in each case or are treated as undefined.

partial derivatives

Like scalar fields and vector fields, we assume that flow maps are smooth and continuous in space and time. Therefore, we can compute the partial derivatives for each dimension

$$\phi_i = \frac{\partial \phi(\mathbf{x}, t, \tau)}{\partial x_i} \quad \text{with} \quad i \in \{1, 2, 3\} \,, \tag{2.7}$$

$$\phi_\tau = \frac{\partial \phi(\mathbf{x}, t, \tau)}{\partial \tau} \,, \tag{2.8}$$

$$\phi_t = \frac{\partial \phi(\mathbf{x}, t, \tau)}{\partial t} \,. \tag{2.9}$$

flow map gradient

The (spatial) *flow map gradient* $\nabla \phi_t^\tau(\mathbf{x})$ is a (2×2) respectively a (3×3) matrix combining the partial derivatives from Equation (2.7).

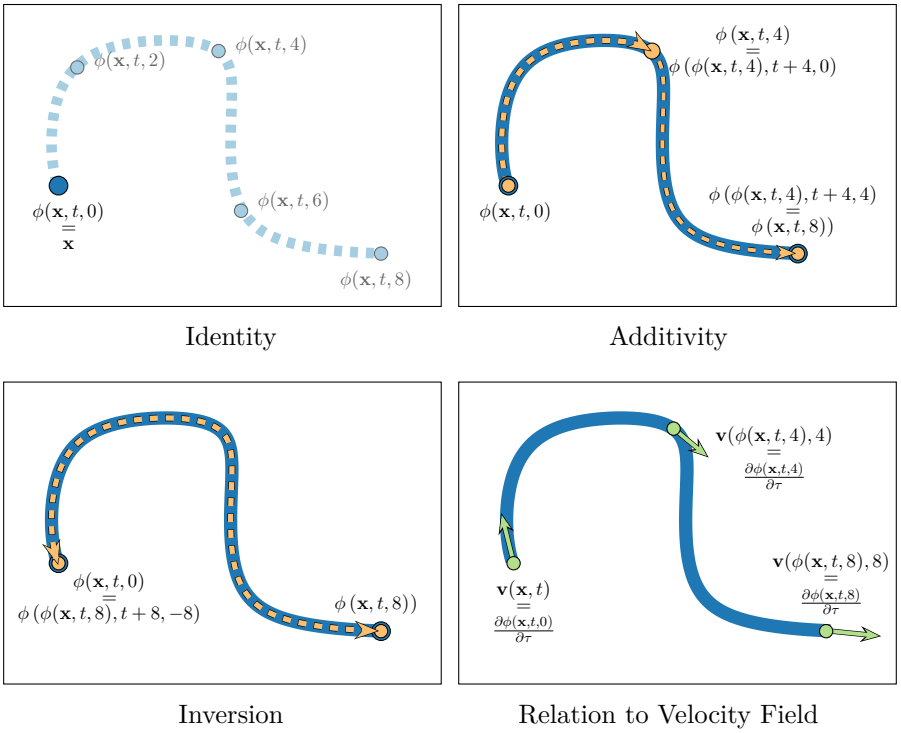


Figure 2.7: Flow Map Properties. Illustration of defining properties and the relation between flow map and velocity field. The blue line represents a single pathline seeded in an unsteady velocity field at location $(\mathbf{x}, t)$ and integrated for a period $\tau \in [0, 8]$. The integration time τ controls the particle's position on its pathline. The orange dashed line shows parts of the pathline for varying τ. The time derivative ϕ_τ corresponds to the velocity field represented by green arrow glyphs. Color markers indicate the particle's position on the pathline.

It describes the separation of particles seeded near $(\mathbf{x}, t)$ after the integration time τ. It is defined as:

$$\text{2D: } \nabla\phi_t^\tau(\mathbf{x}) = \left(\frac{\partial}{\partial x_1}\phi, \frac{\partial}{\partial x_2}\phi\right) \tag{2.10}$$

$$= \begin{pmatrix} \frac{\partial}{\partial x_1}\phi_x & \frac{\partial}{\partial x_2}\phi_x \\ \frac{\partial}{\partial x_1}\phi_y & \frac{\partial}{\partial x_2}\phi_y \end{pmatrix}, \tag{2.11}$$

$$\text{3D: } \nabla\phi_t^\tau(\mathbf{x}) = \left(\frac{\partial}{\partial x_1}\phi, \frac{\partial}{\partial x_2}\phi, \frac{\partial}{\partial x_3}\phi\right) \tag{2.12}$$

$$= \begin{pmatrix} \frac{\partial}{\partial x_1}\phi_x & \frac{\partial}{\partial x_2}\phi_x & \frac{\partial}{\partial x_3}\phi_x \\ \frac{\partial}{\partial x_1}\phi_y & \frac{\partial}{\partial x_2}\phi_y & \frac{\partial}{\partial x_3}\phi_y \\ \frac{\partial}{\partial x_1}\phi_z & \frac{\partial}{\partial x_2}\phi_z & \frac{\partial}{\partial x_3}\phi_z \end{pmatrix}. \tag{2.13}$$

2.4.2 *Defining Properties*

Not every map $\mathbb{R}^{n+2} \to \mathbb{R}^n$ is a flow map. A map $\phi_t^\tau(\mathbf{x})$ must fulfill further defining properties to be a flow map:

$$\text{Identity:} \qquad \phi(\mathbf{x}, t, 0) = \mathbf{x}\,, \tag{2.14}$$

$$\text{Additivity:} \qquad \phi(\phi(\mathbf{x}, t, \tau_1), t + \tau_1, \tau_2) = \phi(\mathbf{x}, t, \tau_1 + \tau_2)\,, \tag{2.15}$$

$$\text{Inversion:} \qquad \phi(\phi(\mathbf{x}, t, \tau_1), t + \tau_1, -\tau_1) = \mathbf{x}\,, \tag{2.16}$$

for all $\mathbf{x} \in D, \{t, (t + \tau_1), (t + \tau_1 + \tau_2)\} \in T_t$, and $\{\tau_1, \tau_2\} \in T_\tau$.

identity

Equation (2.14) denotes the flow map's *identity* for the particular case $\tau = 0$, i.e., a particle is mapped to its seeding position if there is no integration time.

additivity

Equation (2.15) describes the flow map's *additivity*, i.e., a particle stays on its pathline. It moves forward with increasing τ and backward with decreasing τ. The movement happens in several small consecutive steps or one big step. However, a particle never leaves its pathline.

inversion

Equation (2.16) denotes the *inversion* of the flow map. If τ is inverted, a massless particle has to move backward on its pathline, which means it will reach its seeding position again. The inversion can be seen as a particular case of additivity. Nevertheless, we list it explicitly as a property because it clarifies that we can invert the flow map by changing the sign of τ.

Figure 2.7 illustrates the defining properties using the example of a single pathline. Depending on the integration time τ, the particle takes different positions on the pathline. The flow map properties restrict this movement.

2.4.3 *Relations to Velocity Fields*

Equation (2.8) shows the time derivative ϕ_τ concerning τ; it is the direction in which a massless particle currently moves. From this, we can retrieve a particular connection between velocity fields and flow maps. If $\mathbf{v}(\mathbf{x}, t)$ and $\phi_t^\tau(\mathbf{x})$ describe the same flow and are furthermore both smooth and continuous, the following relations must hold:

$$\begin{aligned} \phi_\tau(\mathbf{x}, t, 0) &= \mathbf{v}(\mathbf{x}, t)\,, \\ \phi_\tau(\mathbf{x}, t, \tau) &= \mathbf{v}(\phi(\mathbf{x}, t, \tau), t + \tau)\,, \end{aligned}$$

i.e., the velocity field is the partial derivative of the flow map with respect to the integration time τ. Figure 2.7 (bottom right) illustrates the relation between the time derivative of the flow map and the velocity field.

2.4.4 *Relevance for this Thesis*

The flow map is the main focus of this thesis. Flow maps are a full Lagrangian representation of the corresponding flow.

If a full flow map is available, expensive numeric integration to compute pathlines becomes obsolete. This is a considerable advantage, especially for long integration times. In Chapter 4, we present the first systematic approach to modify flow maps while keeping the defining properties.

The high dimensionality of flow maps leads to much higher storage requirements when compared to velocity fields. In Chapter 5, we present drift fields. Drift fields are a Lagrangian flow representation similar to flow maps. Efficient local operations can extract pathlines, i.e., we do not need an expensive numerical integration. Furthermore, drift fields have the same dimensionality as the corresponding unsteady velocity field. Therefore they are cheap to store.

A systematic analysis of flow map properties can lead to new visualization techniques. Chapter 7 introduces recirculation surfaces, a new flow feature that takes the whole 5D space of the flow map into account.

A common way to visualize flow maps is pathline plots. With an increasing number of pathlines, the visualization quickly becomes cluttered and confusing. Figure 3.2 presents an example of a small number of pathlines, that already show this behavior. Therefore, dense pathline visualization is hardly possible. FTLE fields result in a space-filling dense visualization of Lagrangian flow properties. FTLE utilizes the spatial flow map gradient $\nabla\phi$ and is a standard tool to visualize derived parts of the flow map. In the next section, we give a short introduction to this topic.

2.5 LAGRANGIAN COHERENT STRUCTURES & THE FINITE-TIME LYAPUNOV EXPONENT

Finding useful visualizations for unsteady flows is in the interest of research for over two decades [19, 105]. To gain insights into the flow data, we usually have to observe its behavior over time. We can do this by observing the paths of massless particles in the flow. Because the flow changes over time, a particle's trajectory is sensitive to its initial conditions like seeding time and seeding location. Therefore it is often hard to extract meaningful information from a set of individual pathlines. However, there exist flow features that help us to describe the intricate movement patterns of massless particles. Some of these features are robust over time. A good example is the core line of a vortex – compare Section 2.2.2. *Lagrangian coherent structures* (LCS) are another prominent feature. In the first part of this section, we give an overview and explain what an LCS is. In the second part, we will introduce a tool for their extraction, the *finite-time Lyapunov exponent*.

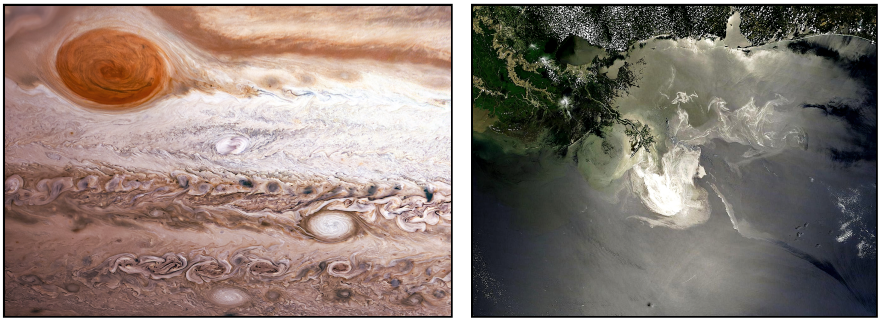


Figure 2.8: COHERENT LAGRANGIAN PATTERNS IN NATURE. Flows in nature whose material transport is influenced by LCS. Left: Atmospheric storms in Jupiter's southern hemisphere, including the old *Great Red Spot* and the young *Clyde's Spot*. Right: Oil spill in the Gulf of Mexico after the oil rig *Deep Water Horizon* exploded. (Image source: NASA/JPL-Caltech/SwRI/MSSS, NASA/GSFC)

2.5.1 *Lagrangian Coherent Structures*

hyperbolic LCS

As the name suggests, LCS are Lagrangian flow features that allow global statements about the flow. They persist for a finite or infinite time. For this thesis, we are interested in *hyperbolic* LCS. Hyperbolic LCS show either attracting, repelling, or shearing behavior to nearby massless particles. Furthermore, they act as barriers to material transport, e.g., a particle's path is not likely to cross an LCS. When we combine LCS, they form a robust skeleton of material surfaces that reveal ordered patterns for the rest of the flow [68]. Therefore, they are an essential tool to investigate the transport processes in unsteady flows. LCS are of interest for a wide range of applications like spiraling eddies in the Mediterranean Sea, prediction of ocean pollution, transport of warm water in the Gulf Stream, or steam rings blown by a volcano. Figure 2.8 shows two examples of natural phenomena containing LCS; storms in Jupiter's atmosphere and an oil spill in the Gulf of Mexico. Haller gives an excellent introduction to the topic of LCS, their properties, and their meaning for flow analysis [68]. In the past years, researchers introduced several methods to extract LCS from different data sets, summarized, e.g., by Hadjighasem et al. [63].

2.5.2 *Finite-Time Lyapunov Exponent*

The *finite-time Lyapunov exponent* (FTLE) is one of the most prominent tools to detect and analyze LCS in unsteady flows [120]. The Lyapunov exponents go back to research by the Russian physicist and mathematician Aleksandr Lyapunov in the 1890s. In the 1970s, they were frequently used in physics and mathematics to analyze dynamic systems' stability and predictability. Researchers introduced multiple different variants and techniques to compute Lyapunov exponents [114].

The finite-time variant became of interest in the late 1980s, especially for systems resulting from numerical simulations [59]. Haller used them for velocity fields of fluid flows and identified height ridges in FTLE fields as LCS [65–67]. Shadden et al. [145] have shown in 2005 that LCS computed from FTLE fields are an appropriate tool to describe the global material transport in unsteady flow fields because the material flux across them is negligible. Since then, FTLE became popular in the Flow Visualization community, and many different methods utilize it.

FTLE explanation & definition

The idea of FTLE is to measure the rate of separation between particles by a single scalar value. A set of massless particles is seeded close to each other. We observe the particles' movement over a finite range of time. Suppose the particles separate, then the FTLE increases. When the particles stay close to each other, the FTLE is low. The FTLE for a location $\mathbf{x}$, a starting time t, and an integration time τ is defined as:

$$\nabla := \nabla\phi(\mathbf{x}, t, \tau)\,, \tag{2.17}$$

$$\Delta := \nabla^{\mathrm{T}}\nabla\,, \tag{2.18}$$

$$FTLE(\mathbf{x}, t, \tau) = \frac{1}{|\tau|} \ln \sqrt{\lambda_{\max}(\Delta)}\,. \tag{2.19}$$

Cauchy-Green tensor

To compute the FTLE, we use the spatial flow map gradient $\nabla\phi_t^\tau(\mathbf{x})$. We use the abbreviation ∇ from Equation (2.17), specifically for this definition. By multiplying ∇ with its transpose ∇^{T}, we obtain the right *Cauchy-Green tensor* Δ. The Cauchy-Green tensor is a strain tensor. It contains information about the deformation of particles seeded very close to each other at the location $(\mathbf{x}, t)$ after an integration by τ [102]. As Δ is symmetric and positive semi-definite by construction, its eigenvalues $\lambda_i \geq 0$ are real, and the associated eigenvectors $\mathbf{e}_i$ are orthogonal. The largest eigenvalue of this tensor corresponds to the squared magnitude of the largest particle separation – therefore, the square root is taken. Because the flow map gradient $\nabla\phi_t^\tau(\mathbf{x})$ grows exponentially with increasing integration time, the natural logarithm is used and the value is normalized by $|\tau|$. When we do this computation for the whole spatial domain, we obtain a field with scalars specifically for the selected t and τ – the FTLE field. Figure 2.9 shows an example of an FTLE field with a ridge that corresponds to an LCS. The pathlines of two particles show separating behavior next to the ridge.

2.5.3 *Relevance for this Thesis*

FTLE fields are derived from the flow map gradient and visualize Lagrangian flow behavior. When we modify the flow map at a point in space-time, the FTLE for this location will also change. We take advantage of this in Chapter 4, where we introduce an approach to modify the flow map. We use FTLE fields to visualize the modified areas.

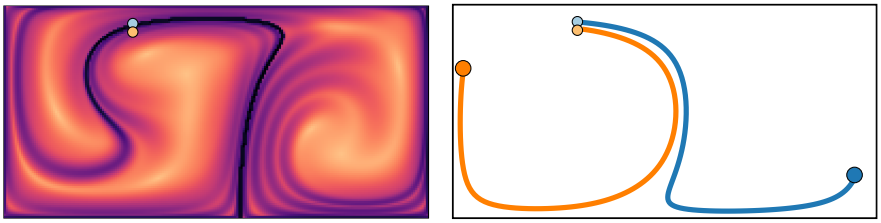


Figure 2.9: FTLE, LCS & PATHLINES. Left: The FTLE field for the Double Gyre with $t = 0$ and $\tau = 10$ shows a solid ridge in the center. This ridge reveals an LCS with separating behavior. Right: Pathlines of two particles seeded above and below the LCS. The markers indicate seeding positions (small) and the positions for $\tau = 10$ (medium). The particles stick together in the beginning but separate after a certain time.

For the calculation of FTLE fields, we need to compute the flow map gradient. A straightforward approach is to perform a discrete sampling of the spatial domain and approximate the gradient based on this discretization. Chapter 6 introduces a technique for an adaptive refinement of such a sampling grid to compute FTLE fields. However, first, we will explain the discretization on a regular sampling grid in the next section.

2.6 DISCRETIZATION, RECONSTRUCTION & DERIVATIVES

For many approaches in Scientific Visualization, we assume that the data we want to visualize represents a field with a continuous function. We can evaluate the data at any given point in the domain without expecting sudden jumps or gaps. The continuity is especially relevant to theoretical considerations. However, most visualization approaches use discretization to represent data sets [156]. In this form, the data is represented at discrete locations and not as a continuous function. We omit the data between these locations. To gain the discretization, we evaluate the data at specified locations and store the information we need, e.g., the original data itself or derived information like derivatives. The discrete representation is a sampling of the continuous field and therefore liable to effects like aliasing. By a careful selection of the discretization scheme and method, we can retain some properties of the continuous data, but in general, not all of them [164].

2.6.1 *Discretization of Data*

The data sets we consider in Flow Visualization often originate from simulation or measuring processes. We store the data as a set of samples. In general, the samples form a grid representing a subdivision of the domain into discrete non-overlapping cells. The samples serve as the cell vertices. The cells usually form a grid consisting of a set of connected lines in $\mathbb{R}$, polygons in $\mathbb{R}^2$, polyhedra in $\mathbb{R}^3$, or polytopes in higher

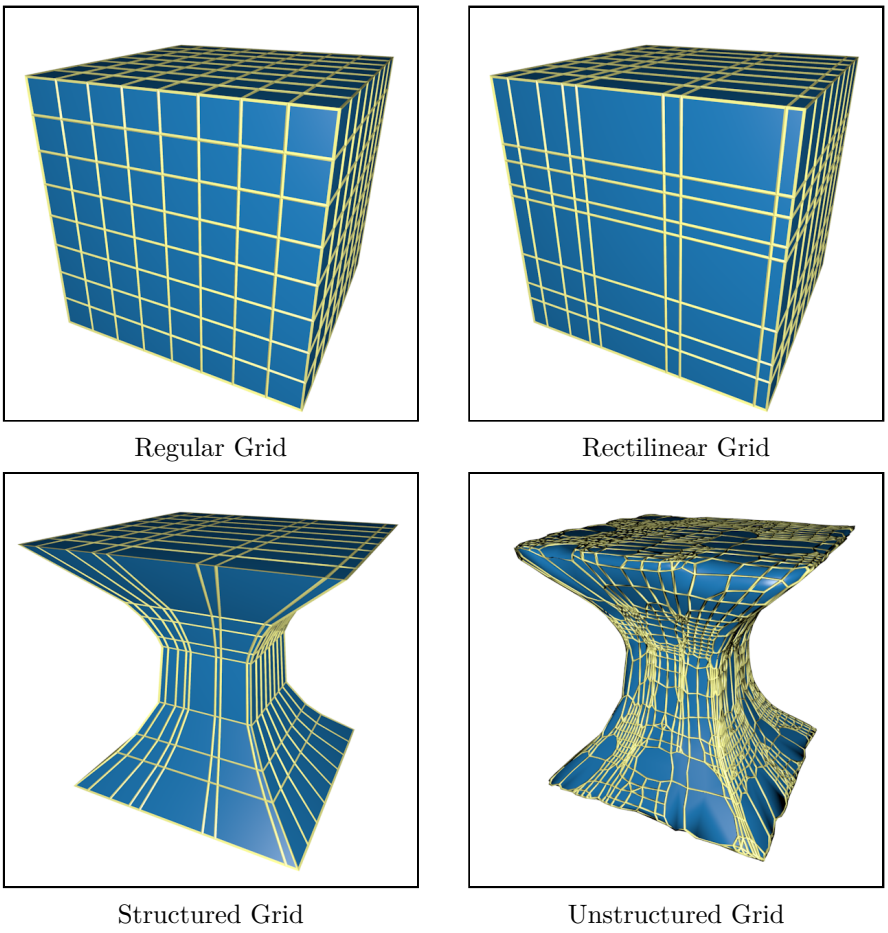


Figure 2.10: GRID TYPES. Examples of different grid types.

dimensions [156]. We can distinguish between two grid types: *structured grids* and *unstructured grids* [8].

Structured Grids

structured grids

The defining property of structured grids is the indexing of the samples. We identify each grid vertex by a tuple of continuous indexes corresponding to coordinates in a Cartesian system. Regular polytopes form the grid cells, e.g., quadrilaterals in 2D or hexahedra in 3D. These polytopes may differ in size and form [8]. However, we can explicitly specify cell locations as long as we keep the samples' index-based arrangement [156]. Structured grids are easy to handle, and elements can be accessed quickly. Figure 2.10 shows examples of different grid types.

regular grids

In this thesis, we primarily use regular grids, which are a subclass of structured grids. The sampling cells are rectilinear and have a uniform size in each dimension, i.e., the samples form a Cartesian grid.

Unstructured Grids

unstructured grids

In unstructured grids, the vertices and cells do not follow a specific layout. They are located quasi-randomly. Various element types may form the cells, e.g., tetrahedra, hexahedra, or prisms. Unstructured grids offer higher accuracy, especially when we approximate the border of geometries. Unfortunately, data access is much more complicated and computationally expensive, which led to different algorithms optimized for specific grid types [8, 52, 88].

2.6.2 *Reconstruction of Discretized Data*

Due to the sampling, data is only available at the grid points. We want to be able to evaluate the data at an arbitrary location in the domain. Therefore, we have to be able to reconstruct the data at intermediate locations between the samples. For this process, different techniques exist [156].

multi linear interpolation

One of the most popular approaches for structured grids is the *multi linear interpolation* between neighboring grid samples [125]. In the lower dimensions, we denote this as linear (1D), bilinear (2D), or trilinear (3D) interpolation. This reconstruction scheme is sufficient for many cases in Scientific Visualization. The majority of approaches in this thesis rely on multi linear interpolation. When using this approach, it is necessary to consider that the result is only $\mathcal{C}^0$-continuous at the cell borders, i.e., the first derivative is not smooth. If necessary, we can achieve a higher quality reconstruction at the costs of higher computational effort. For example, we can use higher-order interpolation schemes that take neighboring cells into account. For more information on this topic, we refer to the literature [1, 8, 101, 156].

2.6.3 *Derivatives of Discretized Data*

finite difference method

Another problem we have to solve when working with discretized data is the computation of derivatives. When we use regular grids, where the samples lie on a Cartesian grid, we can estimate the derivatives with the *finite difference method*. By a linear combination of neighboring grid points, we can approximate the derivatives.

We explain the method for the 2D case; the procedure is analogous for other dimensions. We assume that we have a continuous 2D function $f(\mathbf{x})$ sampled on a regular grid. Each sample's location is described by a pair of coordinates; $\mathbf{x} = (x, y)$ with $x, y \in \{0, 1, ..., n\}$. The number of samples in each direction is denoted by n. We can approximate

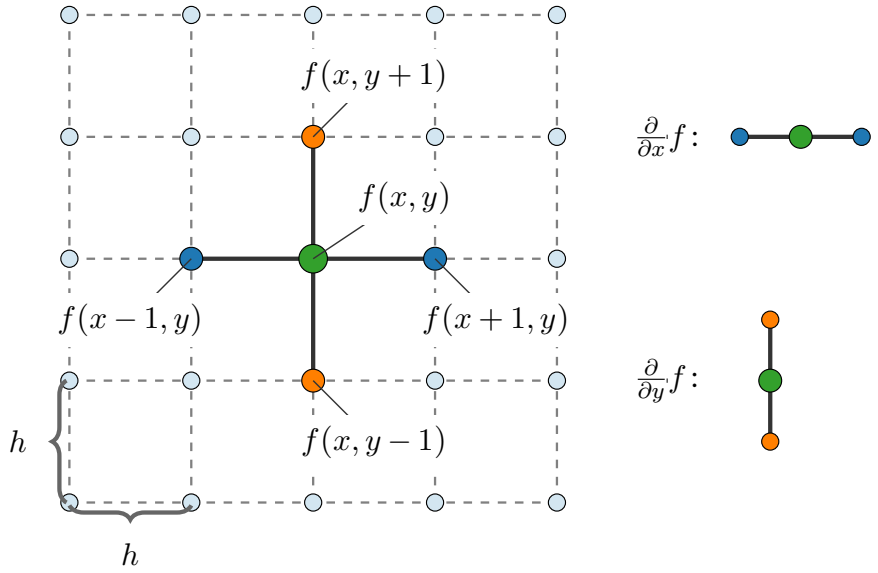


Figure 2.11: FINITE DIFFERENCES. We show an illustration of the stencil scheme used for central differences on a regular sampling grid. We want to approximate the derivative at the green marker. For the derivative in x-direction, we use the stencil with the blue markers. For the derivative in y-direction, we apply the stencil with the orange markers.

the first-order derivative at location (x, y) by the following difference schemes:

Forward Difference (2.20)

$$\frac{\partial f}{\partial x} \approx \frac{f(x+1, y) - f(x, y)}{h} \qquad \frac{\partial f}{\partial y} \approx \frac{f(x, y+1) - f(x, y)}{h}$$

Backward Difference (2.21)

$$\frac{\partial f}{\partial x} \approx \frac{f(x-1, y) - f(x, y)}{h} \qquad \frac{\partial f}{\partial y} \approx \frac{f(x, y-1) - f(x, y)}{h}$$

Central Difference: (2.22)

$$\frac{\partial f}{\partial x} \approx \frac{f(x+1, y) - f(x-1, y)}{2h} \quad \frac{\partial f}{\partial y} \approx \frac{f(x, y+1) - f(x, y-1)}{2h}\,.$$

The distance between two samples is denoted by h and is constant in each direction. We control the samples we use for the computation with a stencil. Equations (2.20) to (2.21) use a 2-point stencil, Equation (2.22) uses a 3-point stencil; these are the most common schemes. Different stencil sizes and sampling positions are possible but require separate coefficients. In general, a bigger stencil leads to a better derivative approximation at higher computational costs. Figure 2.11 illustrates a sampled 2D grid and the stencil for the central difference in x- and y-direction. If the data contains additional dimensions, we can perform

the finite difference in each direction. More information on the finite difference method can be found in [41, 152].

A prerequisite for the finite difference method is a uniform spacing between the grid samples, i.e., a constant factor h. For varying distances and stencils of 3 or more samples, a different scheme is required. A possible solution is to find a polynomial that interpolates the sampled values. We can then derive the polynomial and approximate the function derivative. For this thesis, we use interpolating *Lagrange polynomials*.

Lagrange polynomials

Given is a set of a $(n+1)$ samples in the form (x_i, y_i). The Lagrange polynomials $L_k^n(x)$

$$L_k^n(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}\,, \tag{2.23}$$

$$= \frac{x - x_0}{x_k - x_0} \,...\, \frac{x - x_{k-1}}{x_k - x_{k-1}} \cdot \frac{x - x_{k+1}}{x_k - x_{k+1}} \,...\, \frac{x - x_n}{x_k - x_n}\,,$$

provide a basis in the space of polynomials of degree n with the interpolating property

$$L_k^n(x_i) = \begin{cases} 0 & \text{if} \quad i \neq k \\ 1 & \text{if} \quad i = k \end{cases}\,.$$

For the given sample locations x_i the polynomial $P_n(x)$

$$P_n(x) = \sum_{k=0}^{n} y_k L_k^n(x)\,,$$

$$= y_0 L_0^n(x) + y_1 L_1^n(x) + ... + y_n L_n^n(x)\,,$$

interpolates the values y_i, i.e.,

$$P_n(x_i) = y_i\,, \quad \forall i \in \{0, ..., n\}\,.$$

This means, the coefficients of $P_n(x)$ are the interpolated values y_i. The $(n+1)$ samples needed for the definition of L_k^n, are the samples of the discretized function $f(\mathbf{x})$. Because $P_n(x)$ is a 1D function, we have to align it to the sampling grid. We can do this in each direction separately by fixing one dimension to x_c or y_c. This way, we define the samples for Equation (2.23) based on a sampled 2D function $f(\mathbf{x})$ as:

x-direction
$(x_i, y_i) = (x_i, f(x_i, y_c))\,, \quad i, c \in \{0, ..., n\},\ c$ is constant,

y-direction
$(x_i, y_i) = (y_i, f(x_c, y_i))\,, \quad i, c \in \{0, ..., n\},\ c$ is constant.

Once we computed the interpolating polynomial, we can use the closed-form solution to determine the derivative. Interpolation by polynomials can show oscillating behavior around the sampled function, in particular for higher degrees. In this thesis we use only quadratic $P_2(x)$ or cubic $P_3(x)$ polynomials to avoid this problem. In general, this results in good approximations of the first-order derivative.

2.6.4 *Relevance for this Thesis*

Discretization, reconstruction, and approximation of derivatives play an essential role in this thesis. We created all results we present throughout the thesis using discretized data sets. The complexity ranges from simple 1D functions we use, e.g., for the colormaps, to highly resolved discretizations of flow maps in 5D space. In almost all cases, we use a uniform sampling in each dimension, i.e., the sampling distance can differ in space and time. For the reconstruction, we utilize multi linear interpolation. We obtain most of the derivatives by central differences. Even though we created the results on discretized data, we used synthetic benchmarks in the analytic form to develop our approaches. Analytic benchmarks do not suffer from errors due to sampling. We can evaluate them at any position in space-time, compute ground truths, and compare them to sampled versions with different resolutions. Analytic data sets support systematic research and help to identify shortcomings of developed methods. One of the essential synthetic data sets for this thesis is the Double Gyre. We introduce it in the next chapter.

3

DOUBLE GYRE

This chapter is partly based on the publication:

S. Wolligandt, T. Wilde, C. Rössl, H. Theisel
A Modified Double Gyre with Ground Truth Hyperbolic Trajectories for Flow Visualization
Computer Graphics Forum, 2020

We assume we have an idea for a new flow feature that can help to understand flows better. We usually start by defining the specific properties we expect and formulate a sound theoretical description. Then we develop techniques to extract the new feature from flow data. Usually, we aim for generic methods that cover arbitrary flows. We start by solving simple cases and extend the technique to more difficult subjects step by step. We check our algorithms with tests during this process, each covering a particular issue on its own. When our implementation passes all tests, we will use our method on complex data sets and simulated flows. In this phase of development, we usually get results that give us new information about the data. The problem is that the results might *look correct* but *are wrong*. We still have to verify if the outcome is right for more complex data sets.

Therefore, the use of benchmark data sets is helpful. It is an advantage if benchmarks are well known and used by other researchers as well. They cover different levels of complexity, and we know the results that our new method should compute, or at least can predict them pretty well. A benchmark data set that is essential for this thesis is the Double Gyre.

3.1 EXPLANATION & DEFINITION

double gyre pattern

The double gyre is a flow pattern that consists of two gyres located next to each other. The gyres are oriented in opposite directions, i.e., one rotates clockwise the other in a counterclockwise direction. This pattern can be observed regularly in natural flows. For example, two counter-oriented gyres appear next to the spoon when we move it in a straight line through a cup of tea. Such gyres also appear on a global scale in wind-driven ocean currents. For example, we can observe seasonal gyres – a sub-polar gyre and a sub-tropical gyre – in ocean basins with a diameter of several thousand kilometers [148]. Over the past decades,

this pattern was studied several times in geophysical flow models [24, 109, 149].

steady Double Gyre

For the 2D case, we can easily model a velocity field that mimics this behavior. We regard the stream function $\psi(x, y)$ with its partial derivatives in x- and y-direction

$$\psi(x, y) = \sin(\pi x) \sin(\pi y)\,, \tag{3.1}$$

$$\frac{\partial \psi}{\partial x} = \pi \cos(\pi x) \sin(\pi y)\,,$$

$$\frac{\partial \psi}{\partial y} = \pi \sin(\pi x) \cos(\pi y)\,.$$

From the partial derivatives, we can derive a 2D velocity field $\mathbf{v}(x, y)$ as a co-gradient field

$$\mathbf{v}(x, y) = \begin{pmatrix} u(x, y) \\ v(x, y) \end{pmatrix}\,,$$

$$u(x, y) = -\frac{\partial \psi}{\partial y} = -\pi \sin(\pi x) \cos(\pi y)\,,$$

$$v(x, y) = \quad \frac{\partial \psi}{\partial x} = \quad \pi \cos(\pi x) \sin(\pi y)\,.$$

Figure 3.1 illustrates the stream function using height-mapping, the velocity field as an arrow plot, the corresponding LIC image, and a streamline plot over the domain $D = [0, 2] \times [0, 1]$. The data set is periodic and repeats in x- and y-direction alternating as original and mirrored version.

unsteady Double Gyre

Shadden et al. [145] used this model as a starting point to study material transport over hyperbolic LCS. The LCS were extracted as height ridges from FTLE scalar fields. Shadden wanted to know precisely: *"Do LCS (in FTLE fields) represent invariant manifolds?"* [144]. As explained in Section 2.5, LCS persist over time, even in *time-dependent* flows. Therefore, Shadden et al. extended Equation (3.1) to a *time-dependent* version that is known as the Double Gyre

$$\psi(x, y, t) = A \sin(\pi f(x, t) \sin(\pi y)\,, \tag{3.2}$$

$$f(x, t) = a(t)x^2 + b(t)x\,, \tag{3.3}$$

$$a(t) = \epsilon \sin(\omega t)\,, \tag{3.4}$$

$$b(t) = 1 - 2\epsilon \sin(\omega t)\,. \tag{3.5}$$

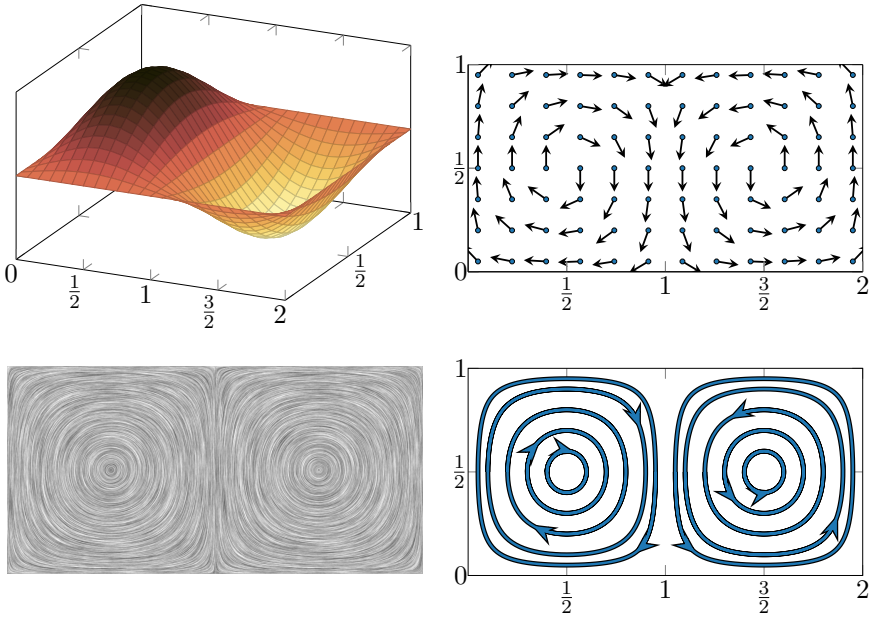


Figure 3.1: STEADY DOUBLE GYRE. Top-left: Stream function as height field. Top-right: Arrow plot of the velocity field. Bottom-left: LIC image. Bottom-right: Oriented streamlines.

We derive the 2D velocity field $\mathbf{v}(x, y, t)$ again as a co-gradient field and get

$$\mathbf{v}(x,y,t) = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \end{pmatrix},$$

$$u(x,y,t) = -\frac{\partial \psi}{\partial y} = -\pi A \sin(\pi f(x,t)) \cos(\pi y),$$

$$v(x,y,t) = \frac{\partial \psi}{\partial x} = \pi A \cos(\pi f(x,t)) \sin(\pi y) \frac{\mathrm{d}f}{\mathrm{d}x},$$

$$\frac{\mathrm{d}f}{\mathrm{d}x} = 2a(t)x + b(t).$$

explanation & parameters

Figure 3.2 shows visualizations for different states of the velocity field. The velocity field is closed in the rectangular domain $D = [0, 2] \times [0, 1]$, i.e., no particles leave or enter this area when advected by the field. Equations (3.2) to (3.5) have some parameters that control the appearance of the data set. Parameter A is a factor that controls the magnitude of the velocity vectors. The function still describes two gyres that rotate in counter-oriented directions. A vertical line, initially located at $x = 1$, separates the gyres. The velocity field contains critical points: static saddles in each corner and moving saddles on top and bottom of the separation line. In the center of each gyre, we can locate another critical point – a center. If $\epsilon \neq 0$, the field becomes time-dependent. The separation line between the gyres oscillates in the x-direction, i.e., the gyres are alternately compressed and stretched with ongoing time t. ω controls this motion's frequency, and ϵ determines its magnitude

to the left and the right. Shadden et al. suggest to use the following parameters:

$$A = 0.1, \quad \omega = 2\pi/10, \quad \epsilon = \frac{1}{4},$$

this is also the parametrization we use for this thesis.

The FTLE fields in Figure 3.2 show an emerging ridge that corresponds to an LCS in the domain center. With increasing integration time, the ridge structures become more complex. Particles placed on either side of the ridge diverge when they reach a separating point at the bottom of the domain. This separating point does not coincide with the saddle point. There exists no closed-form solution that describes the exact movement of this point. Neither exists one that describes the LCS, i.e., we have to determine it numerically [145, 180].

3.2 RELEVANCE FOR THE VISUALIZATION COMMUNITY

After Shadden et al. published their excellent paper [145] and the accompanying online tutorial [144], it became prevalent in the Scientific Visualization community. The paper was cited over 1000 times. Therefore, also, the Double Gyre became well known and gained interest. Today it is one of the most used benchmarks. In the following, we list some typical examples that use the Double Gyre to show its importance and versatility.

Germer et al. [56] study guaranteed material separation along LCS. Schindler et al. [141] evaluate new ridge concepts for LCS. Barakat et al. [6] evaluate adaptive refinement strategies for the flow map as well as extraction of ridge geometries. Kuhn et al. [86] present a new method to detect LCS by tracking timeline cells. Machado et al. [99] extract LCS via space-time bifurcation lines. Günther et al. [61] present a method that computes high-quality FTLE ridges and ridge surfaces based on Monte-Carlo path tracing. Hummel et al. [78] analyze an error estimation for Lagrangian representations of flows. Hofmann et al. [77] extract recirculation surfaces with the dependent vectors operator in 2D. Froyland et al. [44] study mixing enhancement and consider almost-invariant manifolds [43]. The Double Gyre comes with a set of parameters that influence the behavior of the flow. Although most works use the original parameters given by Shadden et al., other versions exist. For instance, Sadlo et al. [130] extend the domain to receive a flow field consisting of four rotating gyres. In Chapter 7, we present a 3D version of the data set that we published in [177]. Wolligandt et al. [180] present a modified version that is hardly distinguishable from the original version. They can provide ground truth closed-form hyperbolic trajectories and use them to compute LCS.

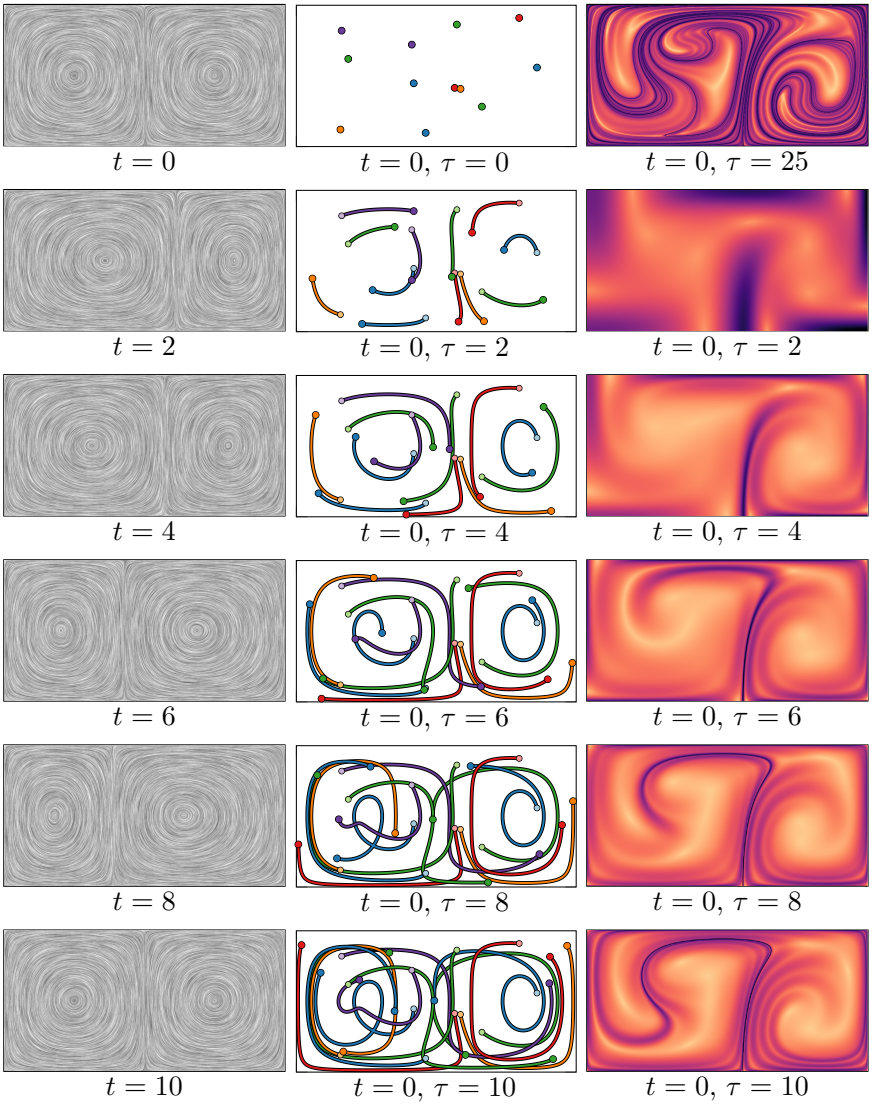


Figure 3.2: DOUBLE GYRE VISUALIZATIONS. Left column: LIC images for different times t. Center column: Pathlines for selected positions and varying integration time τ. Colored markers indicate the seeding position (light) and the current position (saturated). Right column: FTLE fields for varying integration times τ. Top-right: The FTLE field for $\tau = 0$ is not defined. Therefore, we show a field for a long integration time with complex ridge structures.

3.3 RELEVANCE FOR THIS THESIS

All the listed examples give only a small glimpse into the importance of the Double Gyre. It has a simplistic and understandable analytic form and covers different interesting flow patterns from simple to challenging. In the remainder of this thesis we will see the Double Gyre, on several occasions. We pick it up as a benchmark for every presented technique.

"If it works for the Double Gyre, the other data sets should work as well." (Holger Theisel, September 2015)

Part II

FLOW MAP PROCESSING

4

FLOW MAP MODIFICATION BY SPACE-TIME DEFORMATION

This chapter is based on the publication:

T. Wilde, C. Rössl, H. Theisel
Flow Map Processing by Space-Time Deformation
Advances in Visual Computing
(Proc. 15th International Symposium, ISVC 2020, Part I) 2020

In Sections 2.2 and 2.4, we introduced the two common ways to represent flows: time-dependent velocity fields and flow maps. Both have their advantages and disadvantages.

On the one side, we have velocity fields. They are of research interest for several decades now. Velocity fields are lower-dimensional, easy to store, and simple to handle. Besides many techniques to visualize them, also established approaches for modeling, construction, deformation, simplification, and compression exist. We refer to these approaches as *vector field processing* and consider them an inevitable part of Flow Visualization. The main drawback of velocity fields is the expensive numerical integration we must perform to obtain particle trajectories.

On the other side, there are flow maps. They have gained increased interest in recent years. Once we computed the flow map, they enjoy a direct encoding of particle trajectories. We retrieve pathlines by an evaluation – an expensive numerical integration is not necessary anymore. The ultimate goal is to provide methods for *flow map processing* comparable to vector field processing. Techniques to directly process the data, such as smoothing, deformation, construction, and simplification, are desirable. Nevertheless, considering the flow map is still a challenging task. The reason for this is fourfold:

challenges

1. Their computation is expensive – it involves massive numerical particle integrations beforehand.
2. They are higher dimensional and need more storage space in comparison to velocity fields.
3. The inherent properties lead to a complex inner structure with highly connected data.
4. Their gradient shows exponential behavior with increasing integration time, making many standard techniques unusable.

fundamental problem

When we work with flow maps, we have to face another fundamental problem: the result we get after a small modification is, in general, not a flow map anymore but leaves the space of valid flow maps. The cause of this is that a slight change would immediately destroy the inherent properties we introduced in Section 2.4.2. We explain this by example in Section 4.2. This is a substantial difference when we compare flow maps and velocity fields. Nevertheless, the possibility to modify flow maps is a prerequisite for further flow map processing techniques.

chapter overview

In this chapter, we present the first approach that tackles this problem. We introduce a concept for modification techniques that keep the properties intact. Section 4.1 gives a short overview of related work. In Section 4.2, we present a function that remaps locations in space – this is the basis for a flow map modification. We show how to restrict this function to a specific area in space-time, making local changes possible. Even if the deformation area is local, the impact it has is global. We show which regions must be adapted to keep the global flow map properties. Furthermore, we present a technique to lower the computational effort regarding discretized data. Hence, we achieve a performance that is suitable for interactive real-time modification. In Section 4.3, we give details about the implementation of a demonstrator we use to explore and deform flow maps. Finally, in Section 4.4, we demonstrate our technique with two deformation tools we apply to the Double Gyre.

4.1 RELATED WORK

This section will give a short overview of previous research regarding vector fields and flow maps. We already discussed visualization techniques in Sections 2.2 and 2.4. Therefore, we will focus on design and processing techniques.

Vector Field Design & Application

vector field design

Before we can work with vector fields, the first step is to create them. Velocity fields are often the results of flow simulations. Nevertheless, the manual creation of velocity fields is necessary, e.g., for artistic purposes or specific benchmarks. We need automated or user-controlled techniques that allow the design of vector fields.

Theisel [158] presents an early approach to designing arbitrary 2D steady vector fields based on a set of control polygons. With the help of non-overlapping control polygons, we can describe a topological skeleton. It contains critical points and separatrices and is used to compute the vector field. Weinkauf et al. [173] extend this approach to 3D. The topological skeleton is designed interactively by placing

the control polygons. From this, a piecewise linear vector field with the same topology is computed. The presented approach also contains a visualization technique for higher-order critical points in 3D vector fields.

Zhang et al. [183, 184] offer a system that aims for fast and interactive manipulation of vector fields on surfaces. The core idea is to place, move, and remove singularities on surfaces to modify vector fields. The authors demonstrate artistic purposes like texture synthesis on 3D objects or painterly renderings of images. Chen et al. [25] contribute an element-based framework for designing time-varying vector fields. The framework includes several design metaphors like streamlines, pathlines, and singularity paths. Based on a key-frame-like design, a spatio-temporal optimization is employed to compute vector fields. The paper presents different application examples containing animated images and animated textures on 3D objects.

vector field application

Vector fields are a powerful tool to solve a variety of problems. For example, many different approaches to texture synthesis on 3D surfaces rely on vector fields, compare, e.g., Praun et al. [124], Turk [162], or Wei and Levoy [168].

Funck et al. [45, 46, 165] present a new method for the deformation of mesh-based 3D objects. The authors design deformation shapes that contain vector fields. The shapes are placed on a 3D mesh, and the mesh vertices are transformed based on the shape's vector field. We formulate our approach for flow map modification similarly.

For more information regarding vector field synthesis, design, and processing, we refer to the survey by Vaxman et al. [164].

Flow Map Processing & Application

flow map processing

Most visualization and simulation approaches use sampled representations of data sets [156]. During the sampling and reconstruction process, an error is induced into the data. Hummel et al. [78] present a method to estimate the upper error when reconstructing pathlines from sampled short-time flow maps.

flow map application

There exist different meaningful flow properties and features we derive from flow maps. Many state-of-the-art Flow Visualization techniques are based on advection in velocity fields [19, 91, 92, 135]. All of the methods that rely on integrating massless particles use at least a part of the flow map.

Another prominent example is the FTLE we introduced in Section 2.5. Besides the FTLE, the finite-space Lyapunov exponent (FSLE) [4, 131] is used in the Flow Visualization community. The FSLE rates the time that passes until two closely seeded particles separated for a fixed amount in space.

Weinkauf and Theisel [170] derive streakline fields from the spatio-temporal flow map gradient. Streakline fields are vector fields that contain streaklines as tangent curves. Hence we can use standard techniques to integrate streaklines.

Hlawatsch et al. [76] present an approach to speed up the process of computing particle trajectories. They use parallel computation and concatenation of short-time flow maps.

Besides the faster computation of flow maps, we are not aware of other techniques that directly address flow map processing.

4.2 FLOW MAP MODIFICATION

In this section, we introduce the theoretical foundation for a flow map modification approach. A modification must preserve all defining properties. Due to the highly connected data, it is impossible to change the flow map in a small area without breaking the global flow map properties. This is a fundamental problem, and we explain it in more detail by example.

example complex structure

Flow maps explicitly encode pathlines, i.e., a modification changes pathlines. When we think about deforming a single pathline, it finally becomes clear why the flow map has a complex inner structure. Imagine we seed a single particle in an unsteady flow at position $\mathbf{x}_0$ and seeding time $t_0 = 0$. We observe its movement for 10 discrete time steps, i.e., the integration time $\tau_0 \in \{0, 1, ..., 9\}$. The flow map $\phi_t^\tau(\mathbf{x})$ yields this specific particle's positions for any observed integration time, i.e., $\phi(\mathbf{x}_0, 0, \tau_0)$ with $0 \leq \tau_0 \leq 9$. We now imagine placing another particle at the same pathline but a little bit later for $t_1 = 1$. We denote the seeding position for this second particle as $\mathbf{x}_1$. Because $\mathbf{x}_0$ is a position on the same pathline, we know $\phi(\mathbf{x}_0, 0, 1) = \phi(\mathbf{x}_1, 1, 0) = \mathbf{x}_1$. From our initial observation, we can derive further flow map entries for the particular second particle. We can immediately set all $\phi(\mathbf{x}_1, 1, \tau_1)$ with $-1 \leq \tau_1 \leq 8$. More flow map entries become clear if we increase the seeding time and pick other positions on the pathline. Hence, from

the observation, we can derive the flow map values for the following (t, τ)-combinations:

$$
\begin{aligned}
t_0 = 0 \quad &\text{with} \quad \tau_0 \in \{\pm 0, +1, +2, +3, +4, +5^*, +6, +7, +8, +9\}, \\
t_1 = 1 \quad &\text{with} \quad \tau_1 \in \{-1, \pm 0, +1, +2, +3, +4^*, +5, +6, +7, +8\}, \\
t_2 = 2 \quad &\text{with} \quad \tau_2 \in \{-2, -1, \pm 0, +1, +2, +3^*, +4, +5, +6, +7\}, \\
t_3 = 3 \quad &\text{with} \quad \tau_3 \in \{-3, -2, -1, \pm 0, +1, +2^*, +3, +4, +5, +6\}, \\
t_4 = 4 \quad &\text{with} \quad \tau_4 \in \{-4, -3, -2, -1, \pm 0, +1^*, +2, +3, +4, +5\}, \\
t_5 = 5 \quad &\text{with} \quad \tau_5 \in \{-5, -4, -3, -2, -1, \pm 0^*, +1, +2, +3, +4\}, \\
t_6 = 6 \quad &\text{with} \quad \tau_6 \in \{-6, -5, -4, -3, -2, -1^*, \pm 0, +1, +2, +3\}, \\
t_7 = 7 \quad &\text{with} \quad \tau_7 \in \{-7, -6, -5, -4, -3, -2^*, -1, \pm 0, +1, +2\}, \\
t_8 = 8 \quad &\text{with} \quad \tau_8 \in \{-8, -7, -6, -5, -4, -3^*, -2, -1, \pm 0, +1\}, \\
t_9 = 9 \quad &\text{with} \quad \tau_9 \in \{-9, -8, -7, -6, -5, -4^*, -3, -2, -1, \pm 0\}.
\end{aligned}
$$

From the seeding of the initial particle and the observation of 9 further discrete time steps, we already determine 90 flow map entries that describe this particle's movement.

We now want to modify the pathline at half the integration time, i.e., we change the value for $\phi(\mathbf{x}_0, t_0, 5)$. This implies that we must adapt all flow map entries that describe the same location on this pathline. We marked these with a '$*$' in the (t, τ)-list. Because the flow map is continuous in space-time, we also have to adapt the entries next to the modified location. This includes locations on the initial pathline but also other pathlines in the spatial neighborhood. Furthermore, we have to make sure that all inverse mappings are still correct.

With this simple example, we can show that a small change always has a global impact in the space-time domain. This makes modifications of flow maps a challenging problem. In the following, we propose an approach to solving this task, divided into three steps:

1. Define a space deformation function to modify flow map entries.
2. Define a local area in space-time to perform a modification.
3. Globally identify all influenced regions and adapt them.

In this section we use pathline plots to visualize the deformation process; in the results section, we also use FTLE fields. To clarify ideas and concepts, we will illustrate time-dependent flows in the 1D and 2D spatial domain.

4.2.1 *Definition of a Space Deformation*

Given a flow map ϕ, we want to apply a local modification in space-time such that the new map $\widetilde{\phi}$ is a flow map again. The conversion must change the mapping of positions in space at selected domain locations

in space-time. An example may be to pick a pathline and locally change its shape. Due to continuity reasons, a modification must also affect the region near the modified pathline, i.e., we also have to change the shape of adjacent pathlines. Still, at a certain distance in space-time, all remaining pathlines are untouched.

definition space deformation

We model such behavior by defining a space deformation $\mathbf{y}(\mathbf{x}, t)$ as:

$$\mathbf{y} : D \times T \to \mathbb{R}^n \,,$$
$$\mathbf{y}(\mathbf{x}, t) = (\widetilde{\mathbf{x}}, t) \,.$$

properties space deformation

D denotes the spatial domain and T the temporal domain. The function $\mathbf{y}$ maps a point $(\mathbf{x}, t)$ in space-time to the new point $(\widetilde{\mathbf{x}}, t)$, i.e., the spatial location is changed, but the time step stays the same. We demand $\mathbf{y}$ to be:

- local – it affects only a particular limited region of the space-time domain unless $\mathbf{y}$ is the identity;
- continuous – $\mathbf{y}$ is smooth and at least $\mathcal{C}^1$-continuous, i.e., it contains no sudden jumps or gaps;
- invertible – the spatial gradient $\nabla\mathbf{y}$ does have full rank, i.e., the inverse map $\mathbf{y}^{-1}$ is well-defined.

We compute the new flow map $\widetilde{\phi}_t^\tau(\mathbf{x})$ from $\phi_t^\tau(\mathbf{x})$ and $\mathbf{y}$ by:

$$\widetilde{\phi}_t^\tau(\mathbf{x}) = \mathbf{y}(\phi(\mathbf{y}^{-1}(\mathbf{x}, t), t, \tau), t + \tau) \,. \tag{4.1}$$

Note that $\widetilde{\phi}$ is a flow map as well; we give proof for this in Appendix A. We explain, why we demand $\mathbf{y}$ to be local, continuous, and invertible.

local

The transformation from ϕ to $\widetilde{\phi}$ should be local in space-double-time, i.e., position $\mathbf{x}$, time t, and integration time τ. This way, we can define a particular area in which $\mathbf{y}$ should perform a modification. The rest of the flow map remains mostly untouched. In areas where no modification is performed, $\widetilde{\phi}_t^\tau(\mathbf{x}) = \phi_t^\tau(\mathbf{x})$. We achieve this when $\mathbf{y}(\mathbf{x}, t)$ is the identity

$$\mathbf{x} = \mathbf{y}(\mathbf{x}, t) \,,$$
$$\phi(\mathbf{x}, t, \tau) = \mathbf{y}(\phi(\mathbf{y}^{-1}(\mathbf{x}, t), t, \tau), t + \tau) \,.$$

Nevertheless, global transformations are still possible by extending the modified area to the whole domain.

continuous

In general, the flow map describes a smooth and continuous function. If $\mathbf{y} \in \mathcal{C}^1$, we can keep this property even after a modification.

invertible

The space deformation $\mathbf{y}(\mathbf{x}, t)$ also has to be invertible. The reason for this is *not* to undo the transformation $\widetilde{\phi} \to \phi$. Instead, it keeps $\widetilde{\phi}$ continuous in space. Figure 4.2 illustrates this. Imagine we pick a part of a pathline and move it a little bit. The pathline would move to a new position and leave a gap. Another pathline fills this gap by moving up. To identify this gap-filling pathline, we must invert $\mathbf{y}$.

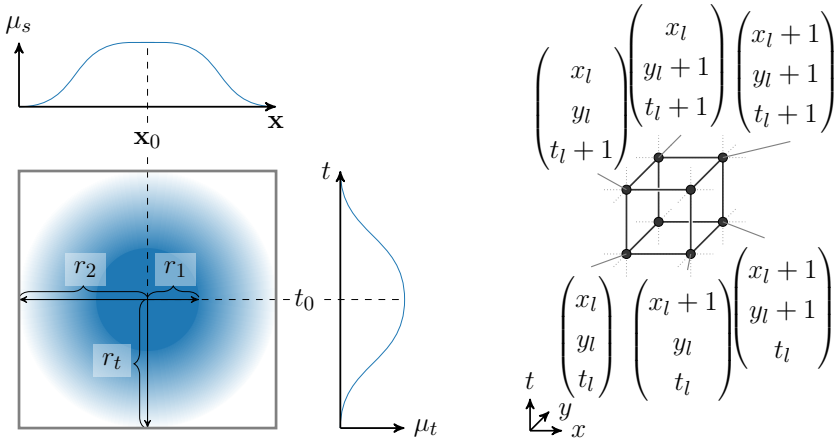


Figure 4.1: MODIFICATION SHAPE & LOOKUP CELLS. Left: Illustration of the modification shape S. Right: Illustration of a lookup-cell in (x, y, t)-space.

4.2.2 *Definition of a Modification Area*

For an interactive manipulation of flow maps, we follow a metaphor using interactive shape modeling by deformations [12, 45] and adapt this to flow map modeling. We define the shape S in space-time, with n dimensions in space and one dimension in time; therefore, $S \subset D \times T$. S describes an area of effect in which the modification is performed. It is placed in the space-time domain, e.g., interactively at a user-defined location.

modification area shape

We divide the shape into three parts:

1. An inner part, with a rigid modification.
2. An outer part, where no transformation takes place.
3. A middle part, where an energy minimization computes the deformation to ensure continuity and other useful properties.

Figure 4.1 (left) illustrates the shape S for the 1D case, i.e., one spatial dimension and one temporal dimension.

We define a local space deformation $\mathbf{y}$ by setting the following values for the modification shape S: $\mathbf{x}_c, t_c, r_1, r_2, r_t, \mathbf{x}_d$. Here $(\mathbf{x}_c, t_c)$ denotes the center position of S in space-time, and $\mathbf{x}_d$ denotes a deformation vector. Furthermore, we demand

$$0 \leq r_1 < r_2\,, \qquad 0 < r_t\,, \qquad \|\mathbf{x}_d\| < r_2 - r_1\,.$$

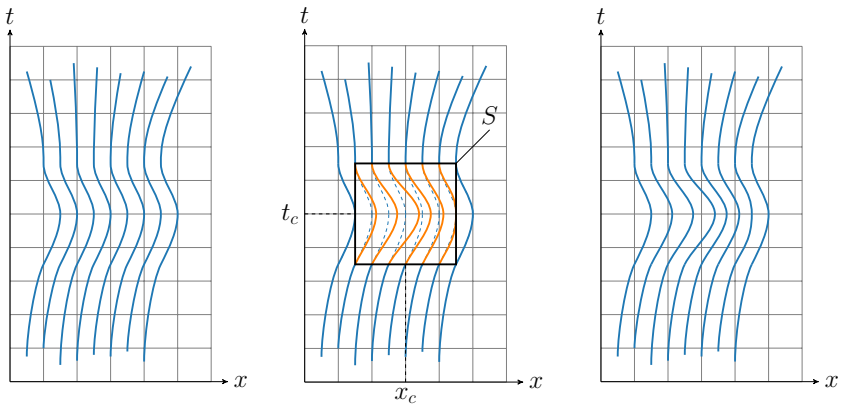


Figure 4.2: ILLUSTRATION OF A MODIFICATION IN 1D. Left: Original pathlines. Center: Process of modification, it is strongest in the center of S. Right: Smooth modified pathlines.

Then we define a space deformation as:

$$\mathbf{y}(\mathbf{x}, t) = \mathbf{x} + \mu \mathbf{x}_d \tag{4.2}$$

$$\mu = \mu_s \mu_t \,, \tag{4.3}$$
$$\mu_s = (s_{\mathbf{x}}^3 + 3s_{\mathbf{x}}^2(1 - s_{\mathbf{x}}))\,, \quad \mu_t = (s_t^3 + 3s_t^2(1 - s_t))$$

$$r = \|\mathbf{x} - \mathbf{x}_c\| \tag{4.4}$$

$$s_{\mathbf{x}} = \begin{cases} 1 & r < r_1 \\ \frac{r - r_2}{r_1 - r_2} & r_1 \leq r \leq r_2 \\ 0 & r > r_2 \end{cases} \qquad s_t = \begin{cases} 1 - \frac{|t - t_c|}{r_t} & |t - t_c| < r_t \\ 0 & \text{else} \end{cases} . \tag{4.5}$$

μ is a scaling factor for the deformation, defined in the dependency of S. μ is 0 next to the border (no modification) and 1 at the center (full modification) of S. This definition of the modification shape S leads to a smooth transition from original to modified areas in the $\widetilde{\phi}$ domain.

Figure 4.2 gives an example of a modification of pathlines for the 1D case. We place S at a specific location $(\mathbf{x}_c, t_c)$, e.g., guided by the user. Figure 4.2 (left) shows pathlines extracted from the flow map. Note that they are defined in space-time and do *not* necessarily start or end at the same time t. Figure 4.2 (center) illustrates a modification in the form of a translation in the positive x-direction. We show the modified pathline pieces in orange; they stay inside S. Outside of the modification shape, $\mathbf{y}$ maps to the identity, i.e., $\widetilde{\phi} = \phi$. This also holds for the particular case $\tau = 0$. Figure 4.2 (right) shows the final result.

Note that by construction, $\mathbf{y}$ is $\mathcal{C}^1$-continuous and invertible. However, the inverse $\mathbf{y}^{-1}$ does not have a simple formula. Because of this, we numerically precompute $\mathbf{y}^{-1}$ on a uniform grid in a space-time box with the same size as S. We use this precomputed box as a lookup table to estimate $\mathbf{y}^{-1}$.

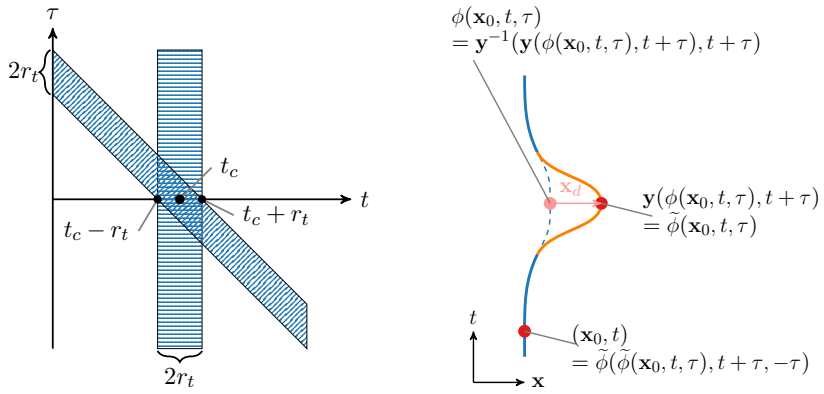


Figure 4.3: AFFECTED AREAS. Left: Areas with (t, τ)-pairs in which $\widetilde{\phi}$ has to be updated. Right: The effect of $\mathbf{y}$ for two locations on a single pathline.

4.2.3 *Local Modification & Global Adaption*

regions for global adaption

Having defined the space deformation $\mathbf{y}$, we have to update the flow map $\widetilde{\phi}$ accordingly by applying Equation (4.1). Since $\mathbf{y}$ is local, i.e., only in a specific area, it is not the identity, $\widetilde{\phi}$ is local as well. To ensure the flow map properties, we need to identify the parts we have to change for a local modification process. We classify these parts into two groups:

1. Parts mapping from *inside* S to its *outside*, i.e. $\phi(\mathbf{x}, t, \tau) \notin S$ with $(\mathbf{x}, t) \in S$.
2. Parts mapping to the *inside* of S, i.e., $\phi(\mathbf{x}, t, \tau) \in S$.

Keep in mind that we define S in space-time. Therefore the terms *inside* and *outside* also refer to space-time. For the spatial part, we have to check if a location $\mathbf{x}$ or its mapping $\phi(\mathbf{x}, t, \tau)$ belongs to S.

Regarding the double-time dimensions, only specific (t, τ)-combinations are relevant. However, we must compute the new flow map $\widetilde{\phi}$ only for (t, τ)-pairs with $(t_c - r_t) \leq (t + \tau) \leq (t_c + r_t)$. Figure 4.3 (left) illustrates the relevant parts in the (t, τ)-domain. Regarding a single pathline from the example given in Figure 4.2, the flow map needs to be adapted for all locations on this pathline, where a (t, τ)-pair maps into the modification area S. Figure 4.3 (right) illustrates this adaption for two locations on a pathline. If we modify $\phi(\mathbf{x}, t, \tau)$, then we also have to adapt $\phi(\phi(\mathbf{x}, t, \tau), t + \tau, -\tau)$.

4.2.4 *Discretization of the Flow Map*

discretization in space-time

ϕ is defined as a continuous map describing particle trajectories in flows from the Lagrangian reference frame. As described in Section 2.6, we discretize it in space and time. We discretize the domain over a regular grid. For the discretization scheme, we require the parameters

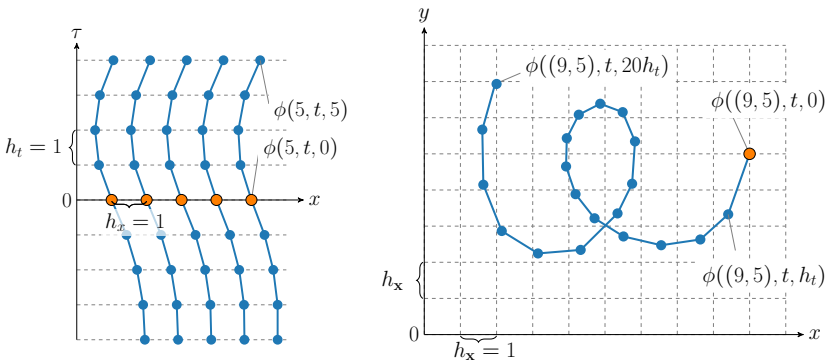


Figure 4.4: FLOW MAP DISCRETIZATION IN 1D & 2D. Left: Sampled pathlines for different x, fixed t, and consecutive τ in 1D. Right: A pathline for a fixed $\mathbf{x}$, fixed t, and consecutive τ in 2D. Orange markers denote $(\mathbf{x}, t)$-coordinates of samples. Blue markers indicate τ-coordinates and mapping destinations.

$h_\mathbf{x}$ and h_t. They denote the constant distance between two samples in the spatial dimensions, e.g., (x, y) and the temporal dimensions t and τ. This discretization of the domain results in a single slice for each (t, τ)-combination containing the spatial grid samples. We perform numerical integration in the underlying vector field for each discrete $(\mathbf{x}, t, \tau)$-sample. The corresponding trajectory endpoint is stored for this sample and builds a single entry in the discretized flow map. This way, we get an approximation of the flow map in each dimension. Figure 4.4 gives an example of the 1D and 2D cases. The access to the flow map entries is given by $(\mathbf{x}, t, \tau)$-coordinates for each sample. Furthermore, each sample gets a unique ID. We need this ID to speed up the retrieval of affected parts – we explain this in Section 4.2.6. Flow map values between discrete samples are obtained by multi linear interpolation of the corresponding neighboring samples. Pathlines for a fixed $(\mathbf{x}, t, 0)$-position can be extracted by evaluating the entries for consecutive τ-coordinates.

sample ID for look-up cell

4.2.5 *Modification of the Discrete Flow Map*

global adaption areas

We have to adapt the discrete samples' entries when we modify the discrete flow map. As stated out, we classify the involved samples into two groups:

1. Samples located *inside* the modification area S mapping *outside*.
2. Samples mapping *into* the modification area S.

The samples covered by the modification area S belong to the first group. They are easy to identify by their coordinates. The samples belonging to the second group are not so easily identifiable. Potentially we have to check all samples with a suitable (t, τ)-combination (see Figure 4.3) for each modification. But in general, only a small group

of samples maps into S. To cope with this problem, we introduce a different data structure.

4.2.6 *Lookup-Cells for Fast Sample Retrieval*

Based on the flow map sampling, we divide the discretization into *lookup-cells*, defined by $(\mathbf{x}, t)$-coordinates, i.e., we omit one temporal coordinate τ. The adjacent samples of a location in $(\mathbf{x}, t)$-space build the vertices of the surrounding lookup-cell. For the 2D case, a lookup-cell C at the sample $(\mathbf{x}_l, t_l)$ is given by the following coordinates:

$$\begin{aligned} C_{(\mathbf{x}_l,t_l)} = \{&(x_l, y_l, t_l), && (x_l+1, y_l, t_l),\\ &(x_l, y_l+1, t_l), && (x_l+1, y_l+1, t_l),\\ &(x_l, y_l, t_l+1), && (x_l+1, y_l, t_l+1),\\ &(x_l, y_l+1, t_l+1), && (x_l+1, y_l+1, t_l+1)\}\,. \end{aligned}$$

Figure 4.1 (right) gives an illustration. For other dimensions, vertex-coordinates follow the same scheme. $2^{(n+1)}$ neighboring samples make up a cell. Vice versa, each discrete sample is part of up to $2^{(n+1)}$ adjacent cells. Each lookup-cell $C_{(\mathbf{x}_l,t_l)}$ holds the IDs of samples mapping into its volume in discretized space-time. The set P of these samples for a single cell is given by:

$$P_{C_{(\mathbf{x}_c,t_c)}} = \{(\mathbf{x}, t, \tau) \mid \phi(\mathbf{x}, t, \tau) \in C_{(\mathbf{x}_l,t_l)}\} \;\text{ with }\; t_l \leq (t+\tau) \leq t_l + 1\,.$$

This way, fast retrieval of all parts that need to be adapted is possible. The temporal domain of the lookup-cells covers the whole (t, τ)-space, i.e., $T_t \times T_\tau$.

4.3 IMPLEMENTATION

After precomputing the complete flow map in a particular resolution, we can reformulate standard visualization approaches using pathline integration as a simple array lookup. Based on this, we implemented an interactive exploration tool for unsteady 2D flows. Since the pathline starting at $(\mathbf{x}, t)$ in a parametrization of τ is just $\phi(\mathbf{x}, t, \tau)$, we can compute every point by a quadrilinear interpolation of the sampled flow map. Note that this way, the pathline's accuracy does *not* depend on the pathline's sampling density: even for a sparse sampling, the points on the pathline are correct (up to the accuracy of the initial flow map sampling). This is contrary to the numerical integration of pathlines, where the integration's step size influences the integration error. Besides the exploration via pathlines, we also implemented a visualization via FTLE fields. By a straightforward lookup, we can compute the FTLE field in the discretized flow map. Given t and τ, we determine the FTLE field at the same spatial resolution as the sampled flow map. For this, we compute the flow map derivatives by central differences.

pathlines plot

FTLE plot

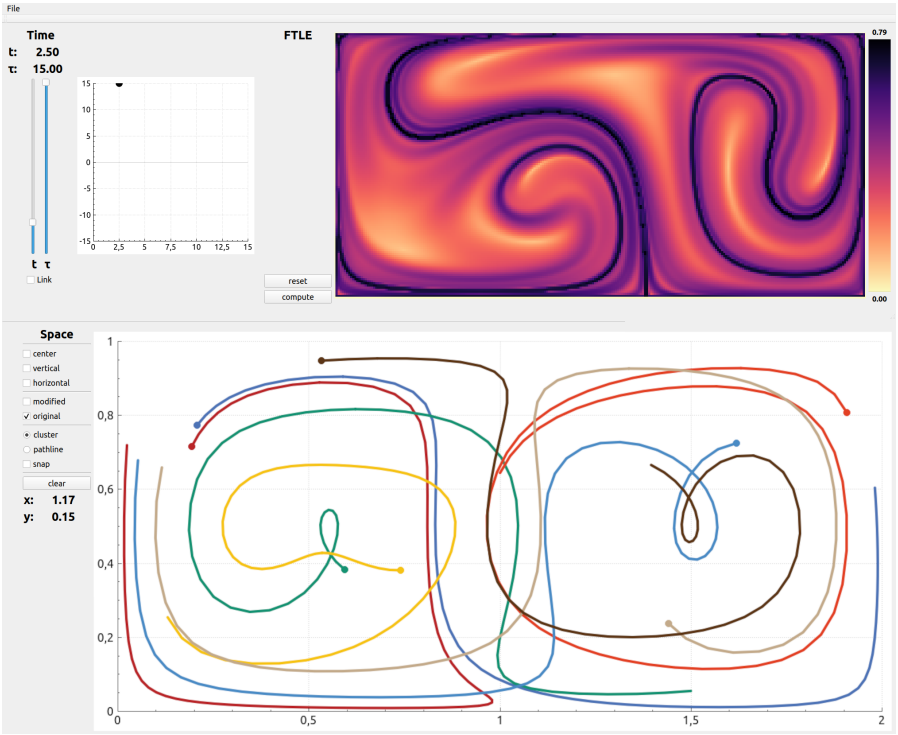


Figure 4.5: The Pathline Explorer. Top-left: (t, τ)-space selection. Top-right: FTLE visualization. Bottom: Random pathlines originate at the circular markers.

Figure 4.5 shows a snapshot of our interactive viewer. The values t and τ are modified either by sliders or by interactive selection in the (t, τ)-space. According to this, we update the pathlines and the FTLE fields in interactive real-time. Furthermore, we implemented a small toolset based on the concepts presented in this chapter to demonstrate interactive flow map modification. The tools include a translation and a rotation of pathline positions.

4.4 Results & Discussion

test system

We show the results as plots of selected pathlines and FTLE fields. Timings were taken on an Intel Core i7-8700K CPU running at 3.70GHz with 32GB of RAM. We executed all algorithms on a single core. The data set we used is the Double Gyre.

data set

Figure 4.5 shows pathlines and the FTLE field for $t = 2.5$ and $\tau = 15$. To obtain the discretized flow map, we sampled the domain with a resolution of 300×150 for $(x, y) \in [0.0, 2.0] \times [0.0, 1.0]$. We discretized the temporal dimension $t \in [0.0, 20.0]$ by 100 samples and $\tau \in [-20.0, 20.0]$ by 200 samples. This sums up to $9.0 \cdot 10^8$ samples, leading to a memory usage of 13.7GB for the discretized flow map.

precomputation

In a precomputation step, we performed a second-order Runge-Kutta integration for each sample; this took about 150 minutes. In another

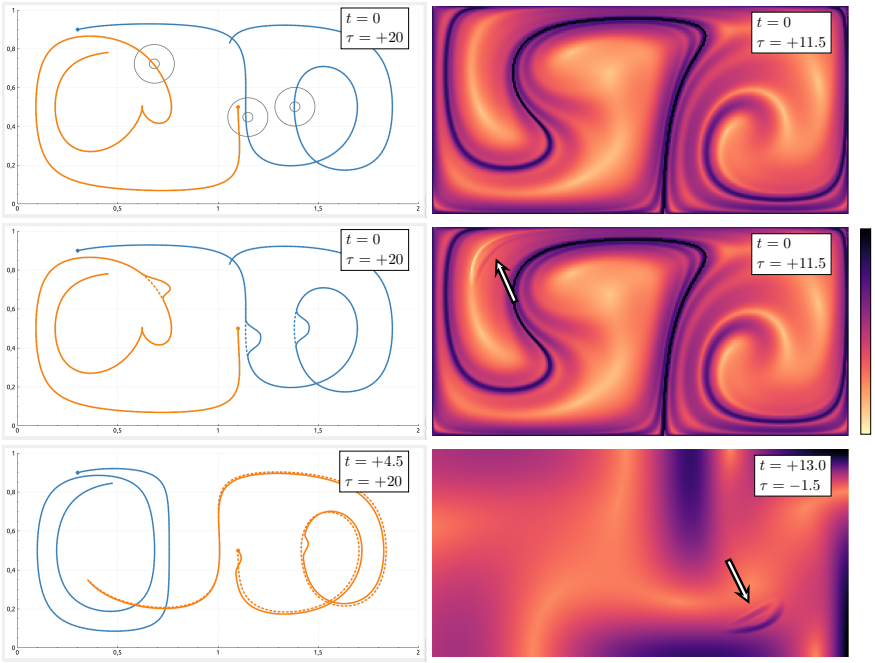


Figure 4.6: TRANSLATION TOOL. Top-left: Original pathlines before deformation. The circular markers indicate the modification areas (in space). Center-left: Pathlines passing the deformed areas (in space-time) are translated to the right. Dashed lines indicate original pathlines. Bottom-left: The modifications have a global effect and influence other start and integration times. The orange pathline starts in the modified area. It is adapted to another pathline. Top-right: Original FTLE field. Center-right: FTLE field after deformation. Bottom-right: The changes have a global effect and influence other times. Both arrows mark parts influenced by the rightmost modification area, i.e., pathlines starting in these areas (in space-time) pass the rightmost modification area (in space-time).

precomputation step, we computed the lookup-cells; this took approximately 64 seconds and cost an additional 3.5GB of RAM. We show results for two different interactive modification tools.

4.4.1 *Translation Tool*

The first tool performs a translation inside the modification area. The user can interactively 'drag & drop' parts of the flow map to the desired positions. Translation direction and amount are controlled with the mouse cursor. For the experiments, we placed the modification area at three different locations and performed a translation in the positive x-direction, with $\mathbf{x}_d = (0.075, 0.0)$.

We illustrate the results in Figure 4.6. The top row shows pathlines and FTLE field before deformation. Top-left also shows the modification tool at the three test locations. Two concentric black circles mark inner and outer regions of the modification area S at each location.

	Location $(x, y, t + \tau)$	Modified Entries	Time in ms
Translation	(1.15, 0.45, 4.5)	$1.58 \cdot 10^6$	278
	(1.38, 0.50, 11.5)	$1.32 \cdot 10^6$	208
	(0.70, 0.70, 9.5)	$1.84 \cdot 10^6$	302
		Σ $4.74 \cdot 10^6$	Σ 788
		$\varnothing$ $1.58 \cdot 10^6$	$\varnothing$ 262
Twist	(0.70, 0.25, 1.55)	$1.69 \cdot 10^6$	309
	(0.46, 0.62, 5.00)	$1.39 \cdot 10^6$	203
	(0.30, 0.75, 10.00)	$1.72 \cdot 10^6$	302
	(1.00, 0.80, 13.35)	$1.78 \cdot 10^6$	315
	(1.50, 0.20, 17.15)	$1.73 \cdot 10^6$	275
		Σ $8.31 \cdot 10^6$	Σ 1405
		$\varnothing$ $1.67 \cdot 10^6$	$\varnothing$ 281

Table 4.1: Modification Info & Timings. The table lists the modification areas' locations, the number of modified entries, and the time needed for the modification.

pathlines & FTLE

Figure 4.6 center-left shows a blue and an orange pathline, which pass the altered areas. Dashed lines indicate the original trajectories before deformation. Figure 4.6 center-right shows an FTLE field that differs from the original one. Pathlines starting in the region marked with a white arrow at $t = 0$ will pass the modified area after $\tau = 11.5$. The global influence of the modification is evident in the images in the bottom row. Pathlines starting in the modified regions regarding space-time get modified seed conditions. Therefore, they are 'replaced' by another path. The global influence is also visible in the FTLE field. Please note, we decided to show only two carefully selected pathlines to avoid visual clutter. The selected ones and the shown (t, τ)-combinations give a good impression of the global influence of (pseudo-)local modifications.

timings & samples

The modification area is defined in space-time and had an extent of 0.2 in space and 1.0 in time. On average, $1.58 \cdot 10^6$ entries had to be modified, which took 0.262 seconds on average. After the process, $4.74 \cdot 10^6$ flow map entries were modified in total, which corresponds to approximately 0.52% of all flow map entries. Table 4.1 lists the center locations of the modified areas, the times needed for modification and adaption, and the total numbers of changed samples for each location.

4.4.2 *Twist Tool*

The second tool performs a twist of entries around the center of the modification area. The twist describes a 90° counterclockwise rotation. Figure 4.7 illustrates the modification and its effect. Figure 4.7 top-left

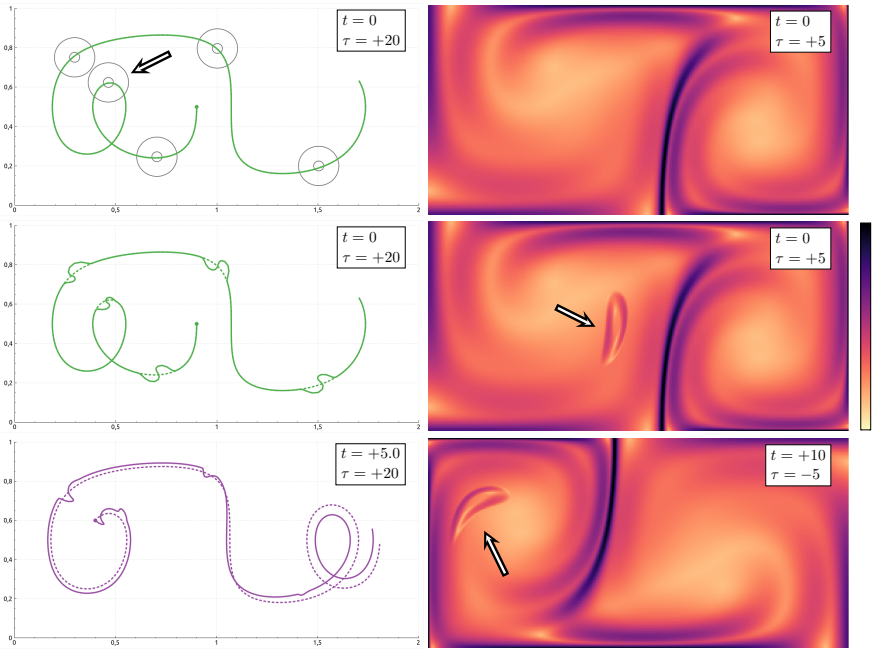


Figure 4.7: TWIST TOOL. Top-left: Original pathline before deformation, a small solid circle marks the seed location. The circular markers indicate the modification areas (in space). Center-left: The pathline passes the modification areas (in space-time) and is twisted counterclockwise. Bottom-left: The deformation has a global effect. The purple pathline is seeded in a modified area and gets another trajectory. Top-right: Original FTLE field. Center-right: FTLE field after modification. Bottom-right: The global effect of the deformation is visible at other times. The arrows mark parts that were influenced (center-right and bottom-right) and the modification area causing the effect (top-left).

pathlines

shows a single green pathline before modification and five locations where deformation was performed. The center-left image shows the pathline after the modification as a solid line. The dashed parts indicate the original path. The purple pathline in the bottom-left image was seeded at a later time $t = 5$ in a modified region. Hence, it gets modified seed conditions and is adapted to another path. Later on, it passes different deformed areas.

FTLE

The right images in Figure 4.7 show the FTLE field before (top) and after the deformation process. The center-right and center-bottom FTLE field highlight regions that indicate the deformation caused by the area marked in the top-left image.

timings & samples

All modification areas have the same extent of 0.2 in space and 1.0 in time. On average, $1.67 \cdot 10^6$ entries were modified in each modification area which took 0.281 seconds on average. After the whole process, $8.31 \cdot 10^6$ flow map entries were modified in total by the twist tool. This corresponds to approximately 1% of all flow map entries. Table 4.1

lists the locations of the modified areas, the times for modification and adaption, and the numbers of modified flow map entries.

4.5 CONCLUSION & FUTURE RESEARCH

conclusion

Flow map processing opens up new possibilities for efficient and effective Flow Visualization – we expect it to be of more interest in future research. An essential prerequisite for processing techniques is to perform modifications without leaving the space of valid flow maps. In this chapter, we introduced an approach for modifications of discretely sampled flow maps. We define a local area in space-time. In this area, we perform a change based on a space deformation. Afterward, we identify all regions that must be adapted to keep global flow map properties. Although we stick to the 1D and 2D cases to explain, all presented concepts are valid for any dimension. We demonstrate the applicability of our approach with two simple tools for pathline deformation.

future research

Future research could be related to better tools and processing methods. We divide these into two groups: user-guided and automatic techniques.

For user-guided techniques, better interaction is necessary. We modify the flow map of a 2D unsteady flow by a tool in a 3D space-time domain. Effectively the modification takes place in a 4D domain (x, y, t, τ). Tools that better fit the modification's 4D effect are desirable. Due to the data's high connectivity and the necessary global adaption, the modification can lead to unexpected results in (space-time) areas currently not visible. For instance, pathlines seeded in a modified area are adapted to entirely different trajectories. This behavior is hard to predict. Therefore, better usability can only be achieved by better techniques to visualize the global behavior of flow maps in space-double-time. The mentioned interaction problems will be more significant for 3D flows and need to be addressed in future research.

We could think of transferring approaches from vector field processing to flow maps for the automated techniques, e.g., smoothing, simplification, or compression. The method presented in this chapter gives a solution for a local transformation with global adaption. A straightforward approach for global operation is to perform a local change consecutively for each location. Even though our presented approach makes this possible while keeping the defining properties, it is not clear if the result is the desired one. Each of the small local operations makes a global adaption necessary. The result could technically be a correct flow map, but the data may have nothing to do with the initial flow anymore. Imagine a flow map that maps only to the $\mathbf{0}$ vector. It fulfills all defining flow map properties but obviously does not reflect a flow. Therefore, global operations need to be addressed in the future.

5

DRIFT FIELDS FOR FLOW MAP PROCESSING

This chapter is based on the paper:

T. Wilde, S. Wolligandt, C. Rössl, H. Theisel
Drift Fields for Flow Map Processing
submitted to EuroVis Conference 2021

In Chapter 4, we introduced an approach to modify flow maps while maintaining flow map properties. We see this as an essential step towards further flow map processing techniques. Nevertheless, when compared to velocity fields, the flow map still keeps its drawbacks. It is high-dimensional, needs more storage, is computationally expensive, and has a complex structure.

drift field properties

This chapter proposes an alternative representation of flows that can be considered a compromise between velocity fields and flow maps. We call them *drift fields*, and they have the following properties:

- drift fields have the same size and dimensionality as velocity fields, i.e., are lower-dimensional than flow maps;
- particle trajectories are directly encoded – to get pathlines, no numerical integration, but a local search is necessary;
- drift fields are closed under perturbation – a small modification of a drift field is still a drift field. Hence, we can use them to modify pathlines.

chapter overview

Sections 5.1 to 5.3 give a formal introduction to drift fields, their properties, and their relation to other structures. In Section 5.4, we show that we can find a drift field for each flow. Section 5.5 presents a numerical approach to compute drift fields. An algorithm to compute pathlines is presented in Section 5.6, followed by a method to modify drift fields in Section 5.7. We apply our approach to different flow data sets and present the results in Section 5.8. In Section 5.9, we discuss drift fields and give an outlook on future research in Section 5.10.

5.1 DEFINITION

definition drift field

Given are a 2D time-dependent velocity field $\mathbf{v}(\mathbf{x}, t)$ and the corresponding flow map $\phi_t^\tau(\mathbf{x})$. We define the drift field $\mathbf{d}(\mathbf{x}, t)$ as a vector-valued function, describing the same flow as:

$$\mathbf{d} : D \times T_t \to \mathbb{R}^2 \,,$$
$$\mathbf{d}(\mathbf{x}, t) = \begin{pmatrix} a(\mathbf{x}, t) \\ b(\mathbf{x}, t) \end{pmatrix} .$$

spatial gradient & time derivative

The drift field $\mathbf{d}$ covers the same domain $D \times T_t$ in space and time as the velocity field $\mathbf{v}$. It assigns each $(\mathbf{x}, t)$-combination two particular scalar values. We obtain these values by evaluating the scalar functions $a(\mathbf{x}, t)$ and $b(\mathbf{x}, t)$. Because we define these functions over the spatial domain D, each of them forms a 2D time-dependent scalar field. When we consider each time step individually and fix the value for t, we can imagine $a(\mathbf{x}, t)$ and $b(\mathbf{x}, t)$ as scalar fields constant for this particular time step. Compare, e.g., Figure 5.3 to get an impression. Furthermore, we describe its spatial gradient $\nabla\mathbf{d}$ and its time derivative $\mathbf{d}_t$ as:

$$\nabla\mathbf{d} = \Big(\nabla a, \nabla b\Big)^{\mathrm{T}} = \begin{pmatrix} a_x & a_y \\ b_x & b_y \end{pmatrix}, \qquad \mathbf{d}_t = \begin{pmatrix} a_t \\ b_t \end{pmatrix} .$$

We demand the following two conditions to hold:

$$\det(\nabla\mathbf{d}) > 0 \,, \tag{5.1}$$
$$\mathbf{d}(\mathbf{x}, t) = \mathbf{d}(\phi(\mathbf{x}, t, \tau), t + \tau) \,, \tag{5.2}$$

for all $\mathbf{x} \in D$, $\{t, (t + \tau)\} \in T_t$ and $\tau \in T_\tau$. Equation (5.1) makes sure the gradients ∇a and ∇b are not parallel and well defined, i.e., they do not vanish. Equation (5.2) requires the drift field to contain the same values for $a(\mathbf{x}, t)$ and $b(\mathbf{x}, t)$ along a pathline, which encodes the actual flow behavior.

5.2 PROPERTIES

convert $\mathbf{d}$ *to* $\mathbf{v}$ $\mathbf{d}$ *to* ϕ

If a drift field $\mathbf{d}$ exists for a given flow, it has a remarkable property – it directly encodes the corresponding velocity field $\mathbf{v}$ and the flow map ϕ. We can obtain them by simple local operations, i.e., without any numerical integration

$$\mathbf{v} = -(\nabla\mathbf{d})^{-1}\mathbf{d}_t \,, \tag{5.3}$$
$$\phi(\mathbf{x}, t, \tau) = \mathbf{d}^{-1}(\mathbf{d}(\mathbf{x}, t), t + \tau) \,. \tag{5.4}$$

We will explain this in more detail. Equation (5.3) comes from the following observation: if we consider $\mathbf{d}$ as a time-dependent vector field, we can regard $\mathbf{v}$ as the feature flow field [157] of $\mathbf{d}$. The defining property

of this feature flow field is precisely the condition we demand in Equation (5.2). The corresponding components of $\mathbf{d}$ remain constant when we move along a pathline of $\mathbf{v}$. Transforming the closed-form of feature flow fields to this setup gives Equation (5.3) (compare Equation 26 in [62]).

invertibility

To interpret Equation (5.4), we first keep in mind that Equation (5.1) makes sure $\mathbf{d}$ is invertible. Thus, we can use the inverse $\mathbf{d}^{-1}$ and get

$$\mathbf{d}^{-1}(\mathbf{d}(\mathbf{x}_1, t_1), t_2) = \mathbf{x}_2\,,$$

for $\{\mathbf{x}_1, \mathbf{x}_2\} \in D$ and $\{t_1, t_2\} \in T_t$. Then, we have a look at the right-hand side of Equation (5.4). When we pick a single time slice from the drift field by fixing t_0, we get:

$$\mathbf{d}(\mathbf{x}, t_0) = \begin{pmatrix} a(\mathbf{x}, t_0) \\ b(\mathbf{x}, t_0) \end{pmatrix} = \begin{pmatrix} a_0 \\ b_0 \end{pmatrix}.$$

$\mathbf{d}$ is constant along pathlines

That means $\mathbf{d}$ assigns a unique $(a_0, b_0)^{\mathrm{T}}$ to each location $\mathbf{x} \in D$ for the fixed time t_0. If we now vary the time by the factor τ, there must be a unique location $\mathbf{z}$ in the corresponding time slice that fulfills

$$\mathbf{d}(\mathbf{z}, t_0 + \tau) = \begin{pmatrix} a(\mathbf{z}, t_0 + \tau) \\ b(\mathbf{z}, t_0 + \tau) \end{pmatrix} = \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \begin{pmatrix} a(\mathbf{x}, t_0) \\ b(\mathbf{x}, t_0) \end{pmatrix} = \mathbf{d}(\mathbf{x}, t_0)\,, \tag{5.5}$$

with $\mathbf{z} \in D$, $\{t_0, t_0 + \tau\} \in T_t$, and $\tau \in T_\tau$. In other words, each (a_0, b_0)-combination that we get from the drift field appears exactly once in each time slice but at different locations. We can generalize Equation (5.5) to cover all reasonable times for t_0 and get:

$$\mathbf{d}(\mathbf{z}, t + \tau) = \mathbf{d}(\mathbf{x}, t)\,. \tag{5.6}$$

When we put this all together, we gain Equation (5.4) that states that this unique location $\mathbf{z}$ at the time $(t + \tau)$ corresponds to the flow map $\phi_t^\tau(\mathbf{x})$. Moreover, we can obtain the flow map by a local search in $\mathbf{d}$. We will discuss this process in Section 5.6.

5.3 RELATIONS

In this section we will point out the relations of drift fields to velocity fields, flow maps, feature flow fields, and stream functions.

5.3.1 *Relation to Velocity Fields & Flow Maps*

The overview in Figure 5.1 summarizes the complexity of a velocity field $\mathbf{v}$, a drift field $\mathbf{d}$, and a flow map ϕ, respectively, their relations and how to convert them. It shows that $\mathbf{d}$ contains the *'best of both worlds'*:

- **d** is a low-dimensional flow representation like **v** (in fact, it is lower-dimensional than ϕ);
- **d** directly encodes **v** and ϕ, i.e., we can convert without any numerical integration.

relation $\mathbf{v}, \phi, \mathbf{d}$

However, this holds only if we can show that there is always a valid drift field for a flow. Furthermore, a stable numerical algorithm to extract **d** is necessary.

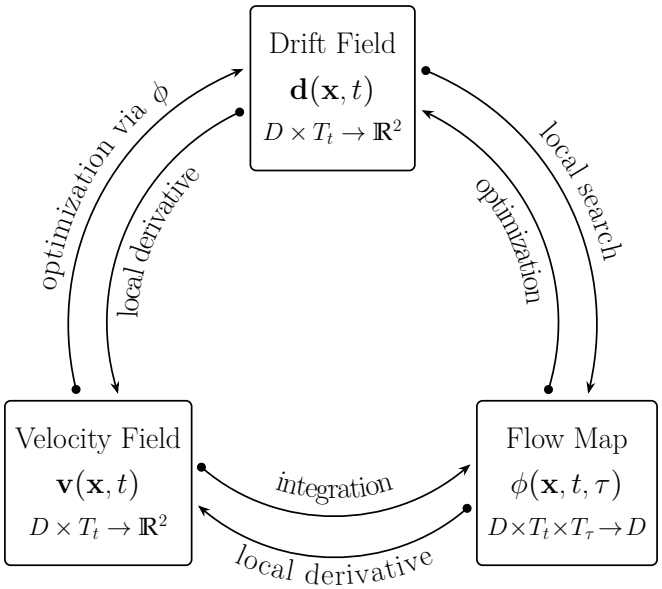


Figure 5.1: Relations & Complexity. Overview of the relations of velocity field **v**, drift field **d**, and flow map ϕ and how to convert them.

5.3.2 *Relation to Feature Flow Fields*

relation to FFF

Given a velocity field **v**, the feature flow field, as introduced by Theisel [157], connects all points in space-time with the same value of **v**. Drift fields describe precisely the opposite. We do not search for the feature flow field but for an unknown field **d**, such that **v** is the feature flow field of **d**.

5.3.3 *Relation to Stream Functions*

relation to stream function

Stream functions describe 2D steady flows such that the velocity is its co-gradient field – compare Section 3.1 Figure 3.1. They allow considering infinitely long integrations of streamlines by a local search. However, we are restricted to divergence-free flows. Contrary, drift fields capture pathlines of arbitrary velocity fields but only for a finite integration time.

5.4 EXISTENCE & UNIQUENESS OF DRIFT FIELDS

When we take a flow map $\phi_t^\tau(\mathbf{x})$, it is easy to find a drift field $\mathbf{d}(\mathbf{x}, t)$ describing the same flow. The only thing we have to do is to select a t_0 and set:

$$\mathbf{d}(\mathbf{x}, t) = \phi(\mathbf{x}, t, t_0 - t). \tag{5.7}$$

good drift fields

Equation (5.7) shows the existence of a drift field for any flow. However, it also reveals that $\mathbf{d}$ is not unique – every choice of t_0 generally gives another drift field representing the same flow. The goal is to find a *'good'* drift field. A drift field is *'good'* if it allows a fast, accurate, and robust extraction of pathlines. We expect this to be the case if the drift field's gradient $\nabla\mathbf{d}$ is smooth and behaves well in every location – compare Equation (5.1). For an arbitrarily selected t_0, $\nabla\mathbf{d}$ is likely to have strong gradients. These strong gradients originate from the flow map gradients $\nabla\phi$ that grow exponentially for an increasing integration time τ (see, e.g., Kuhn et al. [87]). Therefore, we determine the drift field $\mathbf{d}$ by a numerical optimization approach.

5.5 COMPUTING DRIFT FIELDS

We formulate the construction of $\mathbf{d}$ as a linear numerical optimization problem over t_0. For the computation, we utilize the spatial drift field gradient $\nabla\mathbf{d}$ and the spatial flow map gradient $\nabla\phi$. We divide the computation into three steps.

5.5.1 *Step 1 – Create an Initial Field for One Time Slice*

In Section 5.1, we explained that a drift field assigns a unique (a, b)-combination to each location when considering a single slice for a fixed time t. When we observe a particular (a, b)-pair in other time slices, we can find its corresponding locations along a single pathline in space-time. This property directly follows Equations (5.2) to (5.6) and applies to all valid (a, b)-combinations. Hence, the degrees of freedom available to create a drift field reduce enormously. We only have to provide a single initial time slice for the drift field. The pathlines of the flow define the remaining time slices automatically.

Therefore, we aim for a *'good'* initialization for a specific time t_0. As stated earlier, *'good'* means that $\nabla\mathbf{d}$ is smooth, e.g., it should have no sudden jumps. It should behave well, e.g., it is not too strong. It should fulfill Equation (5.1), i.e., it is invertible and does not vanish. Also, we want to avoid numerical problems when determining $\nabla\mathbf{d}^{-1}$.

initialization

If we consider all of these conditions, a rotation matrix would be a good solution to describe $\nabla\mathbf{d}$. Therefore, if we pick a single time t_0, a perfect solution for $\nabla\mathbf{d}$ would be the identity matrix $\mathbf{I}$ at every point. We achieve $\mathbf{I}$ at every point, when initializing $\mathbf{d}(\mathbf{x}, t_0)$ with linearly increasing scalar

values in the x-direction for $a(\mathbf{x}, t_0)$ and the y-direction for $b(\mathbf{x}, t_0)$. Thus, $\nabla a = (1, 0)^{\mathrm{T}}$ and $\nabla b = (0, 1)^{\mathrm{T}}$ yield $\nabla \mathbf{d} = \mathbf{I}$. Figure 5.3 in the results section shows an example for $t_0 = 6$. Two orthogonal linearly increasing scalar fields for $a(\mathbf{x}, t_0)$ and $b(\mathbf{x}, t_0)$ proved to be the best solution for the initialization even when considering further time steps.

5.5.2 *Step 2 – Determine Best Time t_0 for Initialization*

The perfect drift field would result in $\nabla \mathbf{d}$ describing a rotation matrix for all $(\mathbf{x}, t)$ with smooth transitions over time. Only a minority of flows fulfills this, but we can try to come as close as possible.

$\nabla \mathbf{d}$ under advection

For this, we need a way to evaluate the development of $\nabla \mathbf{d}$ along a pathline. We utilize $\nabla \phi$ and $\nabla \mathbf{d}$ and consider

$$\nabla \mathbf{d}(\mathbf{x}, t) = \nabla \mathbf{d}(\phi_{t_0}^{\tau}(\mathbf{x}), t_0 + \tau) \nabla \phi_{t_0}^{\tau}(\mathbf{x}) \,, \tag{5.8}$$

$$\widehat{\nabla \mathbf{d}} := \nabla \mathbf{d} \nabla \phi^{-1} = \nabla \mathbf{d}(\phi, t_0 + \tau) \,, \tag{5.9}$$

$$\mathbf{H}(\mathbf{x}, \tau) := \widehat{\nabla \mathbf{d}} \widehat{\nabla \mathbf{d}}^{\mathrm{T}} = \begin{pmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{pmatrix} , \tag{5.10}$$

for $\mathbf{x} \in D$, $\{t_0, (t_0 + \tau)\} \in T_t$, and $\tau \in T_\tau$. Equation (5.8) directly follows from Equation (5.2) and the chain rule for derivatives. It describes the demanded deformation of $\nabla \mathbf{d}$ along the pathline trough $(\mathbf{x}, t_0)$. The goal is to keep $\widehat{\nabla \mathbf{d}}$ as close as possible to a rotation matrix. This is the case when ∇a and ∇b have unit-length and are orthogonal, i.e.,

$$\nabla a^{\mathrm{T}} \nabla a = 1 \,, \quad \nabla b^{\mathrm{T}} \nabla b = 1 \,, \quad \nabla a^{\mathrm{T}} \nabla b = 0 \,.$$

We express this behavior with Equation (5.10). The entries in $\mathbf{H}(\mathbf{x}, \tau)$ correspond to:

$$h_{11} = \nabla a^{\mathrm{T}} \nabla a \,, \quad h_{22} = \nabla b^{\mathrm{T}} \nabla b \,, \quad h_{12} = \nabla a^{\mathrm{T}} \nabla b \,.$$

optimization

We give proof for this in Appendix B. Therefore, we can use

$$\| \mathbf{H}(\mathbf{x}, \tau) - \mathbf{I} \|_{Fr} \,, \tag{5.11}$$

to measure how close $\nabla \mathbf{d}$ corresponds to a rotation matrix under advection. We construct a cost function that helps us find an optimal drift field based on the initialization gained in Step 1. We evaluate Equation (5.11) for all locations $\mathbf{x}$ and all possible integration times τ and minimize

$$\int_D \int_\tau \ln(\| \mathbf{H}(\mathbf{x}, \tau) - \mathbf{I} \|_{Fr}^2 + 1) \, d\tau \, d\mathbf{x} \to \min \,,$$

for t_0. Instead of using Equation (5.11) in the integral directly, we use the *ln* function to cope with the flow map gradient's exponential growth for long integration times. Thus we reduce the search for a *'good'* drift field to a 1D optimization problem.

5.5.3 *Step 3 – Compute the Remaining Time Slices*

transfer by advection

In Step 1, we compute a good initial slice for a fixed time step. In Step 2, we determine the best time t_0 we should use for the initialization. We can calculate all remaining values by advection along the pathlines. We transfer the values of the initialized time slice by:

$$\mathbf{d}(\mathbf{x}, t) = \mathbf{d}(\phi(\mathbf{x}, t, t_0 - t), t_0)\,,$$

for all $\mathbf{x} \in D$ and $\{t, t_0\} \in T_t$. We use the flow map to 'walk' along the pathline from the current position at time t to the corresponding location at time t_0 and transmit the initial value. The result is a fully initialized drift field $\mathbf{d}(\mathbf{x}, t)$ that serves as a Lagrangian flow representation comparable to the flow map. We use this for further processing.

5.6 COMPUTING PATHLINES FROM DRIFT FIELDS

Computing pathlines is one of the main tasks we need to solve when working with flows. We perform the computation of pathlines from drift fields in 5 steps. The process is illustrated in Figures 5.2 and 5.3.

5.6.1 *Step 1 – Selection of Seeding Location*

select seeding position

Drift fields represent time-dependent flows. The trajectory a massless particle describes depends on seeding position $\mathbf{x}_s$ and seeding time t_s that we have to select. We must choose this location such that $\mathbf{x}_s \in D$ and $t_s \in T_t$. This step is common for velocity fields, flow maps, and drift fields.

5.6.2 *Step 2 – Determine* (a_s, b_s)

determine iso values

As described in Sections 5.1 and 5.2, the drift field assigns an (a, b)-combination to each location, so it does for the start location $\mathbf{d}(\mathbf{x}_s, t_s) = (a_s, b_s)$. This specific (a_s, b_s) describes the position of the particle at each time.

5.6.3 *Step 3 – Selection of Destination Time* t_d

select integration time

Besides selecting the seeding position and time, we have to choose the particle's movement period. For this, we select the destination time t_d at which we want to determine the particle's position. This step is comparable to the definition of τ for flow maps. We can select any value for t_d as long as $t_d \in T_t$. This is done in contrast to velocity fields, where we have to integrate with continuous time steps.

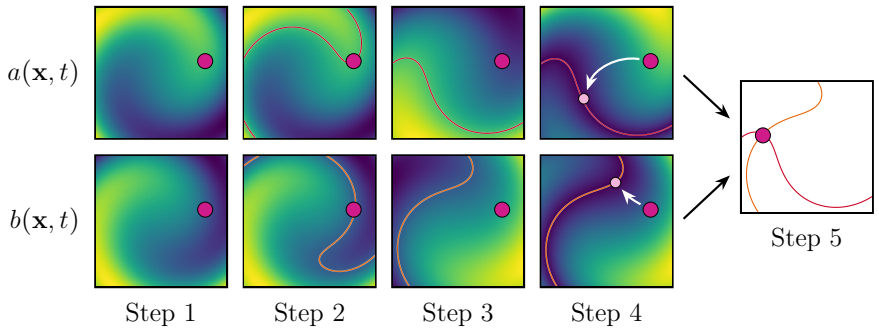


Figure 5.2: PATHLINES FROM DRIFT FIELDS. Illustration for the computation of particle trajectories from drift fields. The isoline intersections define the locations of the particle.

5.6.4 *Step 4 – Searching for Isolines at Time t_d*

local search

The definition of drift fields implies that an (a, b)-combination exists exactly once in each time slice. Nevertheless, we consider a specific value for a in $a(\mathbf{x}, t)$ and a specific value for b in $b(\mathbf{x}, t)$. Each forms an isoline in the particular scalar field. To determine the seeding particle's position at time t_d, we have to find the corresponding isolines in $a(\mathbf{x}, t_d)$ and $b(\mathbf{x}, t_d)$. Therefore, we construct two new scalar fields

$$\tilde{a}(\mathbf{x}, t_d) = |a(\mathbf{x}, t_d) - a_s| \,,$$
$$\tilde{b}(\mathbf{x}, t_d) = |b(\mathbf{x}, t_d) - b_s| \,.$$

We then perform a gradient descent in $\tilde{a}(\mathbf{x}, t_d)$ and $\tilde{b}(\mathbf{x}, t_d)$, starting at $\mathbf{x}_s$ until we reach a 0. At this location, we found the corresponding isoline and derive it. For more information on isoline extraction we refer to Hanisch [71], and Lorensen and Cline [98].

Please note, we do not have to compute $\tilde{a}(\mathbf{x}, t_d)$ and $\tilde{b}(\mathbf{x}, t_d)$ for the whole spatial domain D. We can restrict the search to the current location's immediate vicinity.

5.6.5 *Step 5 – Intersecting Isolines*

determine destination position

The last step we have to take is to determine the crossing between the found isolines. The intersection of isolines is also a solved problem and easy to handle. Because drift fields describe closed flows, this intersection must exist. The intersection corresponds to the location of the seeded particle at time t_d.

Drift fields allow the extraction of particle locations for arbitrary destination times t_d. The difference $(t_d - t_s)$ corresponds to the integration time τ. The effort to compute particle locations depends on the drift field's resolution and is the same in each time step. It does not increase for longer τ. This is a significant difference to computing pathlines in velocity fields, where the computation time linearly increases for longer

τ. If we compute a full pathline with consecutive locations, we can even speed up the extraction of pathlines from drift fields. Therefore, we extrapolate the movement of the particle in the previous time step. We use the extrapolated location as the starting point for Step 4. This generally reduces the search area for isolines and leads to faster intersections. Furthermore, when we seed particles at neighboring locations, they likely end up in the same region. Hence, we can start our search directly in the destination area next to the last isoline intersection.

5.7 MODIFYING DRIFT FIELDS

Drift fields are just like flow maps a Lagrangian flow representation, i.e., they encode massless particles' trajectories. Chapter 4 introduced a modification technique for flow maps. For the modification of drift fields, we utilize the same concept of space-time deformation without the necessity of the complex global adaption.

space-time deformation

We define a deformation function $\mathbf{y}$ and shape S in space-time and place it in the drift field domain. A smooth and continuous transformation is performed inside S, with a $\mathcal{C}^1$-continuous transition from original to modified parts. We adopt Equations (4.2) to (4.5) from Section 4.2 and slightly change Equation (4.2) to:

$$\mathbf{y}(\mathbf{x}, t) = \mathbf{x} - \mu \mathbf{x}_d \,. \tag{5.12}$$

modified drift field

$\mathbf{x}$ describes the current location that is modified, $\mathbf{x}_d$ a deformation vector, and μ is a scaling factor. By applying $\mathbf{y}$ to the original drift field $\mathbf{d}$, we yield the modified drift field $\widetilde{\mathbf{d}}$ as:

$$\widetilde{\mathbf{d}}(\mathbf{x}, t) = \mathbf{d}(\mathbf{y}(\mathbf{x}, t), t) \,.$$

smudge tool

The effect we achieve is comparable to the well-known *smudge tool* that is provided by many painting applications. We can imagine to use this tool simultaneously on the scalar fields $a(\mathbf{x}, t_c)$ and $b(\mathbf{x}, t_c)$ for a fixed time t_c. The drift field would get smudged along the tool's path, with smooth transitions between original and modified areas. The essential difference is that the smudge tool for drift fields also changes neighboring time slices.

This way, we obtain an easy and intuitive deformation of drift fields. We can directly manipulate pathlines and still keep the Lagrangian flow representation consistent. Because we can transform drift fields back into flow maps, our approach implicitly serves as a tool for manipulating flow maps. Unlike the method from Chapter 4, we do not have to *'repair'* the data because we preserve inherent field properties. Furthermore, the new technique is easy to understand and predict because it only transforms a 3D space and mimics a well-known tool.

5.8 RESULTS

data sets

We compute drift fields for four different data sets:

- Rotating Flow, an analytic benchmark ,
- Double Gyre, an analytic benchmark ,
- Cavity Flow, a simulated velocity field ,
- Piped Cylinders Flow, a simulated velocity field .

Each data set will be described in more detail in this section's remainder. All data sets are provided as velocity fields either as an analytic formula or sampled on a regular spatio-temporal grid from a simulation.

data preparation

We sampled a discrete flow map from each velocity field by generating pathlines starting from points in a regular grid. For numerical integration, we applied a fourth-order Runge-Kutta scheme. We chose the sampling for analytic flows such that the time derivative estimated from discrete differences is sufficiently smooth. We sampled the Cavity Flow at the resolution of the discrete input data. Due to the flow map's memory consumption, we had to reduce the sampling for the Piped Cylinders Flow.

experiment explanation

We computed the drift field from the flow map and then reconstructed the original flow map to quantify the accuracy of the drift field model. For the reconstruction, we sampled the drift field as described in Section 5.6 for all times (t, τ) and all initial positions $\mathbf{x}_0$ as required by the given spatio-temporal grid, i.e., we reconstructed the full sampled flow map from the drift field. We measured the mean error as absolute difference per grid point averaged over all (t, τ)-combinations. As the error encodes a distance, we visualize the error relative to the size of the spatial grid cells. For each data set we

- summarize *timings* for flow map computation by integration of the velocity field vs. local search in the drift field, finding optimal t_0 with respect to our cost function, and computing the drift field from the flow map via advection from t_0;
- compare the *memory consumption* for drift field and flow map;
- show the drift field for selected *time slices*;
- study the *error* from the reconstruction of the original flow map from the drift field;
- compare *pathlines* from drift fields vs. flow maps.

Besides, we study the modification of a Double Gyre drift field.

	Time in Seconds					Memory in MB		
	$\mathbf{v} \to \phi$	t_0	$\phi \to \mathbf{d}$	$\mathbf{d} \to \phi$	$\bar{e}_r$	$\mathbf{v}$	$\mathbf{d}$	ϕ
Rotating Flow								
$[0, 3]$	80	260	0.098	46	$< 0.5\%$	<1	10	576
$[0, 6]$	149	269	0.094	46	$< 0.5\%$	<1	10	576
$[0, 12]$	278	267	0.097	46	$\approx 1.0\%$	<1	10	576
Double Gyre								
$[0, 10]$	1808	560	0.038	279	$\approx 4.0\%$	32	32	3200
Cavity Flow								
$[1, 4]$	183	38	0.038	6	$< 3.5\%$	5	5	138
Piped Cylinders Flow								
$[2, 4]$	228	352	0.024	108	$< 0.4\%$	25	25	2464

Table 5.1: TIMINGS & MEMORY CONSUMPTION. The table lists the timings for different operations, relative mean errors for flow map reconstruction, and memory consumptions.

Table 5.1 summarizes timings and relative mean errors. All timings are given in seconds. All experiments were run on a computer equipped with an Intel Core i7-8700K CPU at 3.70GHz with 32GB RAM. All algorithms were executed on a single core. The column labels are as follows:

- $\mathbf{v} \to \phi$ denotes the time for computing the full flow map ϕ from the velocity field $\mathbf{v}$;
- t_0 denotes the time for finding the optimal initialization times for the drift field;
- $\phi \to \mathbf{d}$ denotes the time for the construction of the drift field $\mathbf{d}$ from the flow map ϕ;
- $\mathbf{d} \to \phi$ denotes the time for the reconstruction of the full flow map ϕ from the drift field $\mathbf{d}$;
- $\bar{e}_r$ displays the mean error for all samples relative to the sampling grid cell size;
- $\mathbf{v}, \mathbf{d}, \phi$ list the memory consumptions for the velocity field, the drift field, and the flow map.

color-maps

We visualize selected time slices of drift fields with the *Viridis* color map [150] (compare Figure 5.3), and the relative mean error for flow map reconstruction with the *Lajolla* color map [28] (compare Figure 5.4). Please note, we do *not* list absolute values for drift field color maps because they are irrelevant. In Section 5.4, we explained that only the spatial gradient $\nabla\mathbf{d}$ influences a drift field. It is even possible to shift or scale all values without changing the quality or the result of the drift field.

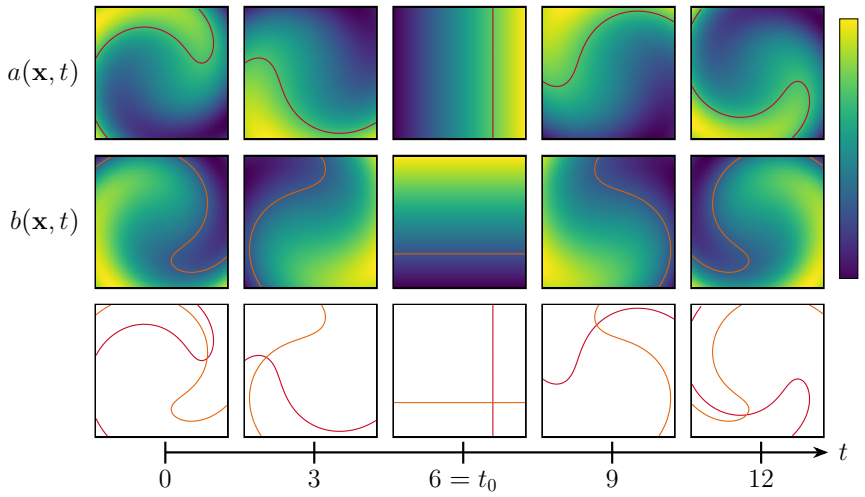


Figure 5.3: ROTATING FLOW DRIFT FIELD. Top and center row: Selected time slices of the drift field components $\mathbf{d} = (a, b)$. Times as indicated on the t-axis. The drift field was initialized at $t_0 = 6$ and advected to $t = 0$ and $t = 12$. Bottom row: Intersections of isolines indicate the trajectory positions of a particle.

closed domain concept

The definition of drift fields requires that any advected particles stay inside the domain. If particles leave the domain, e.g., due to outflow regions, the flow map gradient becomes undefined. In this case, we need to restrict drift fields to a subdomain $D_\circ \subseteq D$, such that no particle leaves $D_\circ$. We determine $D_\circ$ by pathline integration for points in D until the maximum time or a domain boundary is reached. Regions for which this integration stopped early at a boundary are excluded from $D_\circ$. We use the restricted domain $D_\circ$ to compare flow maps and drift fields.

5.8.1 *Rotating Flow*

description & sampling

The Rotating Flow benchmark is defined as:

$$\mathbf{v}(x, y) = \begin{pmatrix} -y\,(1 - x^2 - y^2) \\ x\,(1 - x^2 - y^2) \end{pmatrix}$$

in the domain $[-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}] \times [0, 12]$. It describes a steady flow with a counter-clockwise rotation. The pathlines form perfect circles. We sample the flow map on a uniform $100 \times 100 \times 60$ grid.

timings & memory

For timings, we compare different temporal domains with $t_{\max} = \{3, 6, 12\}$ but always use the same discrete sampling grid, i.e., 60 samples for the time interval $[0, t_{\max}]$. Table 5.1 summarizes timings, relative mean error, and memory consumptions. Due to the additional τ dimension of the flow map, the used memory is much higher. The Rotating Flow is an analytic flow, therefore, the column for $\mathbf{v}$ is an artificial value. It lists the memory that is needed, if we sampled the

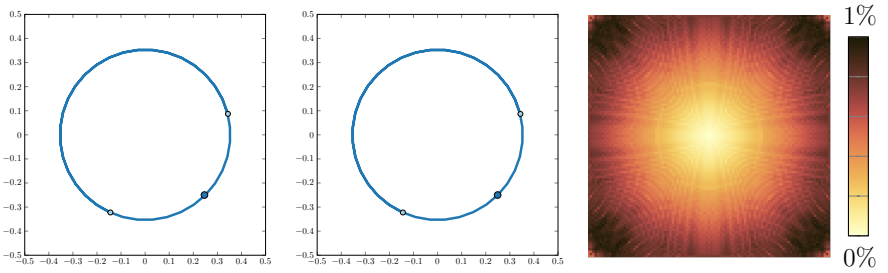


Figure 5.4: ROTATING FLOW PATHLINES AND RECONSTRUCTION ERROR. Left and Center: Comparison of a single pathline constructed from $\mathbf{d}$ (left) and ϕ (center). The circular markers indicate the seed location $\mathbf{x}_0 = (\frac{1}{4}, \frac{1}{4})$ (dark blue) for $t_s = 6$ and the final destinations (light blue) after integration for $\tau = \pm 6$. Note that pathlines cover multiple periods of the circle. Right: Mean error of the reconstruction of ϕ from $\mathbf{d}$ relative to cell size.

flow in the same spatial resolution as $\mathbf{d}$. Because it is a steady flow, only one time slice is necessary.

drift field

Figure 5.3 shows the drift field components $\mathbf{d} = (a, b)$ for selected time slices with $t_{\max} = 12$. The optimal initialization time was found at $t_0 = 6$. The bottom row of Figure 5.3 shows isolines for a and b. Intersections of isolines are the corresponding loci of a particular particle at each time.

pathlines

Figure 5.4 shows a single pathline constructed from the drift field (left) and the flow map (center). We seed a particle at $t_s = 6$. The circular markers indicate the seed position at $\mathbf{x}_0 = (\frac{1}{4}, \frac{1}{4})$ and the final destination after integration in the forward and backward direction for $\tau = \pm 6$.

mean error

Figure 5.4 right shows the average reconstruction error for $\mathbf{d} \to \phi$ relative to the spatial sampling distance. The relative mean reconstruction error is always below 1% of the cell size.

discussion

Table 5.1 shows that the computation $\mathbf{v} \to \phi$ linearly grows with τ. Contrary the computation $\mathbf{d} \to \phi$ is constant for all τ. The mean reconstruction error for $\mathbf{d} \to \phi$ slightly grows with increasing τ. Nevertheless, the reconstructed ϕ is almost identical to the original ϕ, which shows that drift fields work well for this data set. This is especially surprising when comparing the necessary memory to store $\mathbf{d}$ compared to ϕ – the memory consumption of ϕ is more than 50 times higher.

5.8.2 *Double Gyre*

sampling, timings & memory

We sampled the flow map and the drift field for the Double Gyre on a $200 \times 100 \times 100$ grid in the domain $[0, 2] \times [0, 1] \times [0, 10]$. In the selected time interval, the Double Gyre describes one full period. Table 5.1 summarizes timings, relative mean error, and the memory consumption needed for storing the flow with the listed sampling for $\mathbf{v}$, $\mathbf{d}$, and ϕ.

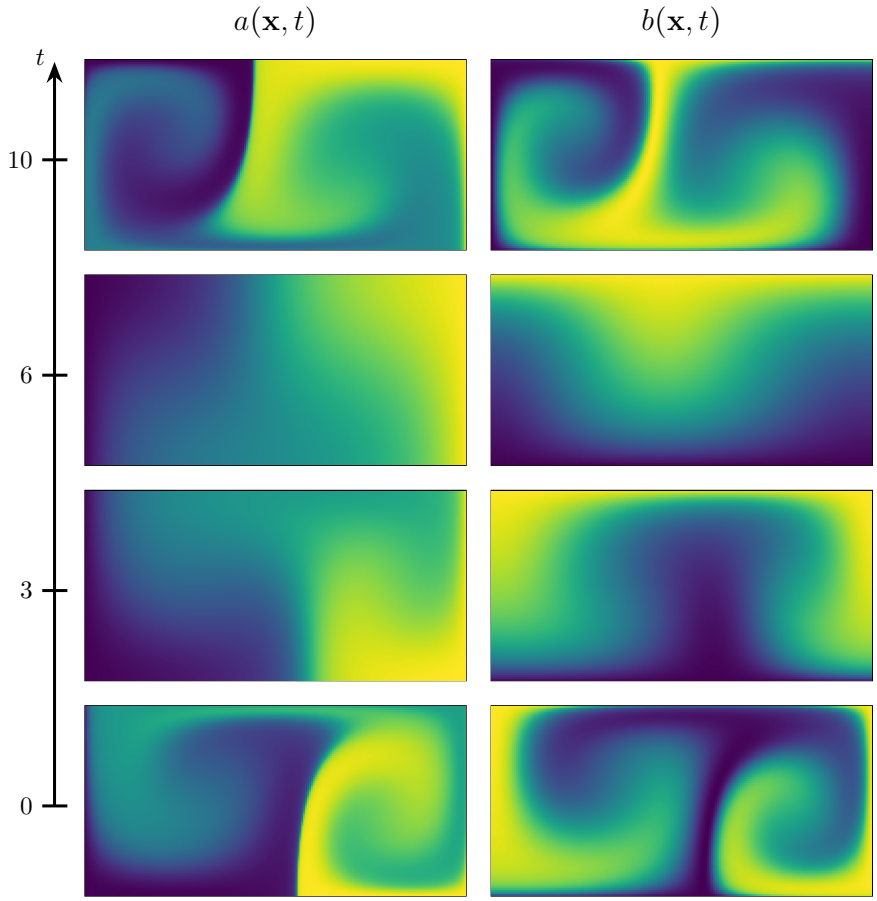


Figure 5.5: DOUBLE GYRE DRIFT FIELD. Selected time slices of the drift field $\mathbf{d} = (a, b)$. The best initialization time was determined with $t_0 = 5$.

Due to the dense sampling in the temporal domain, the number of possible (t, τ)-combinations is high. This leads to significant memory consumption and computation time for the full flow map.

drift field

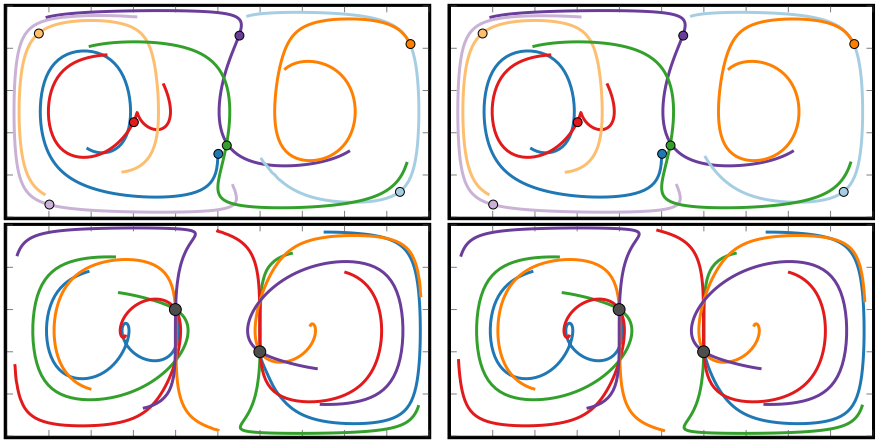
Figure 5.5 shows the drift field components $\mathbf{d} = (a, b)$ at selected times (see axis). We found the optimal initialization time at $t_0 = 5$, i.e., in the middle of the investigated range. This is not surprising because the Double Gyre is a periodic data set with a period length of 10.

pathlines

Figure 5.6 compares pathlines constructed from the drift field and the flow map. The top row shows pathlines for different seeds in space and time. The bottom row shows pathlines for a fixed seed position and varying seed times. Spatial seeds are indicated as circular markers. Pathlines from the drift field correspond well to pathlines from the flow map in all regions.

mean error

The right image of Figure 5.7 displays the mean error relative to cell size. The mean reconstruction error is below 10% in all regions, which means that the pathline accuracy is pretty good even for a medium integration time of $\tau = 10$. The error grows in areas where the flow



Pathlines from Drift Field | Pathlines from Flow Map

Figure 5.6: DOUBLE GYRE PATHLINES. Comparison of pathlines from drift field and flow map. Color markers indicate seeding positions. Top: Varying seeding positions and constant time. Bottom: Constant seeding position and varying time.

shows shearing behavior. In the other areas, the reconstruction works well. This is surprising because we expected the largest error in the center of the domain around the LCS.

modification

For the Double Gyre we applied a local modification on the drift field and reconstructed pathlines. This experiment is interesting because it lays the ground for editing flow maps. A locally modified drift field can still be interpreted as a drift field; in particular, one can reconstruct a consistent flow map. This property enables an indirect flow map modification via editing drift fields.

Figure 5.8 shows the results of this experiment. The left image illustrates the modification area; the regions in the two circles were *'smeared out'* in an up direction. The left circle modified the time range $t = [2, 4]$, the right circle modified $t = [6, 8]$. The left image shows pathlines extracted from the original drift field. The right image shows pathlines after the modification. Pathlines passing the modified area and pathlines starting in this area are deformed. Please note that the red marker's seeding time equals $t_s = 8$, i.e., the deformation affects only the *'earlier'* left part of the pathline. The modification results in smoothly changed pathlines for all affected regions. The left image of Figure 5.7 shows the deformation's effect on the drift field; the darker blue areas were smeared upwards.

discussion

Table 5.1 shows that the flow map computation time $\mathbf{v} \rightarrow \phi$ is 6.5 times higher than $\mathbf{d} \rightarrow \phi$, i.e., a conventional integration in the velocity field is more expensive than the reconstruction from the drift field. The mean reconstruction error for $\mathbf{d} \rightarrow \phi$ grows in areas with shearing behavior but is still $< 10\%$ of the sampling distance in most areas. A higher spatial sampling could increase the accuracy at the cost of more

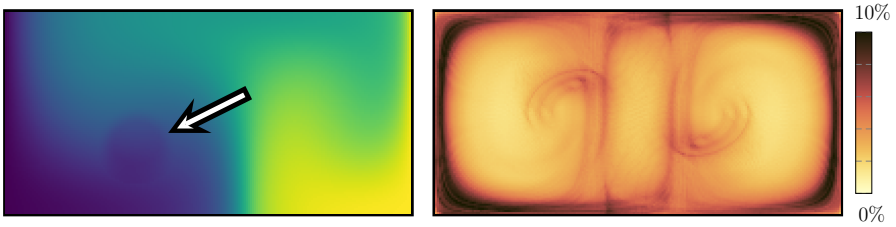


Figure 5.7: DOUBLE GYRE MODIFICATION & RECONSTRUCTION ERROR. Left: $a(\mathbf{x}, t)$ part of modified drift field $\mathbf{d}$, for $t = 3$. Right: Mean error for reconstruction of ϕ from $\mathbf{d}$ relative to cell size.

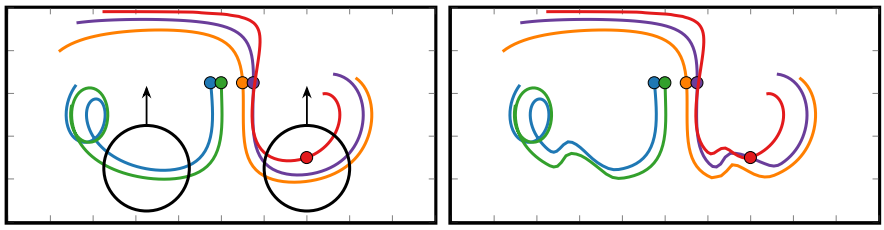


Figure 5.8: DOUBLE GYRE MODIFICATION. We show pathlines extracted from the drift field. Color markers show the seeding locations; the marker position along the pathline indicates the seeding time; $t_s = 0$ for green and blue, $t_s = 5$ for orange and purple, $t_s = 8$ for red pathline. Left: The original pathlines and the deformation areas in the spatial domain illustrated by black circles. Right: Pathlines after the deformation.

memory consumption. Due to the high temporal resolution, the memory consumption for ϕ is much higher than for $\mathbf{d}$; compare Table 5.1.

5.8.3 *Cavity Flow*

description

The Cavity Flow data set is a velocity field describing the flow over a 2D cavity. It was simulated with the compressible Navier-Stokes equations. The flow moves from left to right. Inside the cavity, the flow exhibits a non-zero divergence, while outside the cavity, a quasi-divergence-free behavior is present. Caraballo et al. [23] kindly provided this data set.

Cavity flows (i.e., laminar flows passing over an open cavity) are of interest in many applications in engineering, ranging from the small cavities due to gaps in the bodywork of vehicles, the shapes of river channel beds, via cargo bays on aircraft, to the large scale flows in urban street canyons.

sampling

The Cavity Flow is defined on the domain $[-1, 8] \times [-1, 1.5] \times [0, 5]$ and given on a $256 \times 96 \times 51$ grid with non-uniform cells. For constructing drift fields, we consider the region $[-0.45, 5.15] \times [-1, 0.5] \times [1, 4]$ and take $160 \times 60 \times 30$ samples. We had to trim a considerable amount of the domain; otherwise, too many particles would have left the do-

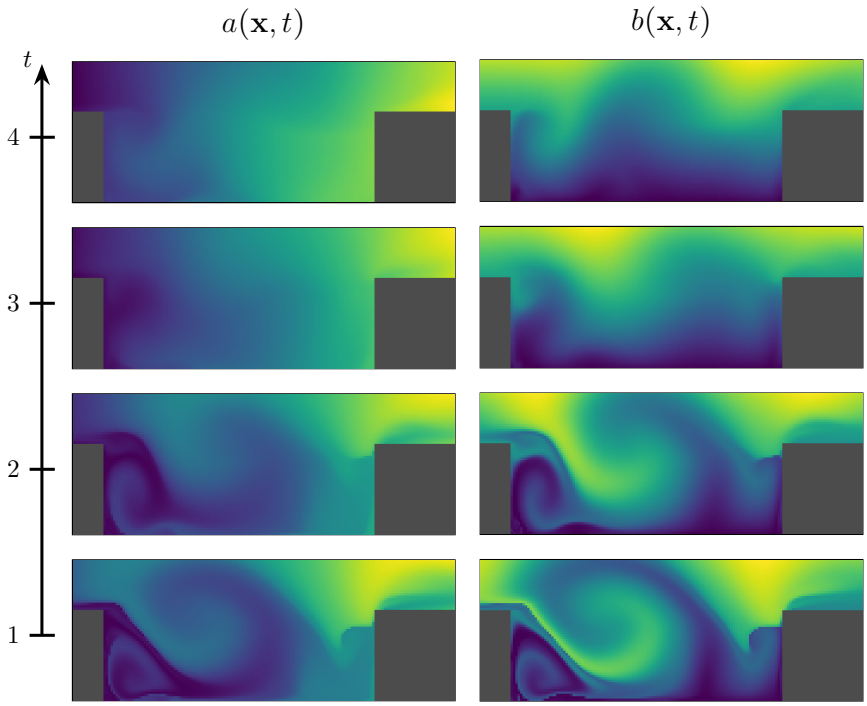


Figure 5.9: CAVITY FLOW DRIFT FIELD. Selected time slices of the drift field components $\mathbf{d} = (a, b)$. The best initialization time was determined to be $t_0 = 3.5$.

main and resulted in an undefined flow map gradient $\nabla\phi$. The optimal initialization time for the drift field was found at $t_0 = 3.5$.

timings & memory

Table 5.1 summarizes timings, relative mean error (with respect to spatial cell diagonal), and lists the memory consumption. The sampling is relatively coarse; therefore, the memory consumption and the computation time are moderate. However, the time needed to compute the flow map ϕ from the velocity field $\mathbf{v}$ is 30 times higher than the computation of the flow map from the drift field $\mathbf{d}$.

drift field

Figure 5.9 visualizes the drift field $\mathbf{d} = (a, b)$ at selected times. The initialization time $t_0 = 3.5$ is close to the upper bound of the selected time range. Hence, the advection time to the lower bound is longer. This results in a smooth drift field for higher time steps and a turbulent drift field for lower time steps.

pathlines & mean error

Figure 5.10 compares pathlines reconstructed from the drift field and the flow map. Circular markers indicate the seeding positions of the particles. The pathline reconstruction from drift fields works well in most areas but becomes inaccurate in highly turbulent areas and some areas next to the border. This becomes visible in Figure 5.11, which displays the relative mean error. We use a non-uniform spacing between samples in x- and y-direction, which resembles the sampling of the simulated velocity field. Therefore, we show the mean error for the flow

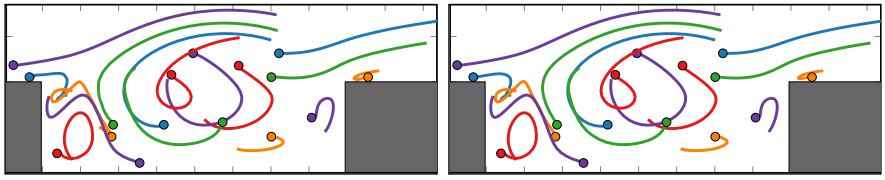


Figure 5.10: CAVITY FLOW PATHLINES. Left: Pathlines reconstructed from drift field. Right: Pathlines reconstructed from flow map.

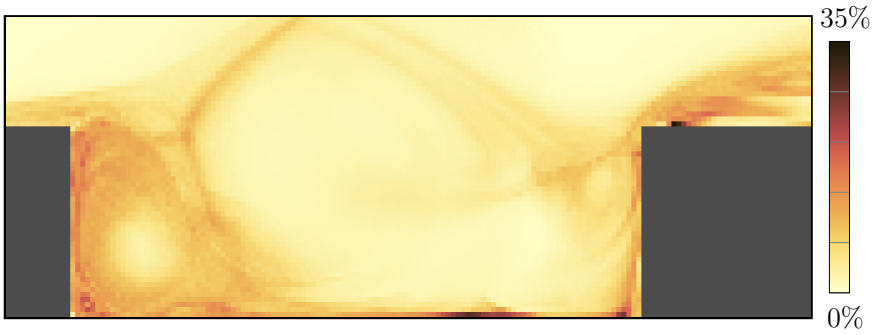


Figure 5.11: CAVITY FLOW RECONSTRUCTION ERROR. Mean error for the reconstruction of ϕ from $\mathbf{d}$ relative to the diagonal of sampling cells. The error grows in turbulent areas and next to the walls.

map reconstruction from the drift field relative to the sampling cells' diagonal.

discussion

The spatial and temporal resolution of the sampled flow is relatively low. This impacts the quality of the reconstruction $\mathbf{d} \to \phi$, which is flawed in turbulent areas and next to the walls. On the other hand, due to the low resolution, the reconstruction $\mathbf{d} \to \phi$ takes only a couple of seconds and results in low memory usage for $\mathbf{d}$; compare Table 5.1. Nevertheless, the mean error falls under sampling cells' size, and the reconstructed pathlines represent the flow very well.

5.8.4 *Piped Cylinders Flow*

description & sampling

The Piped Cylinders Flow is the result of a simulation of a step-shaped pipe and two cylinders inside. The simulation was done with *Gerris Flow Solver* [122] and is provided by Baeza Rojo and Günther [129]. The flow enters the pipe at the left border and leaves at the right border. In the back of the cylinders and the corners, highly turbulent areas appear. The spatio-temporal simulation domain is $[-0.5, 5.5] \times [-0.5, 1.5] \times [0, 15]$, and the data is given on a $450 \times 150 \times 1501$ grid. For our experiments, we consider the subdomain $[1.0, 4.5] \times [-0.5, 1.25] \times [2, 4]$ sampled on a $175 \times 88 \times 100$ grid. Due to the high velocity, particles leave the domain relatively fast, resulting in an undefined flow map gradient. Therefore, we had to trim the domain, and the left cylinder is not contained in the drift field.

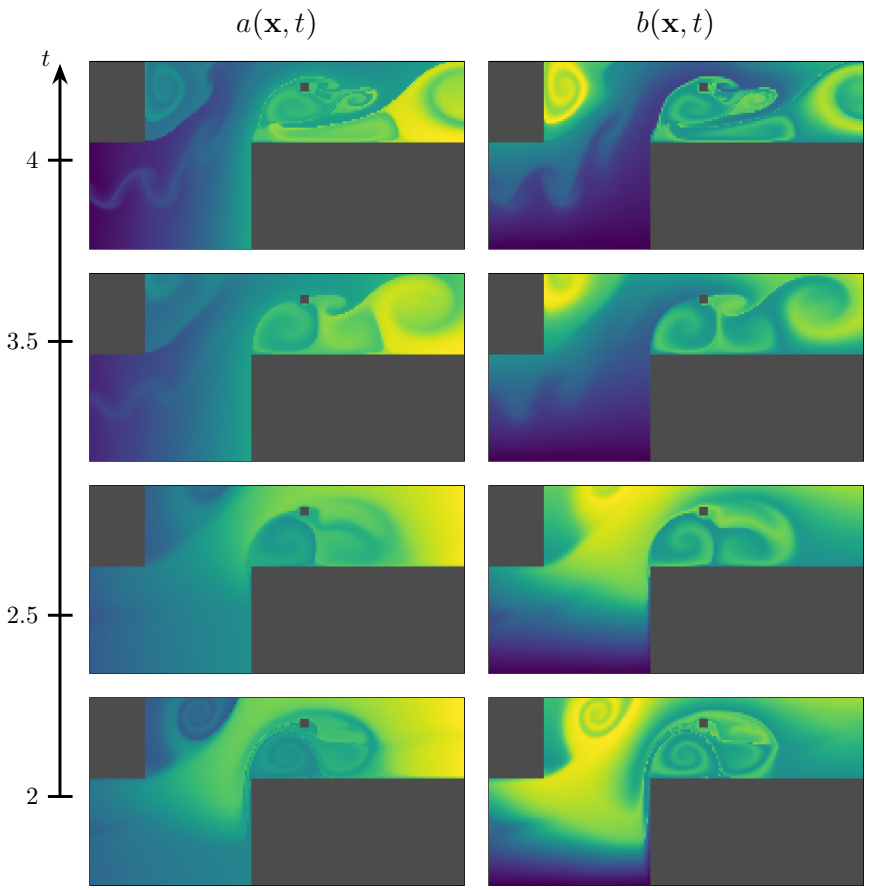


Figure 5.12: PIPED CYLINDERS FLOW DRIFT FIELDS. Selected time slices of $\mathbf{d}$ with $t_0 = 2.8$. The field clearly shows turbulent areas.

timings & memory

Table 5.1 summarizes timings, relative mean error for the flow map reconstruction, and memory consumption. The computation of the flow map from the drift field is only twice as fast when compared to a traditional integration in the velocity field. We see the reason for this in the relatively short integration time $\tau = 2$. The temporal resolution must be high to cover the high velocity of particles. This results in high memory consumption for the flow map. Many time samples lead to more (t, τ)-combinations, which have to be checked to compute the best initialization time.

drift field

Figure 5.12 visualizes the drift field for selected times. The optimal initialization time was found at $t_0 = 2.8$, i.e., it is shifted to the lower bound of the temporal domain. The drift field becomes quickly turbulent in the back of the corners and below the second cylinder.

pathlines, mean error & discussion

Figure 5.13 shows reconstructions of selected pathlines from the drift field compared to the flow map. Figure 5.14 shows the relative mean reconstruction error compared to the flow map. The circular markers denote the particle seeds. Pathlines in laminar regions can be constructed from the drift field well. In turbulent areas, the sampling is not dense enough to reconstruct adequate pathlines. While the reconstructed

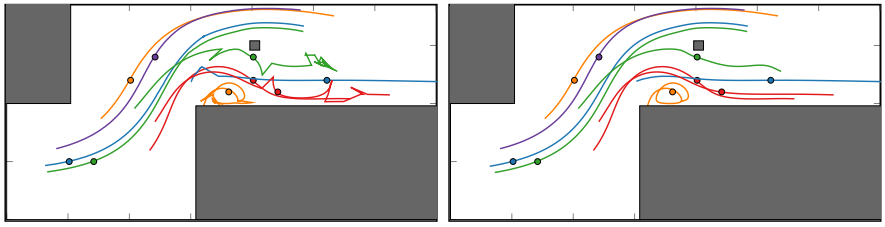


Figure 5.13: PIPED CYLINDERS FLOW PATHLINES. Pathline reconstruction from the drift field **d** is accurate in front of the cylinder; it fails in turbulent regions. Left: Pathlines from drift field. Right: Pathlines from flow map.

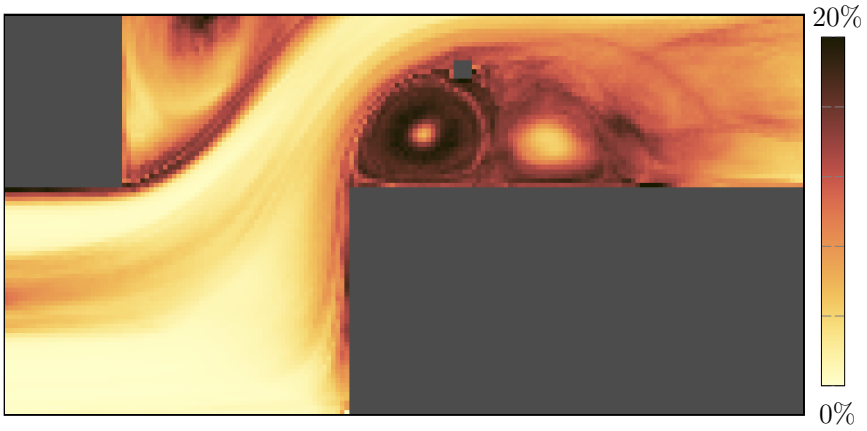


Figure 5.14: PIPED CYLINDERS FLOW RECONSTRUCTION ERROR. Relative mean error for the reconstruction of the flow map from the drift field. The error grows in turbulent regions underneath the cylinder.

pathlines are reasonably accurate in front of the cylinder, it fails behind and underneath it. This is also indicated globally by the visualization of the relative mean error, which becomes unacceptable in turbulent regions. It is still remarkable that pathlines become accurate again when entering less turbulent areas behind the cylinder.

5.9 DISCUSSION & LIMITATIONS

This section discusses the pros and cons of drift fields, particularly the relation to the established representations of flows: velocity fields and flow maps.

discussion

Drift fields do not attempt to replace velocity fields and flow maps as flow representations. Instead, they are an alternative representation complementing the suite of available flow representations. We see several scenarios where drift fields are the representation of choice. This includes:

- Cases where fast access to the flow map is required, but the flow map cannot be precomputed entirely and stored due to limited resources. In this case, a local search in the (precomputed) drift

field can be more efficient and less error-prone than a numerical integration in the velocity field, particularly for longer integration times.

- Cases where local modifications of the flow map are required. For instance, pathlines' behavior is locally adapted to a designer's needs, while the global behavior of the pathline is not affected. Such modifications cannot be modeled by directly modifying the velocity field: a local change in the velocity field results in a global change of all pathlines passing through the region of interest. Further, in a flow map representation, a local modification usually requires a global update to stay within the space of flow maps. In a drift field, such local modifications are possible in a straightforward way.

limitations

Drift fields are relatively expensive to compute. In fact, we need both the velocity field and the flow map to precompute the drift field (even though the flow map does not need to be stored to compute drift fields).

For highly turbulent flows with a strong mixing component, pathline reconstruction in drift fields becomes unstable. This is due to the fact that the gradient of the drift field is directly correlated to the gradient of the flow map. However, in this case, also a flow map representation becomes unstable because an extremely high resolution is necessary to reconstruct trajectories reliably.

By now, we do not see an approach to compute drift fields in an in-situ environment. This is due to the fact that our algorithm needs access to the complete velocity field multiple times.

5.10 OPEN PROBLEMS & FUTURE RESEARCH

future research

For future research, we see the following tasks and problems that could be addressed:

- 3D drift fields: While we restrict ourselves to drift fields for 2D time-dependent flows in this chapter, there is no fundamental reason that prevents extension to 3D. In this case, the drift field consists of 3 scalar fields.

- Further design approaches: While in this chapter, we have considered a relatively simple approach of modifying flows as a proof-of-concept, more complex operations for local flow modifications are possible. Going this way, a fully-fledged flow modeler is ultimately envisioned.

- Further flow map processing approaches: Other approaches based on a repeated local modification of a field can be transferred to flow maps, such as flow map smoothing or flow map simplifications. For this, a drift field representation can be superior because such

local modifications do not leave the space of drift fields, contrary to flow maps.

- Find better drift fields: Our approach so far considers only a subset of all existing drift fields for searching the optimal one. Currently we initialize one time t_0 with linearly increasing scalar values in x- and y-direction. This is a good solution, but we are convinced there must be more appropriate initial fields. Considering more potential drift fields may find better solutions for globally optimal drift fields.

Part III

FLOW FEATURES FROM FLOW MAPS

6

FTLE RIDGE LINES FOR LONG INTEGRATION TIMES

This chapter is based on the publication:

T. Wilde, C. Rössl, H. Theisel
FTLE Ridge Lines for Long Integration Times
IEEE Scientific Visualization Conference, 2018, Short Paper

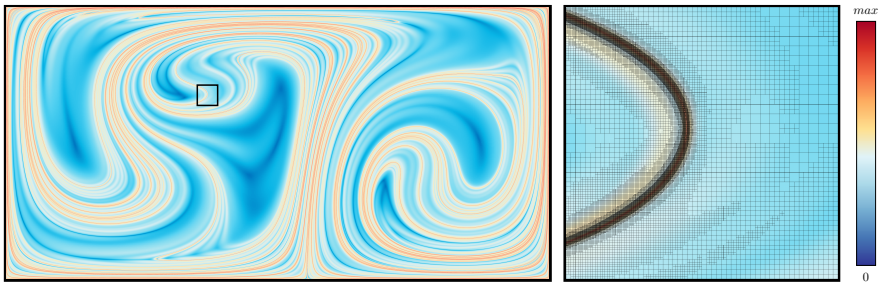


Figure 6.1: FTLE Field for the Double Gyre. Left: The long integration time $\tau = 30$, leads to a high number of clearly distinct ridges. Right: The adaptively refined grid shows a high resolution in areas where ridges are located.

Part II of this thesis was dedicated to flow map processing, i.e., working directly on the flow map data. In the following Part III of the thesis, we deal with features that we extract from the flow map and utilize for visualization.

This chapter presents an approach to extracting FTLE ridges for 2D unsteady vector fields under long integration times. Ridge extraction for long integration times is challenging because such FTLE ridges tend to be sharp and close to each other. Convenient FTLE computation sets seeding time t and integration time τ to fixed values. Our approach's main feature is that it not only uses an FTLE sampling at the desired final integration time τ_e but incorporates samples from prior integration times τ as well. Therefore, we set only one temporal dimension to a fixed value and incorporate a more significant amount of the flow map. With this additional information, the new method produces more and finer ridge lines than existing approaches. Based on this output, we can consider FTLE ridge statistics. We test the approach on synthetic benchmarks and simulated data sets.

chapter overview

This chapter is structured as follows. Section 6.1 gives a short introduction to the overall problem and motivates our solution's basic idea. Section 6.2 presents related work with a focus on FTLE and ridge concepts. Then we recapitulate the definition of FTLE and introduce the sampling problem in Section 6.3. In Section 6.4, we give a detailed discussion of the main problem, followed by a description of our solution in Section 6.5. We explain our method to extract ridges from FTLE fields in Section 6.6 and give implementation details in Section 6.7. Section 6.8 contains our results, followed by a discussion of the results in Section 6.9. Finally, Section 6.10 concludes this chapter with the limitations of our approach and possible future research.

6.1 INTRODUCTION

In Section 2.5, we gave an introduction to LCS and FTLE. LCS are a prominent and promising approach for extracting and visualizing the global behavior of time-dependent velocity fields. We consider LCS commonly as extremal structures, i.e., ridges of flow map derivatives. Among the various alternatives proposed in recent years, ridges in FTLE fields are one of the most typical LCS representatives. Such ridges in FTLE fields are structures that separate regions with different flow behavior. This chapter presents an approach to extracting ridge lines of 2D FTLE fields focusing on long integration times.

challenges

At first glance, this seems to be a common problem. The FTLE field for the desired integration time is a scalar field for which a suitable adaptive sampling has to be computed, and then standard numerical ridge extraction can be applied. However, it turns out that the problem is much harder for the following reasons:

1. FTLE computation is expensive since every sample of the FTLE field requires numerical integration.
2. FTLE ridges tend to be thin and sharp for long integration times.
3. Adjacent ridges tend to be close to each other for long integration times.

We argue that a sufficient adaptive sampling of the FTLE field for a fixed, long integration time cannot be computed with reasonable effort. This is the reason why existing approaches either restrict themselves to considering only the FTLE field – without any extraction of the ridge geometry – or to extracting only simple non-sharp ridges for short integration times.

main idea

The main idea to solve the sampling problem is based on the insight that an adaptive sampling of the FTLE field should incorporate samples at the desired integration time *and* intermediate times. This is because FTLE ridges that are sharp – and hard to extract – have been non-sharp

– and easier to extract – at shorter integration times. Based on this key insight, we construct an algorithm for adaptive FTLE sampling and FTLE ridge extraction. The algorithm is based on a quadtree subdivision of the domain where the subdivision criterion evaluates the FTLE in the whole range of integration times from zero to the desired time.

6.2 RELATED WORK

In Sections 2.1.2 and 2.5, we give a brief overview of ridge lines, LCS, and FTLE. This section gives a more detailed summary to work on the definition and use of ridges in general and in Flow Visualization, followed by work on FTLE and ridges in FTLE fields.

6.2.1 *Ridge Concepts*

ridge concepts

There are various ridge definitions along with numerical extraction techniques in image and shape analysis and Scientific Visualization. In particular, ridges serve as a standard extractor for edges in images. We refer to the discussion presented by Eberly et al. [34] for an overview of the most common ridge definitions, such as height ridges [72], curvature ridges [108], watershed ridges [54], or medial axes [119]. A discussion about the quality of different ridge extractors is given in [9] and [85]. Algorithms for the extraction of ridge surfaces in medical images are given in [48, 49], while [95] describes a polygonal approximation of extremal surfaces. Since ridge structures are sensitive to noise, scale space approaches are standard to deal with this problem [30, 96].

ridge applications

In Scientific Visualization, ridges are known to show relevant structures in many applications, and a number of ridge extraction algorithms have been proposed. In [107, 134], ridge lines are extracted for detecting vortex core lines in flow fields. Most of the different concepts of vortex core lines can be formulated by the parallel vectors operator proposed by Peikert and Roth [115]. Recently Gerrits et al. [58] presented a solution to find approximately parallel vector lines in 3D vector field ensembles. Characteristic lines in symmetric second-order tensor fields are treated by Tricoche et al. [160]. Kindlmann et al. [82] extract ridge surfaces from diffusion tensor magnetic resonance imaging data, whereas Sadlo and Peikert [131] describe a modified Marching Cubes algorithm with eigenvector orientation for ridge extraction. Sahner et al. [134] describe an approach for watershed ridges.

6.2.2 *FTLE & FTLE Ridges*

FTLE ridges

One of the most prominent approaches to extract LCS is the computation of ridges in FTLE fields, as Haller [65–67] presented. FTLE ridges have been used for a variety of applications [67, 93, 146, 176]. Shadden et al.

[145] have shown that FTLE ridges are approximate material structures, i.e., they converge to material structures for increasing integration times. This fact was used by Sadlo and Peikert [132] to extract topology-like structures. Schindler et al. [141], and Lipinski and Mohseni [97] introduce methods for tracking FTLE ridges by locally sampling the FTLE field and estimating the ridge direction and location. Due to the discrete sampling used, the accuracy is limited, especially for very sharp ridges. Farazmand and Haller [39] integrate the minor eigenvector of the Cauchy-Green tensor to track ridge structures. This approach is, however, prone to accumulating integration errors. Also, in the visualization community, different approaches have been proposed to increase the performance, accuracy, and usefulness of FTLE as a visualization tool [50, 51, 56, 121, 131, 133]. Haller and Sapsis [64] additionally explore the smallest FTLE values.

FTLE in 3D

While most approaches mentioned above restrict themselves to ridge curves in 2D flows, a few approaches extract ridge surfaces in 3D flows for reasonable integration times. Schindler et al. [141] show both standard height ridge extraction and C-ridge tracking to get 3D surfaces. Furthermore, the same group utilizes C-ridge surfaces for an analysis of revolving doors [140]. Sadlo and Peikert [131] present an adaptive grid generation for FTLE computation and extract FTLE ridge surfaces. Üffinger et al. [163] present streak surfaces as approximations to FTLE ridges. Depending on the seed structures' accuracy (obtained by extremely high sampling), streak surfaces and FTLE ridges show strong agreement. Barakat et al. [6] propose a smooth adaptive reconstruction of the flow map field from the sample points based on Sibson's interpolant on which the ridge extraction is more stable than on the original sampling. Günther et al. [61] present an approach to compute ground truths for FTLE fields based on unbiased Monte Carlo rendering. Rojo et al. [128] accelerate this rendering process significantly.

6.3 BACKGROUND

FTLE

In Section 2.5, we give a formal definition and explanation of how to compute the FTLE. Accordingly, FTLE is defined as:

$$FTLE(\mathbf{x}, t, \tau) = \frac{1}{|\tau|} \ln \sqrt{\lambda_{\max}(\Delta)}\,,$$

where $\lambda_{\max}(\Delta)$ denotes the largest eigenvalue of the (right) Cauchy-Green strain tensor.

FTLE evaluation & sampling distance

The standard approach to the numerical evaluation of $\mathrm{FTLE}(\mathbf{x}, t, \tau)$ uses a discrete approximation of the flow map gradient $\nabla\phi(\mathbf{x}, t, \tau)$ from finite differences of ϕ. When using central differences, this requires four evaluations of the flow map, i.e., four times a numerical integration of $\mathbf{v}$. Any discrete difference scheme requires size parameter h, which here denotes the spatial distance of two initial points before advection on the flow. The smaller h, the more accurate the derivative's approximation – the error is of order h^2 for central differences. However, if h is chosen too small, other sources of error, e.g., interpolation or the numerical integration, may dominate the result. For ridge extraction, in particular, smaller h – or observing a smaller region that is advected – will decrease the 'probability' that a ridge is passing this region and is detected due to a high separation and a local maximum of FTLE. There is, however, no information about how many, possibly sharp ridges pass the support of finite differences, which disqualifies the choice of a larger h for our purpose.

In order to find ridges, we have to evaluate FTLE. Throughout this chapter, we use the term *sampling* for any set of FTLE samples that supports ridges' extraction. Samples are parametrized by $(\mathbf{x}, t, \tau)$ without an assumption of a particular distribution, e.g., a uniform grid.

sampling problem

Based on the above considerations, we can formulate the problem as follows: Given a fixed starting time t and a fixed, positive – long – integration time τ_e, find a sampling of a scalar field $\mathrm{FTLE}(\mathbf{x}, t, \tau_e)$ sufficient for extracting all ridges within a maximum resolution that is prescribed by numerical methods. We call this the *sufficient sampling problem*.

6.4 PROBLEM ANALYSIS

The solution of the sufficient sampling problem is the main challenge in FTLE ridge extraction. Given a desired final integration time τ_e and a fixed starting time t, a sampling of $\mathrm{FTLE}(\mathbf{x}, t, \tau_e)$ in the spatial domain is searched, which sufficiently supports all ridges' extraction. We analyze this problem in the following.

6.4.1 *Importance of FTLE Ridges*

ridge completeness

Explicit ridge geometry gives more information than the scalar FTLE field alone. An FTLE field and its ridges have a similar relationship as a scalar field and, e.g., its isosurface geometry – both representations have been established in coexistence. For volume data, both direct volume rendering and isosurface extraction are well-accepted techniques. In particular, the extraction of FTLE ridge geometry allows considering ridge statistics.

6.4.2 *Importance of Ridge Statistics*

ridge statistics

FTLE is usually applied to non-turbulent flows. The transition to turbulent flows makes ridges behave 'wild', i.e., they become sharp and dense. In this case, the behavior of one particular ridge is of less interest. Instead, insights can be gained from statements on the set of all ridges. Ridge statistics can help manifest such statements and enable the use of FTLE to justify the transition towards turbulent flows.

6.4.3 *Importance of Ridge Separation*

ridge distinction

Adjacent FTLE ridges generally correspond to separating events that occur at different integration times. Imagine a group of spatially close particles traveling with the flow. If, after a particular time, a separation takes place, the particles divide into two subgroups, each showing different further flow behavior. After an even longer integration time, each subgroup could be separated again into new subgroups. This results in spatially close FTLE ridges around the initial locations of the particles.

6.4.4 *Analysis of FTLE in 1D*

ridge sharpness & distance

We simplify the analysis by restricting the domain of FTLE to a straight line in the flow domain, where FTLE ridges are the local maxima of a 1D FTLE function. Figure 6.2 (left) illustrates a typical behavior of such a function; it is relatively small and almost zero outside a narrow band. Within this band, there are local maxima, each representing a ridge. We consider the distance d between two ridges and the sharpness s of a single ridge. We measure d as the distance between two consecutive maxima. The function decreases to the left and the right of a maximum. If the function reached half of its maximum, we measure the span at that location. We denote this span as the sharpness s of a ridge – the smaller s, the sharper the ridge.

properties P1 & P2

We observe how d and s change with increasing integration time τ and formulate

- *Property 1 (P1):* The sharpness s of an FTLE ridge tends to decrease exponentially for linearly increasing integration time τ.
- *Property 2 (P2):* The distance d to the next ridge tends to decrease exponentially for linearly increasing integration time τ,

Here, *'tend to'* means that there exist realistic flows that show this behavior. Both properties were observed and stated similarly by Kuhn et al. [87].

1D example

For demonstration, we consider the Double Gyre data set. We study a horizontal 1D cut through the domain and place dense samples along a

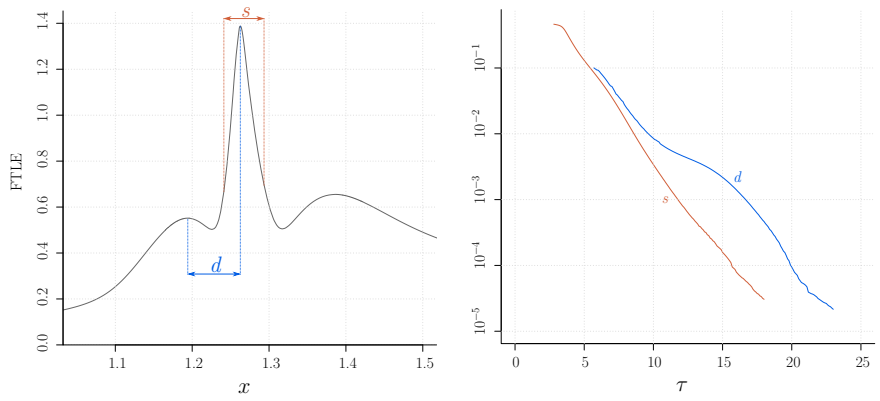


Figure 6.2: RIDGE BEHAVIOR IN 1D. Left: Illustration of the measures for sharpness s and the distance d of ridges. Right: Example for sharpness s and distance d plotted in a semi-log graph. Sharpness and distance decrease exponentially, i.e., ridges get sharper and come closer together.

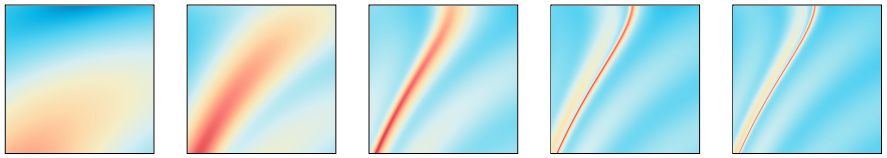


Figure 6.3: EMERGING FTLE RIDGE. FTLE time series of the Double Gyre's center region, $[1.2, 1.4] \times [0.65, 0.85]$. The ridge emerges over time and becomes sharp.

line at $y = 0.75$. We compute the FTLE values for these samples with a fixed $t = 0$ and increasing τ, with $\tau = {}^{k}/_{10}$, $k \in \{1, 2, \ldots, 250\}$.

The plot in Figure 6.2 (left) shows a part of the resulting FTLE function for $\tau = 6.5$. The strong center ridge appears 'early' after a few discrete time steps. Shortly after, the two smaller ridges to the left and right emerge. Figure 6.2 (right) shows the development of the ridge sharpness and distance with increasing τ. We plot the sharpness s for the center ridge and the distance d between the left and center ridges on a semi-log scale.

Figure 6.3 illustrates this ridge in a 2D domain. The ridge slowly emerges and gets quickly strong and sharp with ongoing integration time. The sampling density is almost too low to resolve the ridge at $\tau = 10$. Further ridges located to the left and the right of the center ridge become visible. Graph and images confirm properties *P1* and *P2*.

6.4.5 *Sampling Density*

sampling criteria

In order to capture a ridge in the FTLE field, the sampling density must not fall below the ridge sharpness s. Otherwise, the ridge may be missed entirely. This and *P1* mean that for a large τ_e, we would need an

extremely dense regular sampling to capture the ridge. Even if we had such a sampling, there is no guarantee that an even denser sampling does not reveal more ridges. This problem cannot be solved by adaptive sampling either. Starting with a coarse initial grid, we would need two local criteria for steering the adaptive sampling:

(a) Subdivide a cell because a ridge is expected to pass through it.

(b) Stop further subdivision of a cell containing a ridge because it is sure that only one ridge is passing through. A further subdivision would increase the accuracy of the ridge location but would not reveal new ridges.

A criterion for (a) is challenging because if a ridge falls between two samples, it may be missed entirely and therefore does not give information that a further subsampling is necessary. This problem is central to any sampling. One common way to circumvent it is to increase the size parameter h when estimating the flow map gradient $\nabla\phi$. Coupling h and the sampling density may help answer whether there are any ridges present.

However, it will generally not help to separate multiple close ridges or even distinguish between detecting a single or multiple ridges. We are not aware of any existing criteria for (b). We believe that such a criterion does not exist because of *P2*. This is why we believe that it is *impossible* to get sufficient sampling by considering FTLE values for the final integration time only.

6.4.6 *Main Idea*

observation & main idea

As a solution to this problem, we propose to steer an adaptive subdivision by incorporating FTLE sampling for shorter integration times. The following observations justify this. Ridges emerge (slowly) over time; a very sharp ridge that is close to its neighbors at the final integration time τ_e was 'better behaved' at a shorter integration time $\tau < \tau_e$. Stable ridge extraction is possible for shorter integration times, as *P1* and *P2* indicate. Increasing the integration time from τ to τ_e makes a further subdivision necessary due to the ridge's increased sharpness and spatial motion. However, ridge motion decreases with ongoing integration time, which corresponds to the fact that FTLE ridges converge to material separation structures for increasing integration time (see [145]). This way, we can make sure that *all* ridges of $\mathrm{FTLE}(\mathbf{x}, t, \tau_e)$ are resolved.

6.5 FINDING A SUFFICIENT SAMPLING

We first construct an algorithm for finding a sufficient sampling that supports the extraction of FTLE ridges.

6.5.1 *Algorithm Specification*

algorithm input

The *input* is a flow field $\mathbf{v}(\mathbf{x}, t)$ in a domain D and a final integration time τ_e. The period τ_e is high such that we expect sharp ridges, which are close to each other. Two ridges cannot get arbitrarily close to each other and still be robustly separated. At some point, numerical errors, e.g., from interpolation and integration or floating-point precision, will put a practical limit on computing ridge locations precisely enough to distinguish the two different ridges. For this reason, the algorithm takes a final input parameter, the maximum resolution ϵ, which limits the minimal spatial distance between two samples. We assume that there will be no gain of information if samples get closer than ϵ. Even worse, a denser sampling may lead to inconsistent information due to numerical noise and is therefore not reliable.

algorithm output

The algorithm's *output* is a set of samples $(\mathbf{x}, t, \tau)$ that supports ridges' extraction within the maximum resolution ϵ. For each sample, $\tau \leq \tau_e$ denotes the maximum integration time until either the desired τ_e was reached, i.e., $\tau = \tau_e$ or the maximum spatial resolution ϵ was reached and would have been exceeded at τ_e, i.e., $\tau < \tau_e$. In the first case, the sampling locally supports the extraction of ridge geometry. In the second case, a reliable ridge extraction is possible only for the returned $\tau < \tau_e$.

6.5.2 *Domain Discretization & Initialization*

We discretize the domain by a coarse grid of square cells, where a quadtree subdivision refines each grid cell. The algorithm then finds the minimal amount of subdivisions for all quadtrees and yields samples defined by quadtree leaf cells.

discretization

We use a coarse grid to get a certain amount of refinement that is needed for the initialization. Otherwise, we would lack information about the interior of the domain. Alternatively, we could prescribe a minimum quadtree depth but decided not to 'waste' refinement. Also, the grid facilitates domain decomposition for parallel processing independent of quadtree hierarchy. We use square cells to get the same resolution for the dimensions. This is also convenient and not a limitation; the algorithm would work similarly for rectangular cells.

quadtree

We decided to use a quadtree for adaptive refinement of the domain because it is a relatively simple data structure with mature and efficient operations, e.g., traversal, finding neighbors, and inserting elements (compare, e.g., [40, 42]). While there are established algorithms for the computation of FTLE on unstructured meshes (compare, e.g., [94]), their refinement is more complicated. The regular structure of the quadtree fits our needs best. Note that the choice of the quadtree limits the approach to isotropic refinement. Refinement and alignment of an

anisotropic, possibly unstructured grid over time τ would lead to a significantly more complex data structure and algorithm.

quadtree initialization

The domain D gets covered with square cells. Each cell represents a quadtree of depth 0; the cell itself is the tree's root and the only leaf node. We found that the algorithm is not sensitive to the *size* of the initial square cells. For this reason, we do not list it with the input parameters but suggest squares with a size 1% of the total area of D. Usually, D is rectangular. For instance, if D has the dimensions 200×600, the initial grid will have 20×60 cells.

6.5.3 *Refinement of the Sampling*

FTLE at cell center

We associate a sample with each quadtree cell. The spatial position $\mathbf{x}$ is the cell's center. Assuming a fixed start time t, we store with each cell the current integration time τ and $\mathrm{FTLE}(\mathbf{x}, t, \tau)$, which we compute from finite differences of $\phi_t^\tau(\mathbf{x})$ evaluated at the four cell corners.

increase τ

The refinement iteratively increases the integration time τ by a step $\Delta\tau$. We sample the flow map at each time step for computing finite differences. This sampling is decoupled from the integration (compare Section 6.7), and the choice of $\Delta\tau$ does not interfere with the adaptive step-size chosen for numerical integration. We enumerate iterations $k = \{1, 2, \dots\}$ and set $\tau = \min\{k \cdot \Delta\tau, \tau_e\}$ until $\tau = \tau_e$. Each iteration proceeds in two steps:

1. Update FTLE_τ and τ for all *leaf cells*.
2. Mark all leaf cells that must be split, then apply the splits.

Here, the term 'leaf cells' refers to all leaf nodes of *all* quadtrees.

update FTLE

The first step is straightforward. With a simple bookkeeping of flow map values, the FTLE update requires only continuation of integration from $(k-1) \cdot \Delta\tau$ to $k \cdot \Delta\tau$ if an integration had been started from the same domain point before. This requires few(er) integration steps, at the cost of additionally storing results of flow map evaluations.

split leaf cell

The second step eventually splits quadtree cells at the cell center and refines the sampling locally. A quadtree cell that is split becomes an internal node with four new leaves as children. If a cell's size is within the maximum resolution ϵ, it can be split. We mark a cell for splitting if one of the following conditions is violated for this cell and any of its quadtree neighbors:

(a) The absolute difference of their depth in the quadtree must not exceed 1.

(b) The absolute difference of their FTLE_τ values must not exceed a threshold θ.

Here, two cells are *neighbors*, if they share a corner, an edge, or part of an edge.

quadtree split condition

Condition (a) accounts for a regular refinement of the quadtree. This means that neighboring cells are either on the same level or have at most one level difference. This way, information from the comparison of FTLE_τ values is reliable for neighboring cells.

FTLE split condition

Condition (b) is the essential one, and it is motivated as follows. Ridges gain sharpness over the integration time. They are 'thicker' and 'lower' for shorter τ and become 'thinner' and 'higher' with increasing τ. Whenever we observe an FTLE_τ value that gets locally maximal 'higher' for some τ, the cell is a candidate for supporting a ridge. Also, its neighbors are of interest as we do not know the ridge's exact position, and it may move between cells in the future. Therefore, the condition is symmetric and tests absolute difference, regardless of the observed cell. While this reads trivial, the only reason why this refinement rule can work reliably is that the algorithm tracks ridges and their development over time. At observation time, the ridge is not yet too sharp, and it may become sharper in the future. The refinement takes care of this and adapts the resolution locally – and exponentially – such that the ridge can be tracked further and such that the event that new, close ridges develop can be detected.

The maximum resolution ϵ may stop further refinement locally, and it restricts the maximum depths of the quadtrees. If a cell cannot be split anymore, the algorithm still updates FTLE_τ for the current integration time, but the τ value stored with the cell is not updated anymore. This identifies regions of missing resolution and provides the maximum integration time until ϵ is reached as output.

6.5.4 *Additional Parameters $\Delta\tau$ & θ*

We introduced two additional parameters in the previous section, $\Delta\tau$ and θ.

step size $\Delta\tau$

The *step size* $\Delta\tau$ determines the sampling density of the integration time τ up to the target time τ_e. Smaller values will lead to more iterations of the main loop. However, the overall cost does not increase drastically. This is because splitting events occur at specific times and independently of a higher sampling density. The total number of splits remains the same. A smaller value of $\Delta\tau$ brings additional cost for the extra tests in each loop. It may slow down progress if it limits an adaptive step size in flow map integration. If $\Delta\tau$ is chosen too large, split events may be missed. The extreme case is a standard adaptation that considers FTLE only for the final τ_e. We recommend a choice in the order $\Delta\tau = 1\%\tau_e$. We use $\Delta\tau = 1/10$ in all examples. The algorithm's output is not sensitive to this choice unless a much larger time step is used.

FTLE threshold θ

In contrast, the algorithm *is* sensitive to the choice of the *threshold* θ, which steers the local refinement. If this threshold is too large, there is not enough adaptation, and the generated sampling is not sufficient as ridges may be missed. If it is too low, the refinement will be too aggressive. The extreme is an exhaustive subdivision for $\theta = 0$, representing a uniform grid at maximum resolution, i.e., with cell size ϵ. We recommend a value of 1% of the maximum FTLE value at τ_e that is estimated from a coarse sampling. This worked fine for our benchmarks.

6.6 RIDGE EXTRACTION

The adaptive quadtree refinement provides a sufficient sampling for the extraction of ridges that can be reliably observed within the maximal resolution. A three-step process of filtering, clustering, and post-processing is used for extracting ridges from quadtree cells.

6.6.1 *Cell Filtering*

determine ridge candidates

In the first step, we find or filter out cells, i.e., leaf nodes of the quadtree, that are candidates for supporting a ridge. We reject all other cells. We propose doing this, similar to edge extraction in image processing. Therefore, we utilize the spatial gradient of the cells' FTLE values, estimated from a neighborhood.

In the first filtering stage, we consider a cell as a candidate only if its FTLE value is a local maximum in the co-gradient direction, i.e., it may support a ridge in the orthogonal gradient direction. The second stage filters these preliminary candidates and rejects those not supported by at least one other candidate. Candidates support each other if the two cells are *neighbors* in one or the other gradient direction. This rule rejects isolated candidates and filters out those which support a ridge.

6.6.2 *Ridge Clustering*

connect ridge candidates

The second step partitions the filtered candidates into groups of connected 'chains' of neighboring cells such that each group supports a ridge. This step is essentially a simple bottom-up clustering of candidate cells. Due to the prior filtering, the candidates are just 'chained up' by the local neighborhood. A prerequisite is that each candidate has at most two other candidates as neighbors – one in forward and one in backward direction along the ridge. More than two neighboring candidates exist when two ridges come too close. At some point, they will undergo the maximum resolution and cannot be separated anymore. For three neighbors, we select the two with the most similar FTLE_τ values. For more than three neighbors, the candidate, which introduces a cluster boundary or 'gap', is rejected.

6.6.3 *Post-Processing*

fill gaps

In the last step, we connect clusters if a single cell separates them. Effectively, this stage closes small gaps that may arise from filtering. If a cell's choice for connecting is not unique, FTLE_τ is examined similarly as for clustering. Clusters are connected only if the result is a distinct ridge, i.e., it will not modify the special case's handling from clustering.

extract ridge lines

Finally, we extract ridge lines from cells. We do not aim to find the 'exact' ridge locations but instead output a polyline with the ridge cells as vertices. This gives lines suitable for ridge statistics and avoids pretending exactness by an interpolation or a potentially error-prone optimization near the maximum resolution.

6.7 IMPLEMENTATION

implementation details

We implemented the algorithm in C++. For flow map integration, we use a fourth-order Runge-Kutta integration with adaptive step size. At every grid point (which emerges possibly during refinement), we compute and store the flow map *once* and for the maximum time range as a discrete *function* of t. The integration up to τ_e yields a cubic $\mathcal{C}^1$-continuous spline. Storing the Hermite spline costs memory but frees us from storing or proper reinitialization of the ODE solver state. This also ensures no interference between the adaptive integration step size and the time-step $\Delta\tau$. We use the latter only to evaluate the precomputed spline. Flow map integration is the only part of the algorithm executed in parallel on multiple CPU cores using OpenMP. All floating-point arithmetic is carried out in double precision.

maximum resolution

Our experiments restrict the cell size and set the maximum resolution ϵ to values between 10^{-2} and 10^{-5}. We advise not to use smaller values as we observe that the amount of 'numerical noise' from FTLE computation may otherwise be in a similar order of magnitude and render 'high resolution' results meaningless. Experiments by Kuhn [87] also back this observation. The exact choice of ϵ within this range controls the result's accuracy and the overall computational cost (compare Section 6.8).

split computation

Despite the adaptive sampling, memory usage can get high for small ϵ depending on the data set. We leverage this by treating cells of the coarse grid – each represents a quadtree – individually. This can be done sequentially or in parallel on multiple machines. We summarize typical execution times and memory usage in the next section.

6.8 RESULTS

data sets

There are many examples of FTLE fields in the literature. In most of them, ridges are well-behaved, i.e., there are a few well-separable ridges. We are interested in challenging cases and test the algorithm on

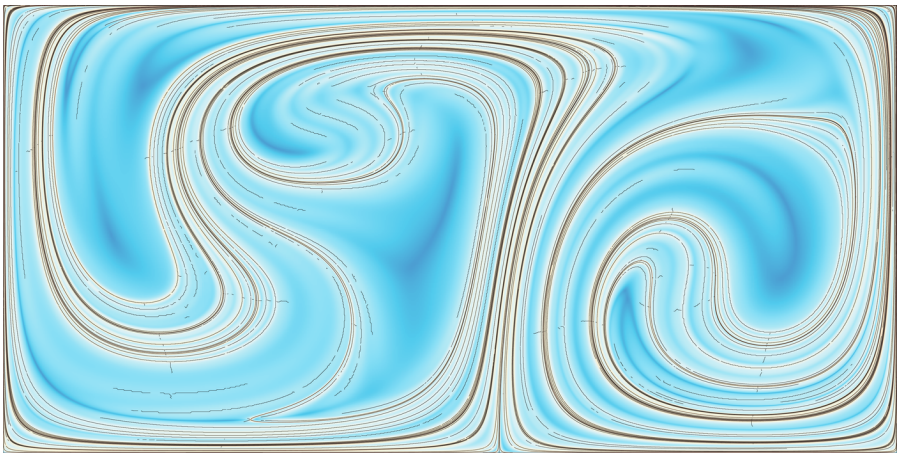


Figure 6.4: DOUBLE GYRE WITH RIDGE GEOMETRY: FTLE field for $t = 0$ and $\tau = 25$ with overlaid ridge geometry.

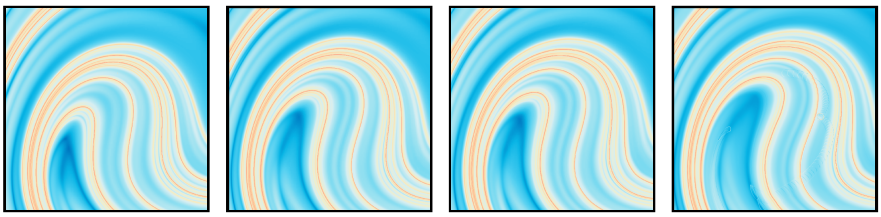


Figure 6.5: DOUBLE GYRE FTLE FOR SAMPLED VELOCITY FIELDS. Comparison between analytic benchmark and discrete sampled versions of the velocity field $\mathbf{v}(\mathbf{x}, t)$. We use $(n_x \times n_y \times n_t)$ samples for x, y, and t and a trilinear interpolation. The overall ridge structure stays the same.

four different data sets. The *Double Gyre* and the *Forced Duffing* are synthetic data sets, which can be evaluated at arbitrary resolution. The *Boussinesq* and *ECWMF* data sets stem from simulations and are given as discrete samples on uniform grids.

color map

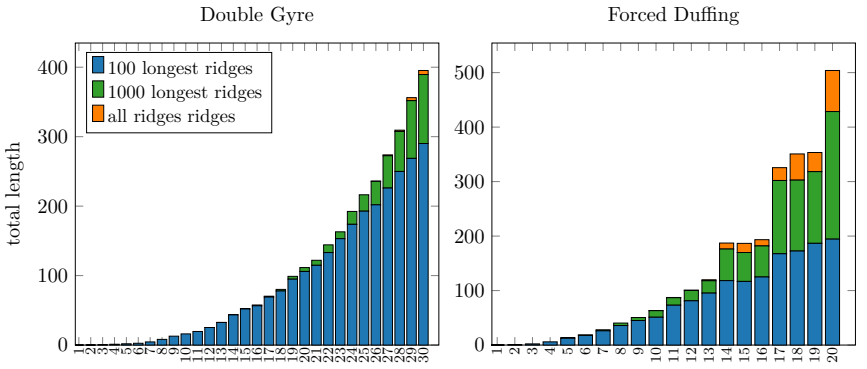
For all FTLE fields we use the *blue-yellow-red* color map. We show it in Figure 6.1 at the beginning of this chapter. Although FTLE fields contain (almost) linearly increasing values, we decided to use a diverging color map. Thus the ridges are better visible.

6.8.1 *Double Gyre*

FTLE & ridge geometry

Shadden et al. [145] introduced the Double Gyre specifically to examine LCS as ridges in FTLE fields. Figure 6.1 shows the result of our algorithm for $\tau = 30$ and a close-up with the quadtree. The quadtree shows a high subdivision in areas with ridges.

Figure 6.4 shows a combination of ridge geometry and FTLE field for $\tau = 25$. For such a long integration time, the overall ridge structure



τ	100 longest	1000 longest	all
5	1.82	1.82	1.82
10	16.06	16.06	16.06
15	51.95	52.33	52.33
20	105.97	111.56	111.56
25	192.73	216.33	216.33
30	290.14	389.46	395.29

τ	100 longest	1000 longest	all
5	12.53	13.45	13.45
10	51.07	63.11	63.38
13	95.36	118.05	119.83
14	118.24	176.21	187.21
17	167.60	302.04	325.58
20	194.38	428.54	503.92

Figure 6.6: RIDGE STATISTICS FOR DOUBLE GYRE & FORCED DUFFING. We list the accumulated ridge lengths over τ. Left: The ridge geometry of the Double Gyre shows exponential increase with a few long ridges. Right: The ridge geometry of the Forced Duffing contains sudden jumps and more medium-long ridges.

is complex and leads to a highly subdivided quadtree in most areas. Due to high memory consumption, we split the domain into eight parts. This leads to missing ridge connections between the domain parts.

The most prominent FTLE ridge for the Double Gyre emerges with increasing integration time in the domain center. The time series in Figure 6.3 shows close-ups of this ridge for different (short) τ.

sampling test

The Double Gyre flow is a synthetic benchmark, which can be evaluated *continuously* at any domain point. Real-world data sets are *discrete* data sets, typically the result of a simulation. Therefore, a flow evaluation requires the reconstruction from samples or equivalently interpolation on a finite basis. We 'simulate' this setting by sampling the Double Gyre and evaluating the flow $\mathbf{v}(\mathbf{x}, t)$ from a trilinear interpolation in space-time.

We compare the original benchmark with samplings at different resolutions in Figure 6.5. The global structure of FTLE ridges remains the same despite the loss of information from sampling and interpolation. This experiment confirms that FTLE in general and extracted ridges are stable under small perturbations, here the added 'noise' from interpolation errors. However, for the lowest resolution FTLE shows visible artifacts due to lack of information and the fact that the piecewise trilinear interpolation is not sufficiently smooth. This can be seen in Figure 6.5 (right).

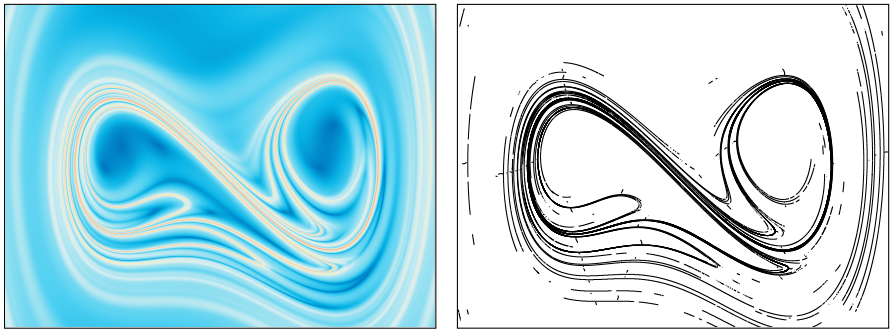


Figure 6.7: FORCED DUFFING FTLE & RIDGE GEOMETRY. Results for the domain $[-2.0, 2.0] \times [-1.5, 1.5]$ with $t = 0$ and $\tau = 15$. Left: FTLE field with long distinct ridges. Right: Extracted ridge geometry as vector graphic.

ridge statistics

Figure 6.6 provides ridge statistics for the Double Gyre. For the given integration times τ, we sum the length of the 100 longest, 1000 longest, and all ridges. The plot reveals that the total ridges lengths grow exponentially with increasing integration time. Nevertheless, only a small number of ridges dominate the domain, i.e., we have a few long ridges. Such behavior might be typical for non-turbulent flows.

6.8.2 *Forced Duffing*

description

The Forced Duffing equations are a set of differential equations commonly used to model complex, chaotic oscillatory movements [37]. There is a variety of different versions, which describe damped and undamped motions.

We use the Forced Duffing oscillator as described in [64] as:

$$\mathbf{v}(x, y, t) = \begin{pmatrix} y \\ x - x^3 + \frac{1}{10} \sin t \end{pmatrix} .$$

This velocity field describes a 2D undamped, area-preserving motion, and it leads to complex ridge structures after short integration times, which makes it a good benchmark. We computed the FTLE for $\tau = 15$ in the domain $[-2.0, 2.0] \times [-1.5, 1.5]$ with maximum resolution $\epsilon = 7.8 \cdot 10^{-5}$.

FTLE & ridge geometry

Figure 6.7 (left) shows the FTLE field and Figure 6.7 (right) the corresponding ridge geometry for $t = 0$ and $\tau = 15$. The ridges form a double-loop in the center of the domain. New ridges emerge at the border and quickly move towards the double-loop. Although we allow a high resolution, the ridges are already too close to be entirely resolved.

ridge statistics

Figure 6.6 gives the ridge statistics. We accumulate the length of the 100 longest, 1000 longest, and all ridges for the listed integration times τ. The plot reveals an exponential growth of ridge lengths with a sudden increase of medium-long ridges at $\tau = 17$ and $\tau = 20$. This might

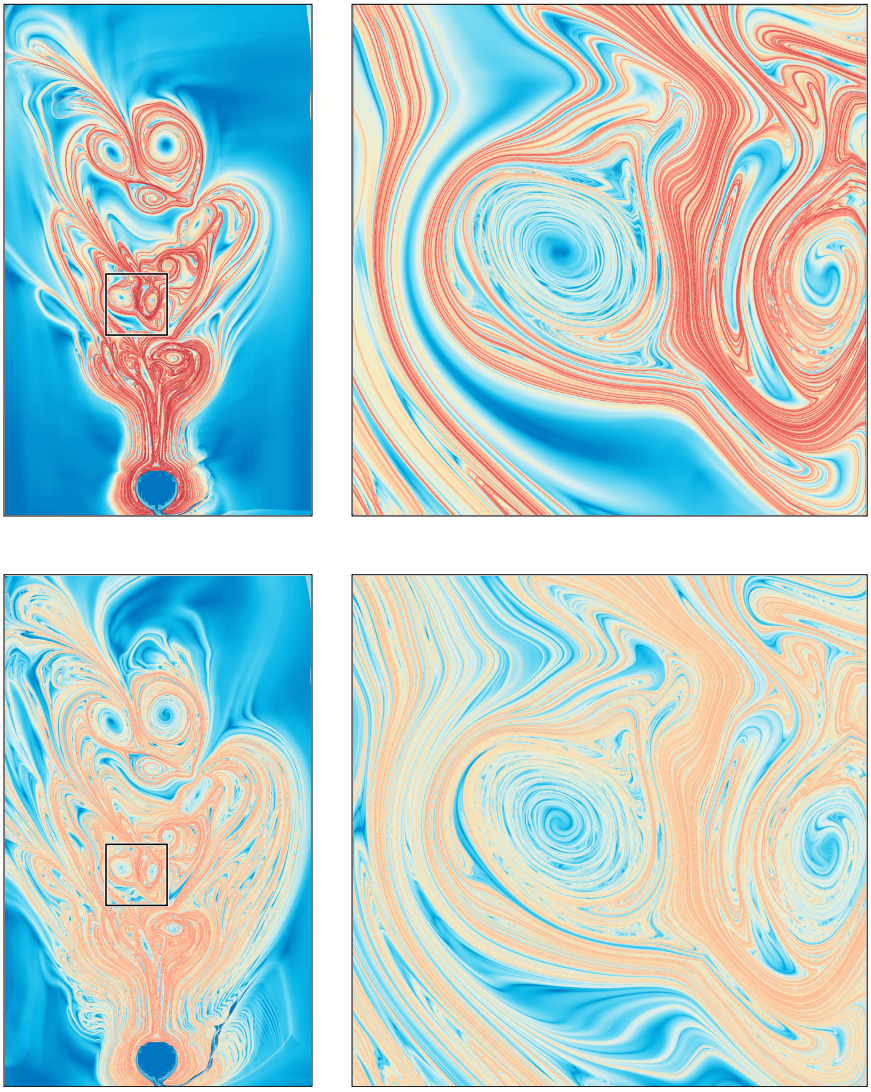


Figure 6.8: BOUSSINESQ FTLE. Top: Field with $t = 5$ and $\tau = 10$. Bottom: Field with $t = 5$ and $\tau = 15$. The right images show close-up views of the marked regions. The flow shows a huge number of complex ridges.

indicate a sudden separation of many particles at these integration times.

6.8.3 *Boussinesq*

description

The Boussinesq data set represents the natural convection generated by a heated cylinder. A stagnant fluid is heated and, after a short time, develops a highly turbulent plume above the cylinder. The flow is a result of a simulation with *Gerris Flow Solver* [122] using Boussinesq approximation. The data is given as a series of 1.600 time steps for $t \in [0, 20]$ of 100×300 spatial grids for $[-0.5, 0.5] \times [-0.5, 2.5]$. The flow field is reconstructed by trilinear interpolation.

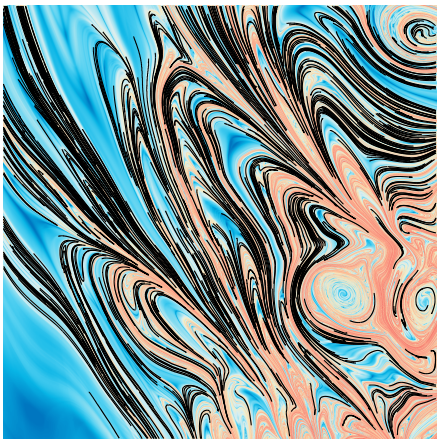


Figure 6.9: BOUSSINESQ RIDGE GEOMETRY. The FTLE field with a part of the extracted ridge geometry with $t = 5$ and $\tau = 15$ for the center left area of the domain. We show the geometry as vector graphics.

FTLE

Figure 6.8 shows FTLE for $t = 5.0$ with $\tau = 10$ (top) and $\tau = 15$ (bottom) for the 'interesting' region $[-0.45, 0.45] \times [-0.45, 0.9]$ of the domain. Figure 6.9 shows a close-up view for $t = 5.0$ and $\tau = 15$ with overlaid ridge geometry extracted with maximum resolution $\epsilon = 7.5 \cdot 10^{-5}$. Due to the quadtree's high memory consumption, we divided the domain into eight equal-sized parts and computed each part individually.

ridge geometry

Please note that we only show the 1000 longest extracted ridges, meaning the most relevant ones for describing the flow. In the FTLE field, there are other ridges visible that are not covered by overlaid geometry lines. The Boussinesq is a highly turbulent flow, which leads to a lot of short and interrupted ridges. We decided not to show these here, to give an uncluttered view of the underlying FTLE field and the geometry of the connected ridges. Furthermore, this example demonstrates the limitations of our ridge extraction technique.

ridge statistics

Figure 6.11 shows statistics for the accumulated ridge length with respect to the integration time τ. The total ridge length shows almost linear increase with a dominance of short and medium-long ridges. This might be the expected behavior for a highly turbulent flow.

6.8.4 *ECMWF Reanalysis*

description

The *European Centre for Medium-Range Weather Forecasts* (ECMWF) provides this data set. It contains a reanalysis simulation of the wind velocity field from April 10th to 19th in 2010 based on the forecast models and previously recorded data. The simulation covers the area from America to Europe in the northern hemisphere (longitude $-120°$ to $60°$ with 1200 cells, latitude $30°$ to $90°$ with 400 cells, height from 0m to 1050m with 90 cells). Due to our algorithm's restriction to 2D, we cut out a single height slice of the dataset. We chose the 40th slice,

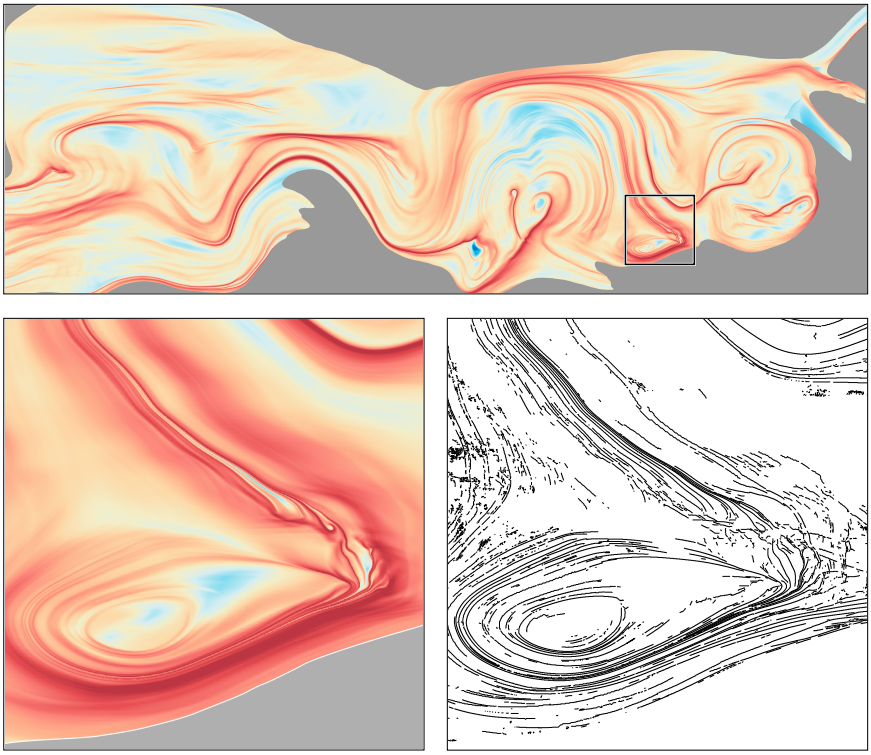


Figure 6.10: ECWMF FLOW. Top: FTLE field for $t = 0$ and $\tau = 10$. Bottom-left: Close-up view of the FTLE field. Bottom-right: Ridge geometry of the close-up area as vector graphics.

which contains the wind velocity field at the height of approximately 470m. This part contains areas with different flow behavior due to characteristic atmospheric features.

FTLE, ridge geometry & ridge statistics

Figure 6.10 shows the FTLE field for the whole domain for this height slice with $t = 0$ and $\tau = 10$ and a close-up with ridge geometry. Areas, where the integrated particles left the domain are marked with a grey mask. Stream-like features can be recognized over the Atlantic Ocean, whereas vortical structures of cyclones are visible over Europe. Figure 6.11 gives the corresponding ridge statistics.

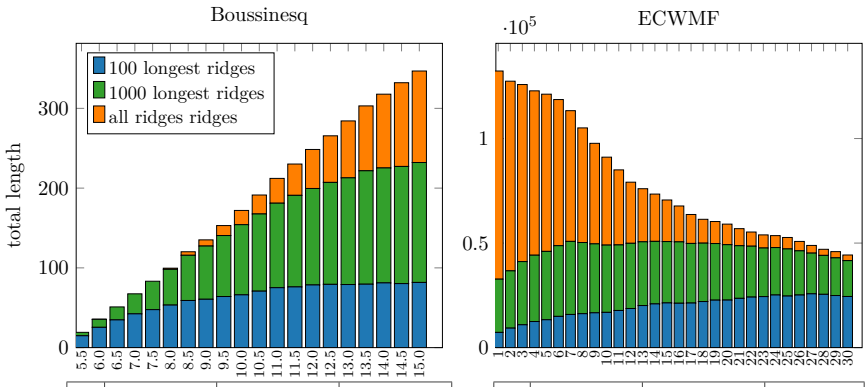
6.8.5 *Timings & Memory Usage*

test system

The algorithm was run on a virtual machine that could access 18 Intel Xeon E5-2690v3 (virtual cores) running at 2.6GHz with 128GB RAM. We used multiple CPUs that could process different parts of the domain – coarse grid cells – in parallel. We add up the times for all parts equivalent to a sequential processing on a single CPU.

data set domains

The Double Gyre was processed in the domain $[0, 2] \times [0, 1]$ and with $t = 0$, $\tau_e = 30$. The Forced Duffing equation was processed in the domain $[-2, 2] \times [-1.5, 1.5]$ with $t = 0$, $\tau_e = 30$. The Boussinesq flow was processed in the domain $[-0.48, 0.48] \times [-0.48, 0.96]$ with $t = 5$, $\tau_e = 15$.



τ	100 longest	1000 longest	all
5.5	14.90	19.17	19.17
7	42.35	67.46	67.46
9	60.64	127.36	135.11
11	75.14	181.10	212.19
13	79.12	212.87	284.25
15	81.74	232.11	346.84

τ	100 longest	1000 longest	all
5	13333.39	46026.55	121213.44
10	16863.88	49034.94	91095.78
15	21377.68	50719.85	70613.11
20	22739.54	49241.72	59067.42
25	24668.56	47200.10	52681.51
30	24458.76	41610.56	44302.59

Figure 6.11: Ridge Statistics for Boussinesq & ECWMF. We list the accumulated ridge lengths over τ. Left: The ridge geometry of the Boussinesq shows almost linear increase with many short and medium-long ridges. Right: The ridge geometry of the ECWMF decreases over time because particles leave the domain and cause undefined FTLE regions.

The ECWMF data set was processed in the domain $[0, 1200] \times [0, 400]$ with $t = 0$, $\tau_e = 30$. Table 6.1 lists the used domain settings for all data sets.

experiment description & timings

For all data sets, we used $\Delta\tau = 1/10$ and chose θ as 1% of $\max \mathrm{FTLE}_{\tau_e}$ that was estimated on the initial coarse grid. Timings are summarized in Table 6.2 for all data sets at maximum resolution ϵ in hours. The columns show times for integrating the flow map ϕ_τ (OpenMP, multi-core), quadtree operations (includes splits, neighbor search, computation, and analysis of FTLE), and ridge extraction, which was performed in *every* time step. Times for ridge extraction appear in two versions – one for the full extraction process and one *without post-processing* (gap-filling). Thus the total time appears also in two columns, the latter excludes the post-processing. As can be expected, times for tree operations depend on the maximum resolution ϵ and the domain's size. Times for flow map integration additionally depend on the integration time. Surprising are the times for ridge extraction that even dominate the total cost for Boussinesq flow regardless of ϵ. A detailed examination revealed that up to 98% of the extraction process is used for filling gaps between ridge clusters. The reason for this is a brute force implementation of gap-filling, which continually checks if a new ridge can be connected to *any* existing ridge. This leads to costly computations if the data

Data Set	Spatial Domain	t	τ
Double Gyre	$[0, 2] \times [0, 1]$	0	30
Forced Duffing	$[-2.0, 2.0] \times [-1.5, 1.5]$	0	20
Boussinesq	$[-0.48, 0.48] \times [-0.48, 0.96]$	5.5	15
ECWMF	$[0, 1200] \times [0, 400]$	0	30

Table 6.1: SPATIAL & TEMPORAL DOMAINS. Listed are the domain settings for the computations for each data set.

Data Set	ϵ	Total	ϕ_τ	Tree	Ridge	w/o Fill
Double Gyre	$7.8 \cdot 10^{-5}$	82	21	49	12	5
Forced Duffing	$7.8 \cdot 10^{-5}$	145	20	81	44	8
Boussinesq	$6.0 \cdot 10^{-4}$	246	1	3	242	4
Boussinesq	$7.5 \cdot 10^{-5}$	247	68	58	121	15
ECWMF	$6.2 \cdot 10^{-2}$	187	89	62	36	10

Table 6.2: TIMINGS. Listed are the computation times in *hours* (rounded) and the maximum resolution for each data set.

contains many short ridges, whose distances furthermore undergo the spatial resolution as for the Boussinesq flow.

memory usage

The peak memory usage was 12GB for the Double Gyre and 20GB for the Forced Duffing. The Boussinesq flow required 8GB and 16GB for $\epsilon = 6 \cdot 10^{-4}$ and $\epsilon = 7.5 \cdot 10^{-5}$, respectively. The ECWMF is the largest data set and required 36GB peak memory usage.

The memory usage primarily depends on the number of cells and flow maps that have to be kept. If memory is limited, one can resort to domain decomposition and sequential processing of parts. We did this, e.g., for the Double Gyre and the Boussinesq. However, this comes at the cost of missing ridge connections at the parts' borders, visible as horizontal 'artifacts' in Figure 6.4.

6.9 DISCUSSION

ridge extraction

For all test data sets, our approach was able to compute high amounts of ridge geometries. The Double Gyre, the Forced Duffing, and the Boussinesq were considered in a comparable domain with side lengths in the order of 1-2 length units. We were able to extract 350 to 500 length units of ridge lines for all three data sets. To the best of our knowledge, no other approach comes even close to extracting such a rich ridge line geometry. This is mainly made possible by a sampling that is tuned to the particular problem: It is based on a sampling of FTLE at the final integration time *and* at intermediate times. Even

though the computing time is relatively high, a dense sampling would have led to prohibitively higher computing times – if the memory limit was not reached before.

ridge statistics

Based on the extraction of the line geometry, ridge statistics for increasing integration times became possible. We do not know any previous publications that compute ridge statistics in this or a similar way. In our opinion, the statistics have the potential to characterize the global behavior of FTLE ridges over time. Hence they could be a future tool for the transition from long integration times to even longer ones. With ongoing integration, distances and sharpness of ridges fall beyond the boundaries of being numerically computable. In this case, a single ridge is of no importance anymore, making the statistics about all ridges the relevant item.

discussion Double Gyre & Forced Duffing

The test data sets show significantly different behavior regarding ridge statistics. The statistics for the Double Gyre and the Forced Duffing are shown in Figure 6.6. Both exhibit an exponential behavior in the total amount of ridges. The Double Gyre has a few ridges that become longer over time – rapidly but smoothly. This reflects the behavior of the underlying flow field, which changes continuously but smoothly over time. The Forced Duffing instead shows three sudden jumps in the summed up ridge lengths. These originate from an abrupt increase of medium-length ridges (green bars). Between these jumps, the changes are negligible. The flow is much more chaotic and comes up with some sudden changes in the flow field.

discussion Boussinesq

The length of the ridge lines grows approximately linearly for the Boussinesq flow. We show its statistics in Figure 6.11 (left). The plot reveals that only a few long ridges (blue bars) do not become longer over time. For additional integration times, we expect the number of short ridges (orange bars) to increase. The Boussinesq flow becomes turbulent after short integration times, explaining the lack or stagnation of long ridges.

discussion ECWMF

Interestingly, the plot of the ECWMF data set shows a different behavior. It is shown in Figure 6.11 (right). Contrary to the other test cases, the total ridge length is vastly higher after short integration times and decreases in the ongoing process. The general decrease can be explained by more and more particles leaving the domain, which leads to fewer areas with well-defined FTLE. The amount of long (blue bars) and medium-long (green bars) ridges becomes stable after a few time steps. However, the amount of short ridges (orange bars) is substantial in the beginning and decreases rapidly afterward. We interpret this as a dominance of the turbulent parts of the flow, which leads to a high number of short ridges at low integration times. After a short time, these turbulent parts become less important. Short ridges vanish or merge and are soon outnumbered by medium-length and long ones.

The constant west-east flow over the Atlantic Ocean becomes more significant in the ongoing process and leads to more elongated ridges.

6.10 LIMITATIONS & FUTURE RESEARCH

limitations

If the flow $\mathbf{v}$ is smooth and differentiable, FTLE is smooth and differentiable as well for a finite integration time. This means that there is only a finite number of ridges, and ridges have finite lengths. It would be desirable to give guarantees that our algorithm finds all of them. However, this cannot be guaranteed because the algorithm depends on numerical integration, which introduces errors, especially for estimating the flow map gradient. This also leads to some separate or unintentionally merged ridge lines. While these numerical artifacts occur to particular lines, they do not impact the ridge statistics.

future research

For future research, an extension to 3D ridge surfaces is desirable. There is no conceptional reason that prohibits the extension. Also, in 3D, ridge surfaces become sharper and closer to each other with increasing integration times, and similarly, an FTLE sampling is required that incorporates intermediate time steps. In order to compute statistics, the FTLE surfaces need to be extracted. We expect this to be the most challenging task for the step to 3D.

7

RECIRCULATION SURFACES

This chapter is based on the publication:

T. Wilde, C. Rössl, H. Theisel
Recirculation Surfaces for Flow Visualization
IEEE Transactions on Visualization & Computer Graphics, 2019

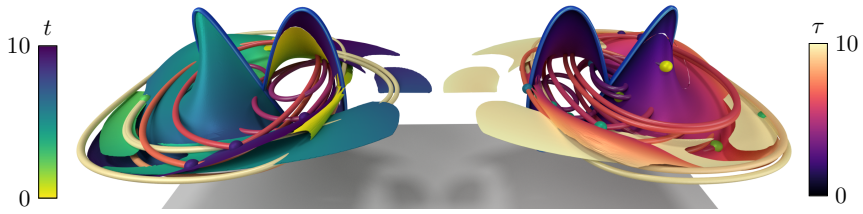


Figure 7.1: Recirculation in the 3D Double Gyre. We show recirculation surfaces and recirculating pathlines. The colors of the surfaces indicate start time t (left) or integration time τ (right). The tubes show a random selection of recirculating pathlines with τ encoded as constant color. The spheres show their start points on the recirculation surface with color-coded t. The boundary curves of the surfaces are defined by the paths of critical points, which are depicted as blue tubes.

In Chapter 6, we introduced a method to compute LCS for 2D unsteady flows. We extracted ridges from FTLE fields to detect LCS. FTLE uses the spatial flow map gradient $\nabla\phi_t^\tau(\mathbf{x})$. Usually, we compute it for a *fixed* starting time t and a *fixed* integration time τ. In the presented approach, we also vary the integration time and incorporate shorter τ values for the computation. Hence, we use several parts of the flow map for the computation – the full spatial domain and parts of one temporal dimension τ. Nevertheless, due to the *fixed* starting time t, a large part of the flow map is *not* utilized for the feature extraction.

recirculation surfaces

In this chapter, we present a new flow feature – *recirculation surfaces*. Recirculation surfaces are the first comprehensive flow feature that uses the *full* 5D flow map of a 3D unsteady flow. Recirculation surfaces are the loci where massless particle integration returns to its starting point after some variable, finite integration. They present a formal approach to the visual analysis of recirculation in flows.

chapter overview

This chapter has the following structure. Section 7.1 gives a short introduction to the phenomenon of recirculation and possible meanings. In Section 7.2, we discuss related work considering recirculation and standard approaches we need for our algorithm, e.g., the extraction of critical points from vector fields. In Section 7.3, we give a rigorous definition of recirculation surfaces as 2-manifolds embedded in 5D space and study their properties. We construct an algorithm for their extraction that we present in Section 7.4. Finding isolated closed orbits in steady vector fields occurs as a particular case of recirculation surfaces. We discuss this in Section 7.5. Section 7.6 presents the results we obtained from different data sets, followed by a discussion of our approach in Section 7.7. Section 7.8 lists the limitations of our method and concludes the chapter with possible future research.

chapter remarks

This chapter uses some special notations and abbreviations, which we introduce at the corresponding places. We will also repeat some elements already introduced if we think they will help to follow the content of this chapter.

7.1 INTRODUCTION

Recirculation is a phenomenon that influences many effects in 3D unsteady flows. Several approaches study recirculation or other phenomena in the presence of recirculation. Recirculation has been covered in quite diverse areas, from biology, chemistry, and physics to Flow Visualization.

recirculation meaning

There seems to be a common intuitive understanding of what recirculation is. The term may refer to locations where moving particles return to their starting locations after a specific time. Alternatively, recirculation describes regions of particles that do not leave a particular region – but may move within this region. Recirculation can also describe areas where particles move 'backwards' or 'against the main flow direction,' which requires a proper definition of what 'backwards' means. For steady flows, recirculation is related to closed orbits. Surprisingly, there seems to be no unique standard definition of the concept of recirculation — a property that recirculation shares, e.g., with the concept of a vortex.

7.2 RELATED WORK

Recirculation is an important phenomenon that has been studied in various application domains. This section provides a brief review of related work on recirculation and algorithms that are the basis for our approach.

7.2.1 *Recirculation as a Phenomenon*

physics

In physics and computational fluid mechanics, recirculation can be essential for controlling the flow in specific manners. For instance, Gautier et al. [55] use a machine learning approach to reduce the recirculation zone of a backward-facing step flow, and Debien et al. [31] apply genetic programming for a flow of the turbulent boundary layer downstream from a sharp edge ramp. Brunton and Noack [18] give a recent overview of turbulence control. Tadmor and Banaszuk [155] track a reference orbit in a combustor recirculation zone to control vortex motion. Chen et al. [26] and Laramee et al. [89] identify recirculating flow behavior as a feature for describing combustion processes in an engine. Noack et al. [112] optimize coarse-scale mixing in a recirculation zone with a simple vortex model by Suh [154].

medicine & biology

Shadden et al. [147] analyze aortic blood flow using LCS that separate regions of recirculation flow. Mittasch et al. [106] introduce a method for influencing cytoplasmic streaming and identify intracellular flows that show recirculating behavior.

Flow Visualization

Recirculation and related phenomena have been studied in Flow Visualization. Weinkauf et al. [172] present a method to extract cores of particles with swirling motion. Peikert and Sadlo [118] study vortex rings in recirculating flows. Sanderson et al. [137] extract orbits for a magnetic field by exploiting a natural Poincaré section.

7.2.2 *Isolated Closed Orbits for Steady Vector Fields*

closed orbits in 2D

For steady vector fields, recirculation is related to closed orbits. There exist several methods for finding closed orbits in 2D fields. Wischgoll and Scheuermann [179] provide the first approach to the detection of closed streamlines in planar flows. Theisel et al. [159] compute closed orbits in 2D by intersecting 3D stream surfaces. Morse decomposition provides a combinatorial approach to vector field topology and is used to detect closed streamlines in a series of articles by Chen et al. [26, 27]. Existing 2D techniques make use of the fact that closed orbits have either a source or a sink behavior. Their extension to 3D is straightforward as long as the closed orbits show source or sink behavior [127].

closed orbits in 3D

In 3D, however, a closed orbit can also have a saddle behavior, i.e., almost all streamlines in the closed orbit's neighborhood move away under both forward and backward integration [3, 116]. For such orbits, the known 2D techniques do not carry over. The only existing solution so far is by Kasten et al. [79]: The intersections of the closed orbit with a plane are obtained by computing the intersection of FTLE ridges in forward and backward integration.

7.2.3 *Extraction of Isolated Critical Points*

analytic methods

3D steady vector fields generally have isolated critical points. We can compute them analytically for a piecewise linear field as a solution system of three linear equations. For piecewise trilinear vector fields, an analytic solution does not exist. Mann and Rockwood [100] utilize an octree-like approach and divide the domain into cells. For each cell, an index is computed, which decides if the cell contains a critical point. If so, the cell gets subdivided. An approach designed explicitly for electric fields defined by a set of point charges was presented by Max and Weinkauf [103].

recursive method

Weinkauf [171] uses a recursive algorithm to locate critical points. The values at all vertices spanning a trilinear cell are examined. If all values share the same sign, a critical point cannot occur in the cell, and the recursion stops. Otherwise, the cell is subdivided, and each subcell is examined recursively until a certain threshold is reached, i.e., the critical point's location is bounded within the smallest subcell. This sign test is easy to implement, fast to compute, and works for arbitrary dimensions. For instance, one must compare the signs for the (x, y, z)-components individually to use it for 3D vectorial data. This algorithm typically generates multiple 'candidates' for the same root, which is resolved by a local clustering. We use the same approach to locate critical points in derived vector fields.

7.2.4 *Tracking Critical Points*

Several methods for tracking critical points exist. Tricoche et al. [161] consider piecewise linear 2D vector fields and compute and connect the critical points on the faces of a prism cell structure obtained by a tetrahedral grid. Garth et al. [53] give an extension to 3D flows defined piecewise on a tetrahedral partition with linear interpolation in both space and time. An adaptation to quadrilinear vector fields, i.e., linear interpolation in each spatial dimension and time, is straightforward. In this case, the restriction to cell boundaries of 4D hypercubes gives 3D trilinear vector fields. Then the extraction of critical points on hypercube faces can proceed as described above. An alternative class of algorithms for tracking critical points uses feature flow fields [157], i.e., they construct space-time vector fields with tangent curves corresponding to the critical points' searched paths. Klein et al. [84] use this approach to track critical points in scale space. Weinkauf et al. present an out-of-core version [174] and a stabilized version [175] of feature flow fields.

7.3 DEFINITION OF RECIRCULATION SURFACES

velocity field & flow map

We consider a 3D time-dependent velocity field $\mathbf{v}(\mathbf{x}, t)$, and the corresponding flow map $\phi_t^\tau(\mathbf{x})$:

$$\mathbf{v} : D \times T_t \to \mathbb{R}^3 \,,$$
$$\phi : D \times T_t \times T_\tau \to D \,.$$

The spatial domain is denoted by D, the temporal intervals are denoted by T_t and T_τ. We define the vector field and the flow map in vectorial space, i.e., $D \subseteq \mathbb{R}^3$, $T_t \subseteq \mathbb{R}$, $T_\tau \subseteq \mathbb{R}$.

$\mathbf{x}$ & $\bar{\mathbf{x}}$

Therefore, the flow map $\phi_t^\tau(\mathbf{x})$ is defined in the 5D space $(\mathbf{x}, t, \tau)^{\mathrm{T}}$. In order to search for recirculation, we have to explore this space. When we refer to elements in this 5D space, we indicate this by a double-bar notation, i.e., $\bar{\bar{\mathbf{x}}} = (\mathbf{x}, t, \tau)^{\mathrm{T}} \in \mathbb{R}^5$.

Y & $\bar{\bar{Y}}$

In the following, the capital letter Y denotes implicitly defined recirculation surfaces. We use the double-bar notation similarly to distinguish between recirculation surfaces $Y \subset \mathbb{R}^3$ in 3D space, and $\bar{\bar{Y}} \subset \mathbb{R}^5$ in 5D space.

0 & $\mathbf{0}$

Furthermore, we want to emphasize that 0 denotes the scalar value, the bold version $\mathbf{0}$ denotes the zero vector, and $\mathbf{I}$ denotes the identity matrix.

7.3.1 *Definition*

definition recirculation surface

We describe a recirculation surface implicitly as the set of all 5D points, for which the flow map gives the identity, i.e., the loci of particles that return to their starting point after a finite integration time. Therefore, we define a recirculation surface in 5D space as:

$$\bar{\bar{Y}} = \left\{ (\mathbf{x}, t, \tau)^{\mathrm{T}} \in D \times T_t \times T_\tau \;\middle|\; \frac{\phi(\mathbf{x}, t, \tau) - \mathbf{x}}{\tau} = \mathbf{0} \;\wedge\; \tau \neq 0 \right\} . \tag{7.1}$$

The projection of $\bar{\bar{Y}}$ from 5D space to Y in 3D (spatial) space is given by:

$$Y = \left\{ \mathbf{x} \in D \;\middle|\; (\mathbf{x}, t, \tau)^{\mathrm{T}} \in \bar{\bar{Y}} \right\} .$$

denominator τ

Figure 7.2 illustrates a recirculation surface. Note that the dependence on the denominator τ in Equation (7.1) is necessary to treat the case $\tau \to 0$ properly. Without the term $1/\tau$, any point in the 5D domain would be part of a recirculation 'surface' for vanishing integration time $\tau \to 0$, which is clearly undesired. With the term included, there exists a well-defined limit surface for $\tau \to 0$, as will be explained in Section 7.3.3.

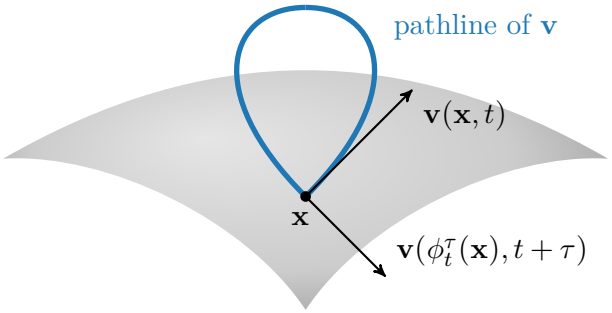


Figure 7.2: ILLUSTRATION OF A RECIRCULATION SURFACE. Starting a pathline from a surface point $\mathbf{x}$ returns to $\mathbf{x}$ after a finite integration time. The flow direction is not necessarily the same. The illustration shows the projection of $\overline{\overline{Y}}$ in 5D space to Y in 3D space.

7.3.2 *Properties*

The following sections require some concepts from differential geometry of manifolds, e.g., the tangent space, and concepts from linear algebra, e.g., the null space and its basis. Furthermore, we apply multivariate calculus and provide non-trivial derivations. For more information, we refer to the literature by Do Carmo [32, 33] and Strang [153].

null space & tangent space

The *null space* N of a matrix $\mathbf{A}$ is defined as

$$N(\mathbf{A}) = \{\, \mathbf{x} \;\mid\; \mathbf{A}\cdot\mathbf{x} = \mathbf{0} \,\}\,.$$

Let $\mathcal{M}$ be a differentiable manifold. The set of all vectors at a location $\mathbf{x} \in \mathcal{M}$ that are tangent to $\mathcal{M}$ build a vector space – it is called the *tangent space* $T_{\mathbf{x}}\mathcal{M}$.

ϕ,$\mathbf{v}_2$,$\mathbf{G}$

To keep the notation clear, we will use the following abbreviations:

$$\begin{aligned} \phi &:= \phi(\mathbf{x},t,\tau)\,, \\ \mathbf{v}_2 &:= \mathbf{v}(\phi,t+\tau) = \nabla\phi\cdot\mathbf{v} + \phi_t\,, \\ \mathbf{G} &:= \nabla\phi - \mathbf{I}\,. \end{aligned} \tag{7.2}$$

Note that the definition of $\mathbf{v}_2$ in Equation (7.2) uses a property of the flow map that we prove in Appendix C. The symbols $\mathbf{v}_2$ and $\mathbf{G}$ will be used literally as 'abbreviations' to facilitate expressions and comparing coefficients/factors in expressions.

distance function $\mathbf{s}$

Equation (7.1) defines recirculation surfaces as the set of all 5D points in the flow map that map to their starting point. We consider the following vectorial function to identify locations with this behavior:

$$\mathbf{s}(\mathbf{x},t,\tau) = \frac{\phi - \mathbf{x}}{\tau}\,. \tag{7.3}$$

The zeros of $\mathbf{s}$ define the recirculation surface, i.e., $\mathbf{s}(\mathbf{x},t,\tau) = \mathbf{0}$. We remark that the denominator τ is necessary for the same reason as for the definition of $\overline{\overline{Y}}$ in Equation (7.1).

derivatives of $\mathbf{s}$

To study the properties of $\mathbf{s}$, we consider its (spatial) directional derivative in direction $\mathbf{r}$, which we obtain by elementary differentiation rules:

$$\frac{\mathrm{d}\mathbf{s}}{\mathrm{d}\mathbf{r}} = \nabla\mathbf{s}\cdot\mathbf{r} = \frac{\nabla\phi\cdot\mathbf{r}-\mathbf{r}}{\tau} = \frac{\mathbf{G}\cdot\mathbf{r}}{\tau}\ .$$

Furthermore, the partial derivatives of $\mathbf{s}$ with respect to starting time t and integration time τ are defined as:

$$\frac{\partial\mathbf{s}}{\partial t} = \frac{\phi_t}{\tau}\,,$$
$$\frac{\partial\mathbf{s}}{\partial\tau} = \frac{\mathbf{v}_2-\mathbf{s}}{\tau}\,.$$

This allows us to assemble the 5D gradient $\nabla\mathbf{s}$ as the matrix

$$\mathbf{M} \;:=\; \nabla\mathbf{s} \;=\; \frac{1}{\tau}\left[\;\mathbf{G}\;\middle|\;\phi_t\;\middle|\;\mathbf{v}_2-\mathbf{s}\;\right] \;\in\; \mathbb{R}^{3\times 5}\ . \tag{7.4}$$

tangent space

The recirculation surface $\overline{\overline{Y}}$ is defined implicitly as a point set. We study the neighborhood of a point $\bar{\bar{\mathbf{x}}}_0 \in \overline{\overline{Y}}$ by constructing the *tangent space*

$$T_{\bar{\bar{\mathbf{x}}}_0}\overline{\overline{Y}} = \{\,\bar{\bar{\mathbf{x}}} \mid \mathbf{M}(\bar{\bar{\mathbf{x}}}-\bar{\bar{\mathbf{x}}}_0) = \mathbf{0}\,\}\,, \tag{7.5}$$

which provides a local linear approximation to $\overline{\overline{Y}}$.

$\overline{\overline{Y}}$ is a 2-manifold

The null space of $\mathbf{M}$ provides a basis for this affine subspace of $\mathbb{R}^5$. We generally expect $\mathbf{M}$ to have full rank, i.e., $\mathrm{rank}(\mathbf{M}) = 3$. Then the null space has dimension $5-3=2$, and $\overline{\overline{Y}}$ is indeed locally a 2-manifold, which justifies the term recirculation *surface*.

$\overline{\overline{Y}}$ is structurally stable

Local configurations where $\mathbf{M}$ has a smaller rank are structurally unstable. With a small perturbation, like adding some noise, all singular values will become non-zero and $\mathbf{M}$ is 'regularized' to have full rank. Note that the null space dimension can only grow with a rank deficit of $\mathbf{M}$. In particular, the dimension cannot become less than 2.

null space basis

We remark that the QR-decomposition $\mathbf{M}^{\mathrm{T}} = \mathbf{QR}$ provides an efficient way to compute an orthonormal basis of the null space of $\mathbf{M}$. The first two columns of the orthogonal matrix $\mathbf{Q}$ span the null space and hence the tangent space of $\overline{\overline{Y}}$ at $\bar{\bar{\mathbf{x}}}_0$. While this is more efficient than a singular value decomposition of $\mathbf{M}$ (or likewise a spectral decomposition of $\mathbf{M}^{\mathrm{T}}\mathbf{M}$), it assumes full rank, whereas singular values (or eigenvalues) would reveal a rank deficit directly.

'project' points to $\overline{\overline{Y}}$

The norm of the vector function $\mathbf{s}$ gives a pseudo-distance to the recirculation surface. This can be used for the 'projection' of points onto the surface using a minimization approach. The gradient of the squared distance can be expressed as:

$$\frac{1}{2}\nabla||\mathbf{s}||^2 \;=\; \frac{1}{2}\Big((\nabla\mathbf{s})^{\mathrm{T}}\mathbf{s}+\mathbf{s}^{\mathrm{T}}(\nabla\mathbf{s})\Big) \;=\; \nabla\mathbf{s}^{\mathrm{T}}\mathbf{s} \;=\; \mathbf{M}^{\mathrm{T}}\mathbf{s}\ . \tag{7.6}$$

This allows moving points $\bar{\bar{\mathbf{z}}}_0 \in \mathbb{R}^5$ that are close to the surface towards and onto the surface. In the simplest case, we can use gradient descent. Also, a more sophisticated nonlinear solver that minimizes, e.g.,

$$||\mathbf{s}(\bar{\bar{\mathbf{z}}})||^2 + \mu||\bar{\bar{\mathbf{z}}} - \bar{\bar{\mathbf{z}}}_0||^2 ,$$

can be used. Here the weighted term accounts for finding the surface point $\bar{\bar{\mathbf{z}}}$ with the shortest (Euclidean) distance to $\bar{\bar{\mathbf{z}}}_0$, i.e., get close to a projection.

Standard solvers can use the gradient vector $\nabla||\mathbf{s}||^2$, and the 'closed form' of Equation (7.6) instead of a finite differences approximation. We note that the computation of $\nabla\phi$ and hence $\mathbf{s}$ typically involves finite differences in 3D.

7.3.3 *The Particular Case $\tau \to 0$*

The case $\tau = 0$ was excluded from the definition of recirculation surfaces. This is due to the denominator τ in the definition. Without this term, any domain point would be part of a recirculation 'surface' for $\tau \to 0$. Also, a numerical root-finding may fail for small τ as the absolute difference $\phi(\mathbf{x}, t, \tau) - \mathbf{x}$ becomes arbitrarily small.

the case $\tau \to 0$

For the given definition of recirculation surfaces based on the scaled distance in Equation (7.3) there is a well-defined limit for $\tau \to 0$ as:

$$\begin{aligned} \lim_{\tau\to 0} \mathbf{s} &= \lim_{\tau\to 0} \frac{\phi - \mathbf{x}}{\tau} \\ &= \lim_{\tau\to 0} \frac{\phi(\mathbf{x}, t, \tau) - \phi(\mathbf{x}, t, 0)}{\tau} \\ &= \frac{\partial \phi}{\partial \tau} \\ &= \mathbf{v}(\mathbf{x}, t) . \end{aligned} \tag{7.7}$$

and

$$\begin{aligned} \lim_{\tau\to 0} \frac{\mathbf{G}}{\tau} &= \lim_{\tau\to 0} \frac{\nabla\phi - \mathbf{I}}{\tau} \\ &= \lim_{\tau\to 0} \frac{\nabla\phi(\mathbf{x}, t, \tau) - \nabla\phi(\mathbf{x}, t, 0)}{\tau} \\ &= \nabla\mathbf{v}(\mathbf{x}, t) . \end{aligned}$$

Furthermore, we can derive

$$\lim_{\tau\to 0} \mathbf{M} = \left[\begin{array}{c|c|c} \nabla\mathbf{v} & \mathbf{v}_t & \nabla\mathbf{v}\cdot\mathbf{v} + \mathbf{v}_t \end{array} \right] .$$

Equation (7.7) states that isolated critical points of the vector field $\mathbf{v}$ are on the recirculation surface Y, i.e.,

$$\mathbf{v}(\mathbf{x}, t) = \mathbf{0} \quad \Rightarrow \quad (\mathbf{x}, t, 0)^{\mathrm{T}} \in \bar{\bar{Y}} . \tag{7.8}$$

7.3.4 *Properties of the 3D Surface Y*

open surface Y

We have shown that $\overline{\overline{Y}}$ is, in general, a closed implicit surface in 5D. However, its projection $Y \subset \mathbb{R}^3$ is an *open* surface that is circumscribed by boundary curves. Consider a location $(\mathbf{x}, t, \tau)^{\mathrm{T}}$ that belongs to the surface $\overline{\overline{Y}}$. A particle seeded at $(\mathbf{x}, t)$ reaches its seeding position $\mathbf{x}$ again after the integration time τ. If we now take the same particle at its destination $(\mathbf{x}, t+\tau)$ and integrate it backward, it shows the same behavior. Both points must be on the surface, i.e.,

$$(\mathbf{x}, t, \tau)^{\mathrm{T}} \in \overline{\overline{Y}} \iff (\mathbf{x}, t+\tau, -\tau)^{\mathrm{T}} \in \overline{\overline{Y}}.$$

boundary curves

This means that almost every point on Y corresponds to *two* original points on $\overline{\overline{Y}}$. The points on $\overline{\overline{Y}}$ where both originals collapse in a single point build the *boundary curves* of Y. Since this occurs for $\tau = 0$, the paths of critical points of $\mathbf{v}$ are the boundary curves of Y due to Equation (7.8).

In general, $Y \in \mathbb{R}^3$ is locally manifold. However, the projection $\mathbb{R}^5 \to \mathbb{R}^3$ may result in *self-intersecting* surfaces and possibly 'degenerate' (locally) to a curve. Both effects are visible in Figure 7.6.

7.4 EXTRACTION OF RECIRCULATION SURFACES

In this section, we describe an algorithm for the extraction of recirculation surfaces. When we consider the properties of $\overline{\overline{Y}}$, there are several starting points for constructing such an algorithm.

gradient-based methods

One possible approach is using a *brute force sampling* and moving from random seed points in 5D onto $\overline{\overline{Y}}$ using $||\mathbf{s}||^2$ and its gradient from Equation (7.6). This works reasonably only for seeds that are close enough to the surface. The generation of the seeds should be adjusted, e.g., with some sort of importance sampling. The collected surface points can be triangulated or rendered as splats – similar to a particle-based extraction of 3D ridge surfaces presented by Kindlmann [80].

A different approach would start from a seed point $\overline{\overline{\mathbf{x}}}_0 \in \overline{\overline{Y}}$ and apply a *surface growing* algorithm that propagates a front over the surface by 'moving' in the tangent space followed by a back-projection onto the surface. The propagating front would expand, e.g., a triangulation, and possibly split and merge.

problems with $\nabla\phi$

Unfortunately, both approaches become numerically unstable, especially for longer integration times, because both rely on the flow map gradient $\nabla\phi$. We have seen in Chapter 6, that $\nabla\phi$ grows for longer integration times and tends to show 'extreme behavior' in the sense that its norm grows exponentially.

For this reason, we introduce an extractor that requires only the flow map but *not its gradient*. The algorithm is based on a repeated search

of isolated critical points in steady 3D vector fields – a common problem with well-established and robust numerical solutions.

7.4.1 *Algorithm Overview*

main idea

The core algorithm aims at finding all intersections of Y with straight lines parallel to the coordinate axes. For instance, we define the line

$$l = \{ (x, y, z)^{\mathrm{T}} \mid x = \hat{x},\, y = \hat{y} \},$$

that is parallel to the z-axis for fixed $\hat{x}$, $\hat{y}$. We reduce the problem to finding intersections of Y with this line l. For this restriction, we extract isolated critical points of the 3D vector field

$$\mathbf{s}_{\hat{x},\hat{y}}(z, t, \tau) := \mathbf{s}((\hat{x}, \hat{y}, z)^{\mathrm{T}}, t, \tau).$$

If we find a critical point $\mathbf{s}_{\hat{x},\hat{y}}(z_0, t_0, \tau_0) = \mathbf{0}$, then Y intersects the line in $(\hat{x}, \hat{y}, z_0)^{\mathrm{T}}$. Similarly, intersections of Y with lines parallel to the x- or the y-axis can be computed. We decompose the spatial domain using a uniform grid and apply this search along all grid lines.

algorithm overview

We summarize the algorithm for the extraction of Y from $\mathbf{v}$ into the following steps:

1. Sample $\mathbf{s}$ on a regular grid and assume a piecewise quintilinear (i.e., linear in every of the five dimensions) interpolation inside the grid cells in 5D.
2. Find the intersections of Y with all 3D grid lines in x-, y-, and z-directions.
3. Extract the boundary curves of Y by tracking the critical points of $\mathbf{v}(\mathbf{x}, t)$.
4. Visualize the sampled surface Y.

Figure 7.3 gives an illustration of the first two steps of the extraction algorithm. We explain the steps in the following in more detail.

7.4.2 *Step 1 - Sample* $\mathbf{s}$ *on a Regular Grid*

regular grid in $(\mathbf{x}, t, \tau)^{\mathrm{T}}$

In Step 1, we cover the domain of $\mathbf{s}$ with a regular grid. The distance function $\mathbf{s}$ is defined in a domain in space-double-time, i.e., $(\mathbf{x}, t, \tau)^{\mathrm{T}}$. Therefore, we must define the sampling grid also in space-double-time.

A complete sampling of $\mathbf{s}$ in 5D is challenging, both in terms of computation time and memory requirements. The choice of the grid resolution should reflect a trade-off between accuracy and performance. A reasonable starting point for a suitable grid resolution is the original grid on which $\mathbf{v}$ is given. However, a denser sampling grid for $\mathbf{s}$ gives better results because $\mathbf{s}$ collects information over several grid cells of $\mathbf{v}$.

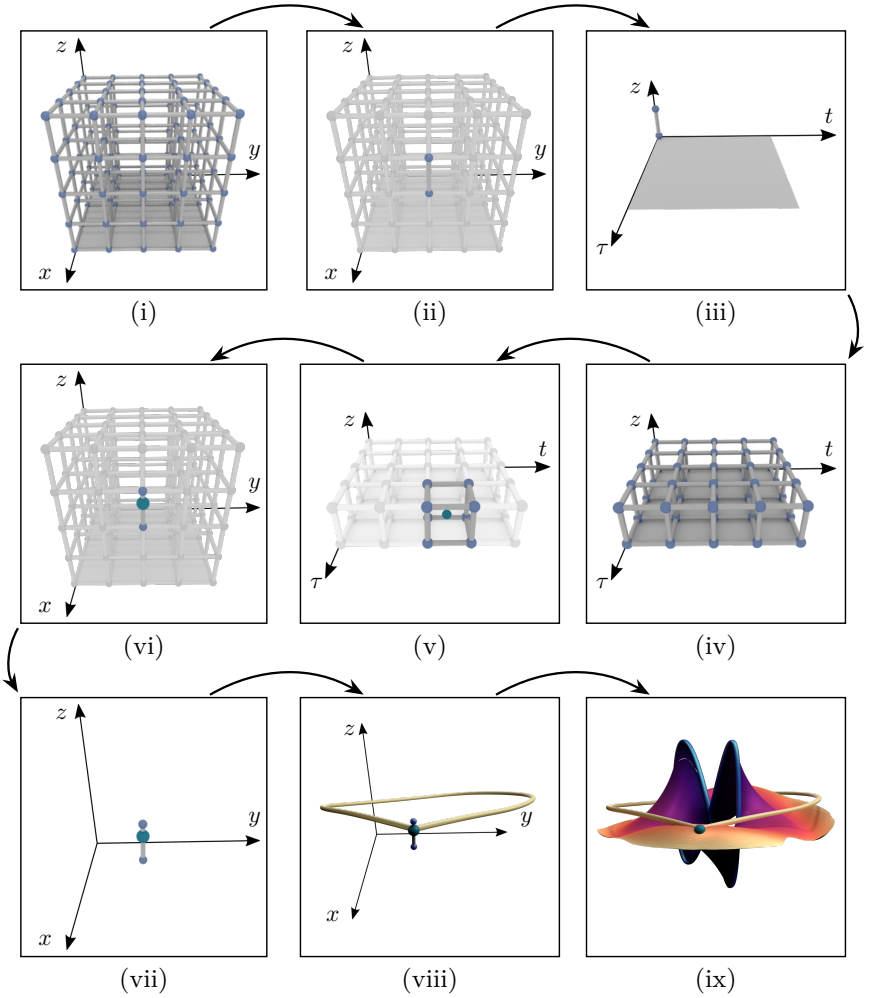


Figure 7.3: ILLUSTRATION OF THE EXTRACTION ALGORITHM.
(i) Sample the spatial space with an axis aligned grid.
(ii) Select a single line l by fixing $\hat{x}$, $\hat{y}$.
(iii) Line l spans a 3D space in $(z, t, \tau)^{\mathrm{T}}$.
(iv) Consider $\mathbf{s}_{\hat{x},\hat{y}}(z, t, \tau) \in \mathbb{R}^3$ as a piecewise trilinear vector field over a regular grid.
(v) Locate critical point at (z_0, t_0, τ_0) in this vector field.
(vi+vii) $((\hat{x}, \hat{y}, z_0), t_0, \tau_0)^{\mathrm{T}}$ is a single point on Y (and $\overline{\overline{Y}}$).
(viii) The correpsonding pathline shows recirculating behavior.
(ix) The location belongs to a recirculation surface.

To save memory, we use an on-the-fly computation of $\mathbf{s}$. Instead of a precomputation of $\mathbf{s}$ on the whole 5D grid, we compute each line l defined by two varying (spatial) grid samples individually and go on with Step 2.

7.4.3 *Step 2 - Locate Intersection Between Line l & Y*

intersect l and Y

We assume a multilinear interpolation in Step 1. Then the restriction to a grid line l with $x = \hat{x}, y = \hat{y}$ yields $\mathbf{s}_{\hat{x},\hat{y}}(z, t, \tau) \in \mathbb{R}^3$ as a piecewise

trilinear vector field over a regular grid. To find a intersection between Y and the line l, we have to search for critical points in this vector field. Finding its isolated critical points is a solved standard problem in Flow Visualization.

recursive **0** *search*

We use the approach described by Weinkauf in [171] that we briefly explained in Section 7.2. We check the vertices that define the current cell of the vector field $\mathbf{s}_{\hat{x},\hat{y}}(z,t,\tau)$. If each of the components x, y, and z appears with positive and negative values, the cell might contain a critical point. We subdivide the cell and repeat the test recursively until

- one component contains the same sign for all vertices, i.e., no critical point was found;
- a desired (small) cell size is reached.

In the second case, the cell center of the last recursion step is treated as critical point location.

arbitrarily oriented lines l

Since Y is double folded in 3D, we can restrict the search to positive integration times τ. Furthermore, we restrict the search space to grid lines that are parallel to coordinate axes. The algorithm would work similarly with a restriction to *any* line $(1-\alpha)\,\mathbf{x}_1 + \alpha\,\mathbf{x}_2$ in 3D.

7.4.4 *Step 3 - Extract the Boundary Curves of Y*

critical points for boundary curves

In Section 7.3.4 we explained, that the movements of critical points in $\mathbf{v}$ yield the boundary curves of Y. Therefore, we have to track the movement of critical points in the vector field $\mathbf{v}$ to determine Y's boundary. This is another solved standard problem in Flow Visualization. We use an approach similar to [53] for a piecewise quadri-linear vector field.

We consider each 4D hypercube $[x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}] \times [t_\ell, t_{\ell+1}]$ in which a quadri-linear interpolation of $\mathbf{v}$ is assumed. In order to search for the intersections with the paths of critical points, we search the isolated critical points in the 8 boundary cubes of the hypercube:

$$
\begin{aligned}
&[y_j, y_{j+1}] \times [z_k, z_{k+1}] \times [t_\ell, t_{\ell+1}] \quad \text{at} \quad x = x_i \\
&[y_j, y_{j+1}] \times [z_k, z_{k+1}] \times [t_\ell, t_{\ell+1}] \quad \text{at} \quad x = x_{i+1} \\
&[x_i, x_{i+1}] \times [z_k, z_{k+1}] \times [t_\ell, t_{\ell+1}] \quad \text{at} \quad y = y_j \\
&[x_i, x_{i+1}] \times [z_k, z_{k+1}] \times [t_\ell, t_{\ell+1}] \quad \text{at} \quad y = y_{j+1} \\
&[x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [t_\ell, t_{\ell+1}] \quad \text{at} \quad z = z_k \\
&[x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [t_\ell, t_{\ell+1}] \quad \text{at} \quad z = z_{k+1} \\
&[x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}] \quad \text{at} \quad t = t_\ell \\
&[x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}] \quad \text{at} \quad t = t_{\ell+1}\,.
\end{aligned}
$$

We find the critical points of $\mathbf{v}$ in each subcube by the same algorithm as in Step 2. To describe the boundary curve of Y, we connect the points by a simple heuristic preferring spatially close points to be connected.

7.4.5 *Step 4 - Visualize the Recirculation Surface Y*

visualize Y

Step 3 generates point samples on $\overline{\overline{Y}}$. In Step 4 we triangulate their projection to 3D space. The resulting triangle mesh is a piecewise linear approximation to Y, which can be rendered with an additional color-coding of either t or τ.

7.4.6 *Surface Reconstruction*

surface Y from samples

Surface reconstruction from point samples is a non-trivial problem that is well-studied in Computer Graphics. However, there are no methods readily available for the triangulation of $\overline{\overline{Y}}$, a closed 2-manifold in 5D. Likewise, reconstruction algorithms for closed or open surfaces in 3D typically assume 2-manifolds, i.e., they cannot deal with self-intersections and degenerate situations that may occur due to the projection $\overline{\overline{Y}} \in \mathbb{R}^5 \rightarrow Y \in \mathbb{R}^3$.

Also, an assumption on either the sampling rate, e.g., in terms of Nyquist frequency, or equivalently the minimum local feature size is required to guarantee an accurate reconstruction. Unfortunately, we cannot guarantee a sufficient sampling of either $\overline{\overline{Y}}$ or Y. Ideally, we would like to work in the original sampling space and reconstruct by triangulation of the samples. However, we note that the projection to a 3D triangle mesh that approximates Y may locally degenerate due to topology changes, which, e.g., may lead to overlapping triangles, and due to the unacceptable 'distortion' from the projection.

We leave surface reconstruction of $\overline{\overline{Y}}$ and Y with guarantees as an open problem and do not claim any novel contribution.

ball pivoting

In our experiments, we used the *ball pivoting* algorithm [7] to triangulate Y. This standard surface reconstruction method served well for our purpose for two reasons:

1. It can handle surfaces with boundaries.
2. It propagates a 'front' on the surface, which allows for partial processing of surface patches defined by specific t and τ intervals.

The second reason is a simple way to deal with self-intersections (or likewise surface sheets that get very close to each other). They are avoided by generating the intersecting patches sequentially and combining them into a single surface. While this works reasonably well in practice, there is of course no guarantee. Our results show that – as expected – the reconstruction suffers from local undersampling, e.g., in regions of high

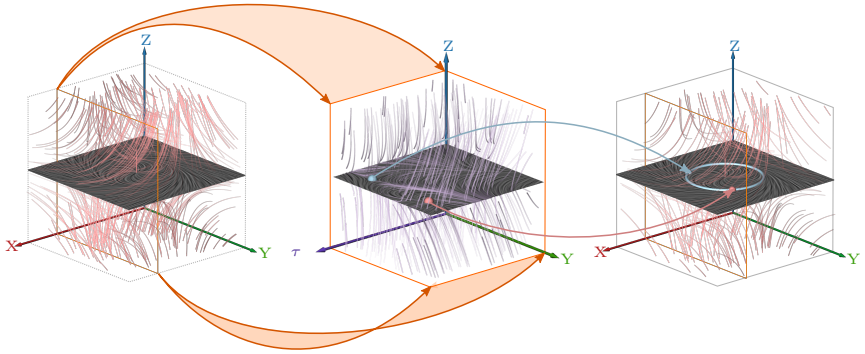


Figure 7.4: CLOSED STREAMLINES FOR THE CLOSED ORBIT FLOW. Left: Place a fixed plane in $x = \hat{x}$ (orange rectangle). Center: Compute $\mathbf{s}(\hat{x}, y, z, \tau)$ for this plane and extract critical points. Right: Reproject the critical points back into the plane $x = \hat{x}$ and integrate closed streamline.

curvature. Figure 7.14 shows an extreme example. We note, that ball pivoting may be a promising candidate for an extension to 5D. However, it is unclear whether a denser sampling is required. The presented algorithm does not provide an adaptive sampling. While possible in principle, a proper adaptation is non-trivial as it must adapt to the differential geometry of a 2-manifold embedded in 5D. Nevertheless, a naïve dense sampling of the search space is too expensive, making the use of a multiresolution approach mandatory.

refine sampling

We apply a simple modification to the algorithm that introduces two fixed sampling resolutions. An initial, coarser sampling grid is used to detect the presence of recirculation. Whenever a candidate sample is found on an edge of this grid, we sample a local region at a finer resolution. In our experiments, the local region's size is twice the cell size in every dimension, and the sampling rate is quintupled in space and doubled in time dimensions.

7.5 ISOLATED CLOSED STREAMLINES IN 3D STEADY VECTOR FIELDS

isolated closed streamline

For 3D steady vector fields $\mathbf{v}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x})$, isolated closed streamlines are a particular (and degenerate) case of recirculation surfaces. In the steady case, the flow map simplifies to $\phi(\mathbf{x}, \tau)$, and the vector field from Equation (7.3) writes as $\mathbf{s}(\mathbf{x}, \tau)$.

algorithm steps

We adapt the extraction of recirculation surfaces and provide an algorithm for finding isolated closed streamlines:

1. Specify a plane perpendicular to one of the coordinate axes, e.g., the plane $x = \hat{x}$ for some constant $\hat{x}$.

2. Determine the intersections of closed streamlines with the defined plane. Therefore, search for the isolated critical points in the

3D vector field $\mathbf{s}_{\hat{x}}(y, z, \tau) = \mathbf{s}(\hat{x}, y, z, \tau)$. If $(y_0, z_0, \tau_0)^{\mathrm{T}}$ is such a point, then the point $(\hat{x}, y_0, z_0)^{\mathrm{T}}$ lies on the closed streamline of $\mathbf{v}$. Note that in this case, the closed streamline has at least two intersections with the plane.

3. Integrate $\mathbf{v}$ from $(\hat{x}, y_0, z_0)^{\mathrm{T}}$ to get the complete closed streamline. Repeat Steps 1 and 2 to find more seeds on the closed streamline if the integration time is too large for 'stable' integration. Due to numerical inaccuracies during integration, there may be a slight deviation between the start and target positions. The integration is considered 'stable' if this deviation is sufficiently small.

sampling planes

In the first step, one has to make sure that the considered 'search' planes are set up densely enough to intersect all closed orbits. This plane selection problem is shared with other algorithms for 3D closed orbit extraction based on, e.g., FTLE [79] or the Poincaré map [127]. We do not provide a new contribution in this part of the algorithm.

search $\mathbf{0}$ *in* $\mathbf{s}_{\hat{x}}$

We sample $\mathbf{s}_{\hat{x}}(y, z, \tau)$ on a regular hexahedral grid and apply a piecewise linear interpolation. Therefore, we can use the same algorithm to find critical points as in Section 7.4. The problem of multiple intersections of a closed orbit with the plane is also shared with existing approaches. Also, this part does not provide a new contribution.

Closed Orbit data set

We apply our algorithm to a simple *Closed Orbit* test data set

$$\mathbf{v}(\mathbf{x}) = \begin{pmatrix} \frac{1-r}{r}\, x - \frac{1}{2} y \\ \frac{1-r}{r}\, y + \frac{1}{2} x \\ z \end{pmatrix} \quad \text{with} \quad r = \sqrt{x^2 + y^2}\,. \tag{7.9}$$

saddle behavior

This test data has an isolated closed orbit with a saddle behavior – the unit circle in the (x, y)-plane. An extraction approach based on the Poincaré map fails because the Poincaré map on a plane does not converge to its invariants depicting the closed orbit. Also, the FTLE-based approach by Kasten et al. [79] fails in this simple example because for a positive integration time τ, FTLE of Equation (7.9) gives a constant value due to the linear third component. This means that it does not build the ridge structures which the algorithm by Kasten et al. requires.

Our algorithm extracts the closed orbit, as shown in Figure 7.4. This result is a *new contribution*. To the best of our knowledge, no existing algorithm can handle this simple example. Our algorithm covers cases that so far cannot be handled otherwise.

Section 7.6 presents isolated closed orbits with a saddle characteristic in a real data set – a time slice of a blood flow simulation in a cerebral aneurysm (compare Figure 7.9). This is – to the best of our knowledge – the first saddle closed orbit on real data shown in the Visualization community, which comes as a side-product of our general approach.

	Number of Samples			Time in h		
	Grid	Initial	Refine	Initial	Refine	Integr.
Double Gyre	$5 \cdot 10^9$	$2.3 \cdot 10^4$	$4.5 \cdot 10^5$	4	2	90%
Aneurysm	$1.3 \cdot 10^{10}$	$2.9 \cdot 10^4$	$4.6 \cdot 10^5$	124	46	98%
Cavity	$5.6 \cdot 10^8$	$2.6 \cdot 10^5$	—	11	—	60%
Square Cylinder	$1.2 \cdot 10^{11}$	$2.7 \cdot 10^5$	$2.2 \cdot 10^6$	43	25	75%

Table 7.1: SAMPLES & TIMINGS. The table lists the number of samples for the sampling grid and the recirculation samples found before and after refinement. Furthermore, timings for the initial search and the refinement are listed and the percentage used for pathline integration. The times are rounded to full hours.

7.6 RESULTS

test system

We show results for one synthetic data set and three simulated flows. Our algorithm was run in parallel on a virtual machine that could access 18 Intel Xeon E5-2690v3 CPUs at 2.6GHz with 128GB RAM.

color maps

t→Viridis
τ→Magma

We project $\overline{\overline{Y}} \in \mathbb{R}^5$ to $Y \in \mathbb{R}^3$ and show the resulting 3D surface. Color-mapping is used to show the additional coordinates start time t and integration time τ. We use the *Viridis* color map [150] to visualize t and the *Magma* color map to visualize τ – compare Figure 7.1 to get an impression. The particular mapping depends on the extrema of the current data set. Recirculation surfaces are shown with color-mapping for either t or τ. All recirculating pathlines are rendered as tubes, color-coded with integration time τ as constant color. Seeding positions for recirculating pathlines are shown as spheres, color-coded with start time t as constant color. In Figures 7.1, 7.5 and 7.6, we show boundary curves depicted with blue color.

initial grid & timings

For the sampling, we give the resolution of the *initial* sampling grid. We used the initial grid to compute candidates for recirculation points on the surface. The candidates were locally refined in a second extraction step, as described in Section 7.4.6. Furthermore, we provide the timings for the computation of the initial recirculation points, the refinement, and the percentage of the time used for pathline integration. The remaining computation time was used for managing the sampling grid and searching for critical points. Table 7.1 lists the sample numbers and the timings for all data sets.

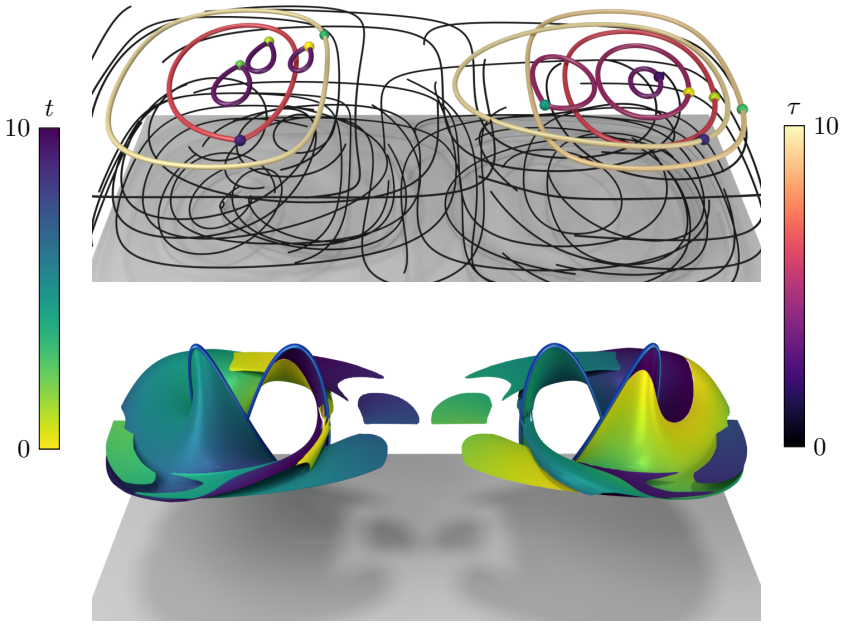


Figure 7.5: RECIRCULATION IN THE 3D DOUBLE GYRE. Top: Random pathlines, including pathlines with recirculation. Recirculating pathlines (constant color τ) are shown with their start points (spheres of color t). Bottom: Recirculation surfaces. The color of the surface indicates start time t. Boundary curves are depicted in blue color.

7.6.1 *Double Gyre*

3D version

The Double Gyre is a synthetic data set, which we introduced in Chapter 3. In the original form, it represents a periodic 2D unsteady velocity field. We add a third component and define

$$\mathbf{v}(\mathbf{x},t) = \begin{pmatrix} -\frac{\pi}{10}\,\sin(\pi f(x,t))\,\cos(\pi y) \\ \frac{\pi}{10}\,\cos(\pi f(x,t))\,\sin(\pi y)\,\frac{\mathrm{d}}{\mathrm{d}x} f(x,t) \\ \frac{1}{5}\,z\,(1-z)\,(z-\frac{1}{4}\sin(\frac{2}{5}\pi t)-\frac{1}{2}) \end{pmatrix} \quad \text{with}$$

$$f(x,t) = a(t)\,x^2 + b(t)\,x\ ,$$

$$a(t) = \tfrac{1}{4}\sin(\tfrac{\pi}{5}t)\ , \quad b(t) = 1 - \tfrac{1}{2}\sin(\tfrac{\pi}{5}t)\ .$$

Particles advected in the domain $[0,2]\times[0,1]\times[0,1]$ never leave this domain. The third component adds a slight sine-like movement in z-direction around the particle's seeding 'height.' The effect is strongest around $z = \frac{1}{2}$ and vanishes at the borders, i.e., $z = 0$ and $z = 1$. Particles with $z_0 \geq \frac{1}{2}$ get dragged slightly to the top of the domain, particles $z_0 < \frac{1}{2}$ to the bottom.

recirculating pathlines

Figure 7.5 (top) gives an overview of the data by visualizing random pathlines in gray color. This includes a few examples of pathlines that show recirculation. Their color indicates the integration time τ, the

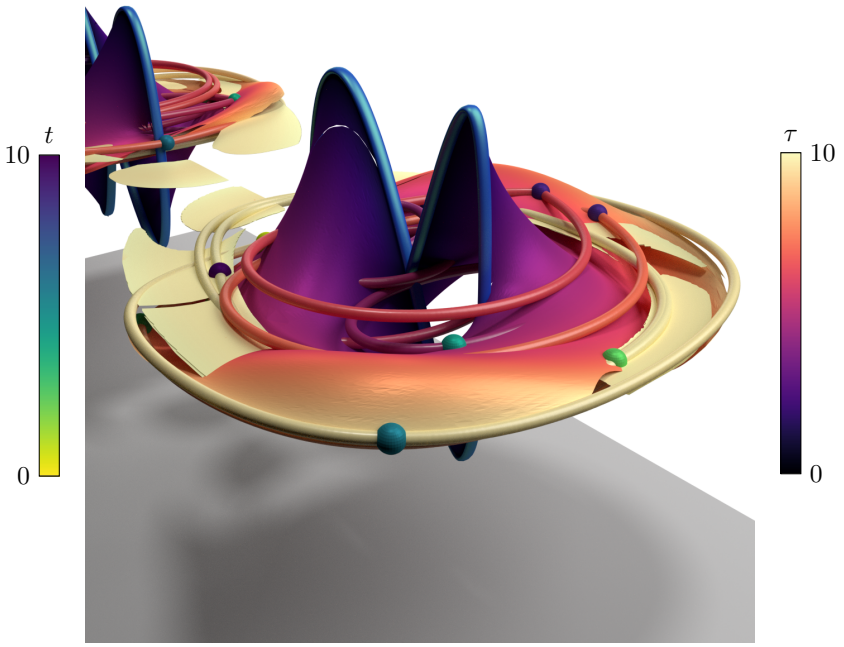


Figure 7.6: RECIRCULATION SURFACE IN THE 3D DOUBLE GYRE. Recirculation surfaces and recirculating lines. The color map of the surface indicates integration time τ. Recirculating pathlines (constant color τ) are shown with their start points (spheres of color t). Boundary curves are depicted in blue color.

starting points are depicted as spheres, and their color encodes the start time t. Note that generally only limited insight into the data can be obtained from rendering pathlines simply because of the sheer amount of existing pathlines.

recirculation surfaces

Figure 7.5 (bottom) shows a reconstruction of recirculation surfaces with t mapped as color. Figure 7.1 shows the same view with t and τ mapped as color and randomly selected pathlines. The close-up in Figure 7.6 maps integration time τ as color and includes few recirculating pathlines (t and τ encoded as for Figure 7.5). We can also see the paths of critical points of $\mathbf{v}$ as boundaries of the recirculation surfaces. The corresponding boundary curves are depicted in blue color.

domain, sampling & timings

The search space was restricted to $(\mathbf{x}, t, \tau)^{\mathrm{T}} \in [0,2] \times [0,1] \times [0,1] \times [0,10] \times [0,10]$. The size of the initial sampling grid was $200 \times 100 \times 100 \times 50 \times 50$. We found 22 667 initial candidates on the recirculation surfaces, which took 254 minutes. Local refinement of the initial candidates generated a total of 445 062 samples and took additional 92 minutes. Nearly 90% of the computation time was spent on pathline integration. Timings and samples are summarized in Table 7.1.

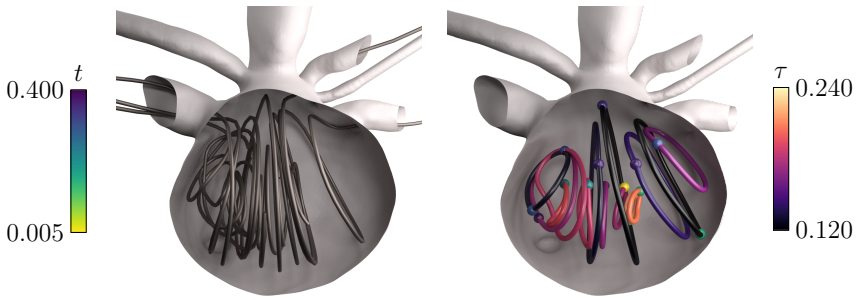


Figure 7.7: PATHLINES IN THE ANEURYSM FLOW. Left: Random pathlines illustrate flow behavior. The flow passes from the left (inflow) to the right (outflow). Right: Some recirculation pathlines – spheres mark the seed positions. Curves are color-coded to show total integration time τ. Spheres are color-coded to show seeding time t.

7.6.2 *Aneurysm Flow*

description

The Aneurysm Flow data set is a 3D unsteady blood flow simulation in a geometric model of a cerebral blood vessel containing a strongly developed aneurysm. The research question behind the simulation is related to rupture prediction. Rupture of an aneurysm is associated with a high mortality rate. Depending on the rupture prediction, decisions about possible surgeries are to be made. Rupture prediction of cerebral aneurysms is still largely unsolved [182].

However, it is known that rupture risk is related to whether the flow in the aneurysm is 'cut-off' and disconnected from the vessel's main flow. The (un-)connectedness of the flows in the aneurysm and the main vessel is related to recirculation. If the main flow mostly passes through the aneurysm, few or no recirculation areas are within the aneurysm.

Our approach finds several recirculation surfaces that cover large areas within the aneurysm. On the other hand, there are also large areas without recirculation surfaces, indicating that the main flow and the aneurysm flow are partly connected. While this does certainly not solve the rupture prediction problem, the systematic study of recirculation in many aneurysm data sets may reveal correlations between the presence, shape, and size of recirculation surfaces and likely rupture events.

recirculating pathlines

Figure 7.7 shows the results as pathlines. Few random pathlines illustrate the flow, which enters the domain from the left side and leaves to the right. A collection of recirculating pathlines indicate the presence of recirculation. The spheres indicate the start positions on the recirculation surfaces with color encoding start time t. The colors of the curves encode the integration time τ.

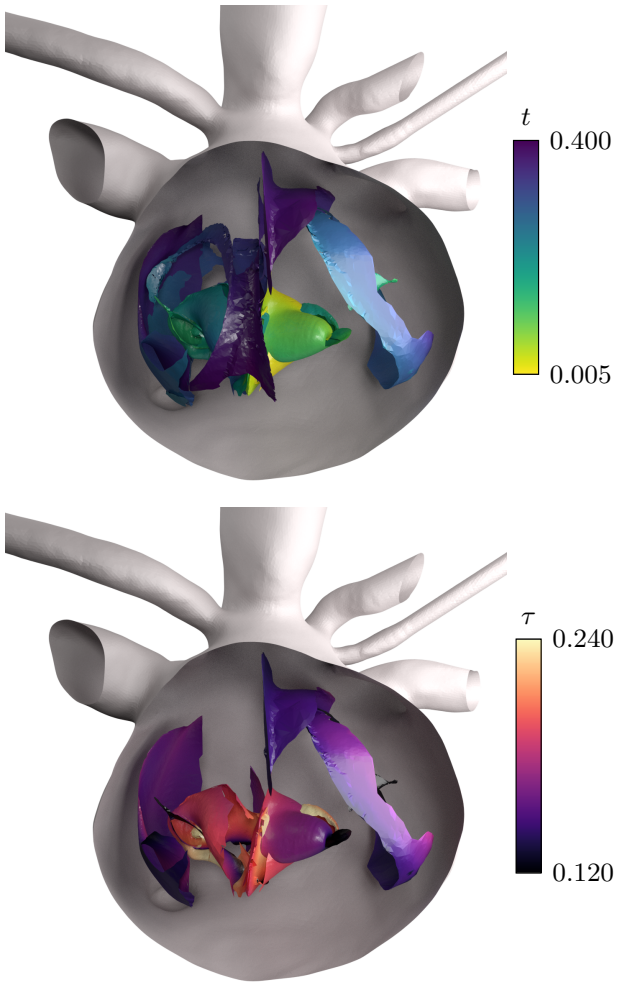


Figure 7.8: RECIRCULATION SURFACES IN THE ANEURYSM FLOW. Top: Recirculation surfaces, color encodes t. Bottom: Recirculation surfaces, color encodes τ. The center surfaces were cut out in the bottom image to clear the view on the surfaces in the center region.

recirculation surfaces

Figure 7.8 shows recirculation surfaces. The color encodes start time t (top) and integration time τ (bottom). In the bottom image, we removed parts of occluding surfaces. Therefore, recirculation surfaces in the domain's center are better visible.

domain, sampling & timings

The search space was restricted to $(\mathbf{x}, t, \tau)^{\mathrm{T}} \in [-0.0155, -0.025] \times [0.2245, 0.2355] \times [-0.1765, -0.1645] \times [0.0, 0.75] \times [0.0, 0.25]$ and the size of the initial sampling grid was $130 \times 110 \times 120 \times 150 \times 50$. We found 29 228 candidates, which took 124 hours. The refinement resulted in 458 362 total samples, and needed additional 46 hours. The integration of pathlines accounts for 98% of the time. Timings and samples are listed in Table 7.1.

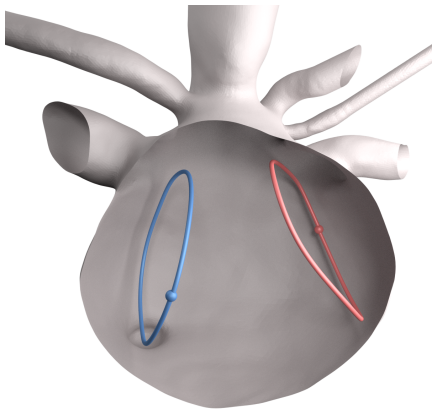


Figure 7.9: ISOLATED CLOSED STREAMLINES IN THE ANEURYSM FLOW. We show the results for a constant time. The single time slice defines a steady vector field, which contains two *isolated closed streamlines*. The spheres denote the start positions.

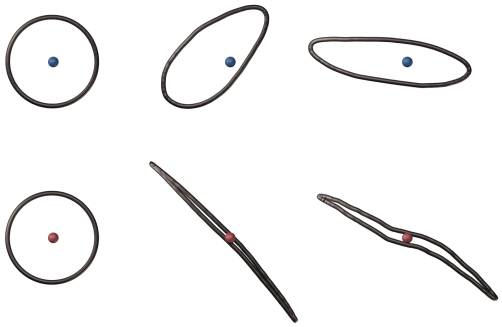


Figure 7.10: POINCARÉ MAPS. The planar curves in each row visualize the Poincaré maps for both isolated closed streamlines at the start positions (colored spheres). The columns show the circular seed structure (left), the seed advected by one turn in the forward (center), and the backward direction (right). Both isolated closed streamlines show saddle behavior.

isolated closed streamlines

In addition, we examined a single time slice of the Aneurysm Flow data and searched for isolated closed streamlines in the steady flow. We found two closed streamlines, both with *saddle behavior*. Figure 7.9 shows the streamlines. Figure 7.10 visualizes the Poincaré maps. For the Poincaré map visualization, we consider planes through the red and blue points orthogonal to the velocity direction. In each plane, a small closed seed curve – a circle – is constructed; see left column in Figure 7.10. This circle's points are integrated both in the forward and backward direction until they intersect the plane again. The resulting intersection curves are the closed curves in the center and right columns of Figure 7.10. The relation of these integrated curves and the seed circle characterize the closed orbit. Since the curves are partially inside and partially outside the seed circle, we have a closed orbit with saddle behavior.

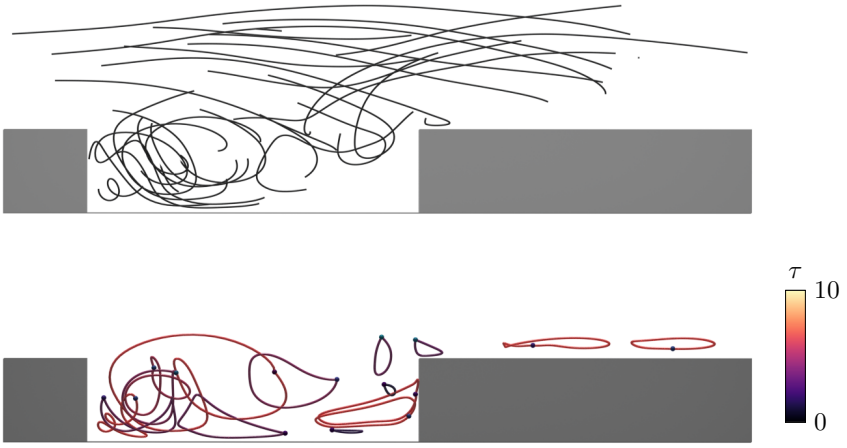


Figure 7.11: PATHLINES IN THE CAVITY FLOW. Top: Random pathlines – seeded particles move from left to right. Bottom: Collection of recirculating pathlines in and behind the cavity. The spheres mark the seed; color encodes integration time τ (curves) and t (spheres).

Figure 7.12: RECIRCULATION SURFACES IN THE CAVITY FLOW. Recirculation surfaces extended to $(x, y, t)^{\mathrm{T}}$-space. The surface color encodes the integration time τ.

7.6.3 *Cavity Flow*

description

The Cavity Flow data set is a vector field describing the flow over a 2D cavity. We introduced it already in Section 5.8.3. One of the primary analysis question for the data set is: do particles in the cavity always move out after a while, and if so, what is the maximal dwell time in the cavity? This question is strongly related to recirculation within the cavity.

recirculating pathlines & recirculation surfaces

As this is a 2D time-dependent flow, recirculation surfaces are defined in 4D $(x, y, t, \tau)^{\mathrm{T}}$-space. We found a relatively dense set of recirculation surfaces with rather long τ values inside the cavity, which indicate a high number of particles that remain in the cavity for a longer period. Further, we found more recirculation surfaces downstream outside the cavity, which indicate the cavity's impact even at a certain distance. Figure 7.11 shows randomly selected pathlines (top) and randomly selected recirculating pathlines (bottom). We use the same color-coding as for the previous figures. Figure 7.12 shows recirculation surfaces in

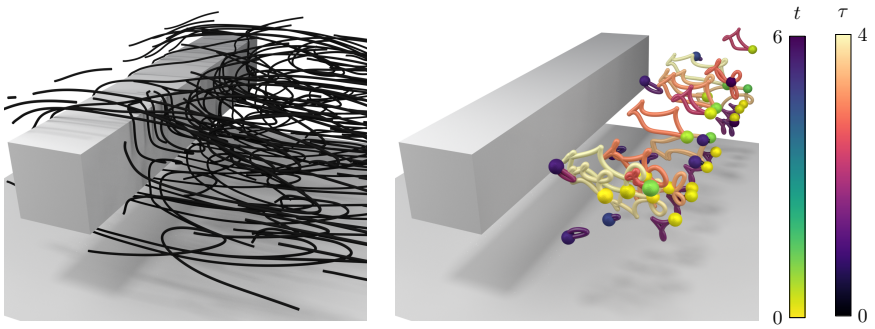


Figure 7.13: PATHLINES IN THE SQUARE CYLINDER FLOW. Left: Random pathlines next to the obstacle. Right: Recirculating pathlines close behind the obstacle, color encodes total integration time τ (curves) and t at the start (spheres).

the Cavity Flow. Because the Cavity Flow is a 2D data set, we can use the third spatial dimension for the visualization. We use the start time t as the third dimension and visualze the recirculation surfaces in $(x, y, t)^{\mathrm{T}}$-space. The integration time τ is mapped to the surface color.

domain, sampling & timings

The search space was restricted to $(\mathbf{x}, t, \tau)^{\mathrm{T}} \in [-1, 8] \times [-1, 1.5] \times [0, 10] \times [0, 10]$, and the size of the initial sampling grid was $900 \times 250 \times 50 \times 50$, i.e., there is no z-direction. We computed 257 080 samples on the recirculation surface, which took 10.7 hours. No refinement was done because the initial result was sufficient for surface reconstruction. About 60% of the computation time was used for pathline integration. Table 7.1 lists samples and timings.

7.6.4 *Square Cylinder Flow*

description

The Square Cylinder Flow data set describes the simulated flow around a 3D square cylinder based on the Navier-Stokes simulation by Camarri et al. [22]. Tino Weinkauf provided the uniformly resampled version of this velocity field (compare [47]).

recirculating pathlines

Figure 7.13 shows a few random pathlines (left) and recirculating pathlines (right) in a region close behind the obstacle. We use the same style as for previous figures. The image shows recirculating loops with short integration time and 'curvy' pathlines with long integration time.

recirculation surfaces

Figure 7.14 shows recirculation surfaces in the same region of the domain. The recirculation surfaces near the obstacle show high curvature, and the reconstruction algorithm could not generate a consistent surface mesh due to undersampling. The recirculation surfaces at higher times show smaller curvature and we thus obtained a more sufficient sampling and a better reconstruction.

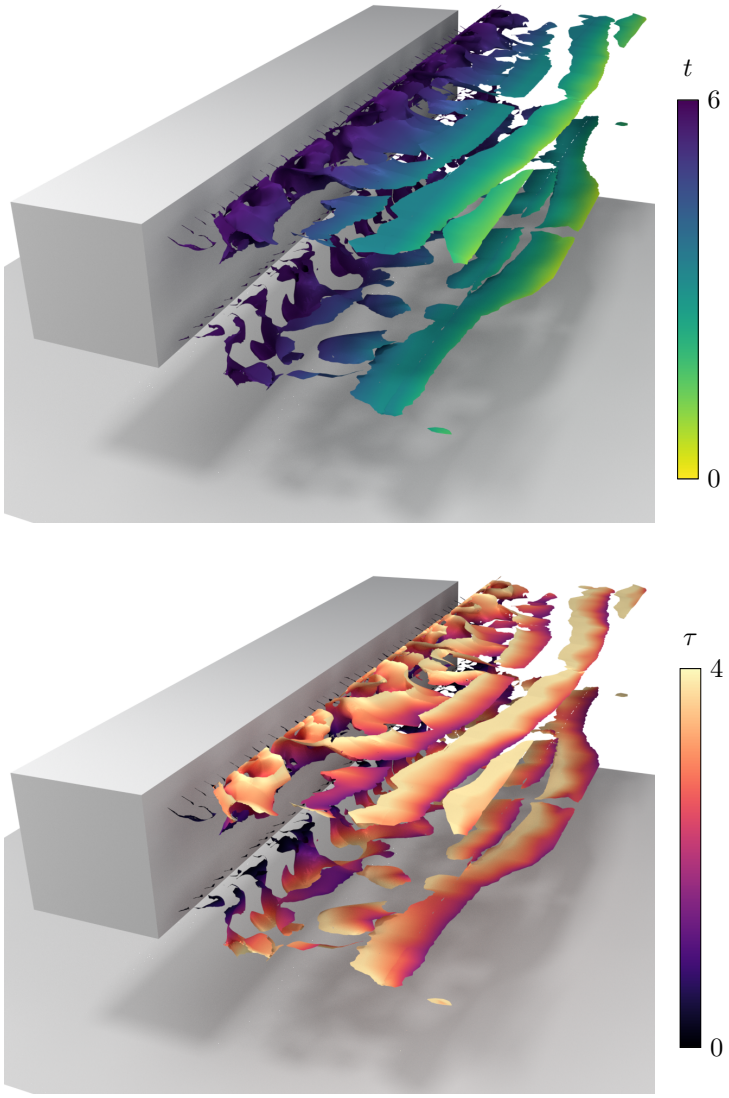


Figure 7.14: RECIRCULATION SURFACES IN THE SQUARE CYLINDER FLOW. Top: Recirculation surfaces behind the obstacle, color encodes t. Bottom: Same with color-coded τ. The surface shows a high spatial curvature directly behind the cylinder. The sampling was not dense enough to capture the recirculation everywhere sufficiently. Therefore the reconstructed surface has jagged borders and some holes.

domain, sampling & timings

The search space was restricted to $(\mathbf{x}, t, \tau)^{\mathrm{T}} \in [0.5, 2.5] \times [-0.65, 0.65] \times [0, 6] \times [0, 6] \times [0, 6]$. The size of the initial sampling grid was $400 \times 270 \times 1200 \times 30 \times 30$, and we obtained 269 187 candidates at the surface. A local refinement generated a total of 2 217 204 samples. The computation of the initial samples took 43 hours, the refinement additional 25 hours. About 75% of the computation time was spent on pathline integration. Timings and sampling information is summarized in Table 7.1.

7.7 DISCUSSION

In the following, we discuss recirculation surfaces, their value for Flow Visualization and their algorithmic reconstruction.

7.7.1 *Relation to Other Flow Visualization Techniques*

$\overline{\overline{Y}}$ uses complete flow map

There is a variety of visualization techniques that locally focus on the velocity field. In recent years, integration based techniques moved into the focus of research, i.e., techniques that apply a visual analysis of the flow map. We claim that our technique is the *first comprehensive feature extraction approach incorporating the complete flow map*. Our technique opts for a *strong abstraction* – we search for a steady 3D surface representing the relevant information of a complete 5D field. Domain experts would like to see a 'still image' – that can be physically printed as a figure in an article – that tells as much as possible of the story of a 3D unsteady flow. While there exist several solutions for steady flows, we are not aware of any similar approaches for unsteady flows. In this sense, this work could be considered as the first one in this direction.

useful for still images

comparison to FTLE

Because of the high computational complexity, existing methods for flow map analysis focus on subsets of ϕ. For instance, the well-known FTLE visualization techniques usually pick a particular t and τ to either volume render or do feature extraction on the resulting 3D scalar field. Further, different t or different τ can be shown via animations. We are not aware of any approach to systematically computing and visualizing FTLE fields for all t and all τ values. We are neither aware of any other technique that systematically explores the whole 5D flow map.

7.7.2 *Degeneration of Recirculation Surfaces*

degeneration is unstable

Recirculation surfaces can degenerate to points, lines and volumes, but these cases are structurally unstable. Adding noise to the data will break the points, lines, or volumes into a finite number of surfaces. An exception is the case of steady flows where – as shown in Section 7.5 – recirculation surfaces degenerate to structurally stable isolated closed streamlines.

To illustrate the idea of structural stability, consider algorithms for isosurface extraction in 3D scalar fields. Such isosurfaces can collapse to a single point (for instance, the scalar field $s(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{x} = 0$ in its origin) or a volume (for instance $s(\mathbf{x}) \equiv 0$). These cases are usually excluded from an isosurface extraction algorithm because of their structural instability. We do the same for recirculation surfaces.

7.7.3 *Recirculation as a Phenomenon*

We are not sure if recirculation surfaces are the best approach to treat recirculation as a phenomenon. Neither exists a standard definition of recirculation nor are there ground truth data or other approaches to compare with. Thus, a final answer can only be given in the future by a test of time.

However, some points give evidence on the usefulness of our approach:

positive feedback

- User feedback: We presented the approach to domain experts that are involved in numerical flow simulation. They acknowledged the usefulness and found the detected structures interesting and informative. However, we consider this positive feedback as anecdotic. A formal user study was neither done nor are we aware of a proper approach to the verification – or falsification – of recirculation surfaces by a user study.

surprising results

- A mixture of expected and surprising results: The found recirculation surfaces mainly correspond with the expectation. For instance, for the Square Cylinder Flow, we expected recirculation behind the cylinder, and our approach confirms this. On the other hand, there were surprising results. For the Cavity Flow, we found recirculation surfaces both inside the cavity and outside downstream the flow.

closed orbits

- Closed orbits: We interpret the fact that a particular case of our algorithm solves the problem of extracting isolated closed orbits with saddle behavior as a point of evidence for the usefulness of recirculation surfaces.

7.7.4 *Computation Time*

high computational costs

The computation time is long. The reason is in the nature and inherent properties of flow maps. We have to integrate vast amounts of pathlines to locate samples on the recirculation surface. This is the most expensive part of the algorithm. Our experiments show that up to 98% of processing time is used for pathline integration. We are not aware of any faster algorithm doing a systematic analysis of the whole flow map.

However, our approach can be seen as a *preprocessing step*. Recirculation surfaces are computed once and stored along with the flow data. Then recirculation surfaces can be explored by interactively seeding path lines on them and observing their behavior.

7.7.5 *Parameters*

The presented algorithm comes with some parameters that can be divided into two classes:

1. Parameters of the existing standard algorithms that our algorithm is based upon.

2. New parameters that are inherent to the new algorithm.

inherited parameters

The first group includes the choice of ODE solver and its parameters required for numerical pathline integration. For instance, we use a fourth-order Runge-Kutta solver with an adaptive step size controlled, e.g., by error tolerance and maximum step size. Furthermore, there is the initial grid resolution for finding isolated critical points and the maximum recursion depth to limit the subdivision and a strategy for clustering multiple detected points. Finally, a heuristic is required to track critical points, that decides when to connect extracted critical points at different time steps. We do not further discuss these parameters here because they are well-discussed in the original publications. Instead, we consider the algorithms as black-boxes for our approach.

inherent parameters

The parameters of the second group reduce to the grid resolution for sampling $\mathbf{s}$. Its variation does not show any surprising behavior. A higher sampling rate increases accuracy and computation time. In theory, the sampling rate should not fall below the Nyquist frequency to capture the recirculation surfaces. It can be bounded by using assumptions on the local feature size or curvature of the surface. However, we currently see no way to reliably determine the (local) sampling rate.

sampling & reconstruction

The situation is significantly more difficult as for surface reconstruction in 3D – which is a difficult problem already. Sampling and reconstruction must account for the curvature of a 2-manifold embedded in a 5D space. We know that this manifold has regions of high curvature. Moreover, the differential geometry in 5D does not carry over to the projection to 3D, the ultimate visualization space. For instance, the projection $Y \in \mathbb{R}^3$ is, in general, a 2-manifold.

However, configurations in the projection can lead to different topologies of the projected surface in 3D;

- The points $(\mathbf{x}, t, \tau)^{\mathrm{T}}$ and $(\mathbf{x}, t+\tau, -\tau)^{\mathrm{T}} \in \overline{\overline{Y}}$ in 5D are mapped to the same point on the surface Y in 3D, i.e., Y is double-folded (see Section 7.3.4).

- The points $(\mathbf{x}, t, 0)^{\mathrm{T}} \in \overline{\overline{Y}}$ build boundary curves of Y (see Section 7.3.4).

- The closed streamlines in steady flows $\mathbf{v}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x})$ constitute a degenerate recirculation surface (see Section 7.5).

In this work, we chose the most straightforward approach and used regular sampling. The disadvantages of this approach are potential undersampling, i.e., loss of information, or oversampling, i.e., waste of computation time. Both are likely to occur at least locally – the grid provides no means of adaptation. However, this property is shared with

many very successful algorithms that restrict themselves to operating on uniform grids. The simplicity of this approach is, at the same time, its most significant advantage. Our setting allows to rely on relatively simple and proven standard methods, and it comes with a moderately complex implementation.

7.8 LIMITATIONS & FUTURE RESEARCH

conclusion

Recirculation is an important phenomenon that is studied in many application domains. Recirculation can be explained and understood intuitively. Surprisingly there existed no formal definition so far. In Section 7.3, we gave a formal definition and properties of *recirculation surfaces* and showed how to compute them in Section 7.4. We search the surface in 5D by intersections with lines. This way, we can project the problem to a search for critical points in linear steady 3D vector fields. The search for isolated closed streamlines is a particular case of our approach, which was explained in Section 7.5. In Section 7.6 we showed results followed by a discussion in Section 7.7.

limitation & future reserch

Our approach provides points of the recirculation surface on the edges of a regular grid. On the one hand, this ensures a certain minimum density of sample points (within the range covered by the grid). On the other hand, it hinders a higher density of the samples, leading to artifacts in the point-based rendering. We see two ways to deal with this:

1. A modification of the algorithm to adaptive grids.
2. The reconstruction of a triangle mesh from sample points.

For the latter option, surface extraction could be carried out by an algorithm similar to Marching Cubes [98]. However, in addition to the standard Marching Cubes configurations, few more cases have to be considered. The 3D surface Y can have self-intersection and therefore it can have more than one intersection point with an edge of a cell. Moreover, in the presence of boundary curves, Y may enter a cell but does not necessarily leave it and end in the boundary curve instead.

Another current limitation is performance. The apparent solution for future research is adaptive sampling in 5D, which has the potential to speed up the algorithm. Then again, no extreme improvement of the performance can be expected since our algorithm does an exhaustive feature extraction in an expensive 5D space.

Part IV

CONCLUSION & FUTURE RESEARCH

8

CONCLUSION

This thesis contributed different approaches to Flow Visualization that are entirely related to the flow map.

part I background

In Part I, we set the theoretical background. Chapter 2 gave a short introduction to the field of Flow Visualization. We introduced velocity fields and flow maps – the two common ways to represent flow data in Flow Visualization. Visualization techniques were listed, and we explained the inherent properties of flow maps. In Chapter 3, we discussed the Double Gyre as an important benchmark data set used throughout the thesis.

part II flow map processing

Part II was dedicated to the field of flow map processing. Many techniques for the synthesis, design, and processing of vector fields exist. There is a lack of similar methods for the flow map. We see the reason for this in the complexity of the flow map. In Chapters 4 and 5, we presented research results as the first step towards powerful and comprehensive flow map processing techniques.

flow map deformation

Chapter 4 introduced an approach to modify flow maps directly. Based on a space deformation, we change flow map entries in a local area defined in space-time. The deformation ensures a continuous transition between original and modified regions. Local changes entail a global adaption of connected flow map parts. To quickly identify related areas, we use a lookup table that contains global mapping information. The affected regions are then explicitly adapted to the modification. We demonstrated the applicability of the concept with an interactive tool to transform pathlines. We can modify the flow map at arbitrary locations with the presented approach and ensure the defining flow map properties are kept. We see this as a prerequisite for further flow map processing techniques.

drift fields

Chapter 5 introduced *drift fields* – a new approach to represent flows from the Lagrangian perspective. Drift fields are situated between velocity fields and flow maps. They are low dimensional like velocity fields but directly encode particle trajectories like flow maps. We gave a formal definition of drift fields, discussed their properties and their relation to velocity fields and flow maps. An optimization-based approach to compute them was introduced. We presented a method for pathline extraction from drift fields that relies on local operations. The computational effort to gain pathlines increases with the sampling resolution but is independent of the integration time τ. Pathline extraction can be used to convert drift fields to flow maps. Drift fields are closed under

perturbation, i.e., a small change of a drift field is still a drift field. We adapted the space-time deformation technique to get an intuitive tool for modifying them. Hence, we can utilize drift fields as a convenient way to deform flow maps. We applied drift fields to different data sets and discussed their advantages and disadvantages.

part III flow features

In Part III, we studied flow features that are extracted from the flow map. Many flow visualization and feature extraction techniques rely on massless particle integration. Hence, they use the flow map, but in most cases, only a small part of it. In Chapters 6 and 7, we presented two flow features that use a more significant part of the flow map.

FTLE ridge lines

In Chapter 6, we developed a method to determine LCS based on ridge extraction from FTLE fields. Computing LCS from FTLE fields is a well-studied technique that was used several times in the past decades in Flow Visualization. The standard approach integrates particles for a finite time interval with *fixed* starting and end times. Hence, only a limited part of the flow map is used. We presented a method to compute FTLE fields also using intermediate times. The additional information is used to steer the refinement of an adaptive sampling grid. Thus, we can resolve thin and dense ridge structures for long integration times. Furthermore, we extracted statistical information about the ridge geometry for the observed time interval. We applied our method to different synthetic and simulated velocity fields and discovered different behavior in the ridge statistics for each data set.

recirculation surfaces

Chapter 7 introduced *recirculation surfaces*. To the best of our knowledge it is the first flow feature that incorporates the *full 5D flow map* of a 3D unsteady flow. A particle has recirculating behavior in a flow if it reaches its starting position after a finite integration time. We showed that the set of particles with this behavior builds a 2-manifold in 5D space – the recirculation surface. We presented a flow map based distance function that implicitly defines recirculation surfaces and studied its properties. Furthermore, we explained an algorithm that extracts samples on recirculation surfaces by intersections with lines. Thus the problem is transformed into a search for critical points in steady 3D vector fields. With a small adaption, our algorithm can find isolated closed streamlines with saddle behavior in steady 3D vector fields. We extracted recirculation surfaces from four different data sets and presented the results.

9

FUTURE RESEARCH

We already pointed out some possible ways for future research in each chapter.

extension to 3D

The theoretical foundations of the presented approaches should hold for arbitrary dimensions. Nevertheless, only recirculation surfaces were realized for 3D unsteady flows. Hence, the most obvious extension is the implementation of the remaining three methods for 3D unsteady flows. This sounds easy at the first moment but is indeed not trivial. The problems that have to be solved for higher dimensions are either of technical or conceptual nature. On the technical side, we need new data structures that also work for 3D. Furthermore, the additional dimension leads to linear growth of the computational effort and memory consumption. A portation to GPU can compensate for the increase of the computational effort. Especially when it comes to particle integration, massively parallel computation is possible. On the conceptual side, we have to think about solutions for one additional dimension. For instance, where we have ridge lines in FTLE fields in 2D, we get ridge surfaces in 3D. The presented deformation of drift fields is intuitive. Flow map modification by space-time deformation is already hard to predict for 2D flows. Both approaches rely on a deformation area in space-time, i.e., a deformation sphere. For 3D flows, we would need a 4D sphere or complete new tools for an interactive modification. Furthermore, for a higher dimension, the visualization gets more complex. 2D FTLE fields and 2D time slices of drift fields would become 3D volumes that are hard to visualize. In summary, we have to deal with the same problems as many other approaches when we want to realize 3D unsteady flows. The questions get more exciting when we think about an approach-oriented improvement.

technical problems

conceptual problems

flow map deformation

With the presented approach, local modifications of flow maps are possible. For instance, we could think about smoothing a local area by 'averaging' flow map entries. Is it possible to extend such an operation to a global scale? In theory, we could extend the method to the full flow map by a repeated local smoothing operation at all positions. Each local smoothing entails a global adaption to keep the inherent flow map properties. The result would be a valid flow map, but it is not clear if it still represents a 'useful' flow. A possible solution could be to rate the quality of a flow map. This implies measures that evaluate whether a flow map is 'good' or 'bad.' If these measures were present, we could decide if a deformation improves or worsens the flow map.

drift fields

A desirable improvement of drift fields lies in their initialization. For our approach, we use linearly increasing scalar fields and determine the best initialization time. We are convinced that there must be better solutions, but we could not find a robust algorithm for their determination yet. We see two possible ways to go – either a new way to rate a drift field's quality or a different approach for their initialization.

FTLE ridge lines

Our approach for FTLE ridge computation would benefit from a better extraction of ridge geometries. Furthermore, it would be interesting to evaluate if the proposed ridge statistics and their development give useful insights into flow behavior. Therefore a large collection of statistics for different flows would be necessary to compare them.

recirculation surfaces

A similar question arises for recirculation surfaces: Where are they useful? We presented our results to experts in different domains. Most of them were surprised that such structures even exist. Recirculation, as a phenomenon, was discussed in many different areas. Therefore, we are convinced that recirculation surfaces deliver useful information for some application areas.

We see further improvements in the computation of the surfaces. The current technique related to an initial sampling grid is straightforward but computationally expensive. A possibly better solution could be an advancing front algorithm that expands the surface in 5D. Experiments have shown that these kinds of solutions suffer from extreme numerical instability. Furthermore, we do not know any method that evaluates whether the initial sampling grid is good or bad, i.e., is it too sparse or too dense. Another sampling strategy could be a solution, e.g., some kind of Monte Carlo method, that randomly samples the domain and possibly delivers a ground truth surface.

Recirculation surfaces would also benefit from a better surface reconstruction either in 3D or even better in 5D. We could think about a version of the ball-pivoting algorithm particularly customized for recirculation surfaces in space-time.

flow map representation

The last research direction we want to point out is related to the flow map representation. We sampled the flow map on a regular grid in $(\mathbf{x}, t, \tau)^{\mathrm{T}}$ space and used multi linear interpolation for reconstruction in all our algorithms. It would be desirable to have a data structure that is optimized for each particular flow map. A straightforward approach would be an adaptive or an unstructured grid whose cells are defined in 5D space. Another solution could be a completely different representation. For instance, one could think about a representation in frequency space, similar to the Fourier transformation for images. This would open entirely new ways to work with the flow map.

Part V

APPENDIX

A

MODIFICATION BY SPACE-TIME DEFORMATION KEEPS FLOW MAP PROPERTIES

H. Theisel has developed the following proof in [178].

In Section 2.4, we introduce the defining properties of a flow map ϕ. These properties are the identity, the additivity, and the inversion. We define them as:

$$\text{Identity:} \qquad \phi(\mathbf{x}, t, 0) = \mathbf{x}\,, \tag{A.1}$$

$$\text{Additivity:} \qquad \phi(\phi(\mathbf{x}, t, \tau_1), t + \tau_1, \tau_2) = \phi(\mathbf{x}, t, \tau_1 + \tau_2)\,, \tag{A.2}$$

$$\text{Inversion:} \qquad \phi(\phi(\mathbf{x}, t, \tau_1), t + \tau_1, -\tau_1) = \mathbf{x}\,, \tag{A.3}$$

In 4, we present an approach for the modification of flow maps by space deformation. We define a space deformation $\mathbf{y} : \mathbb{R}^{n+1} \to \mathbb{R}^n$ that maps a point $(\mathbf{x}, t)$ in space-time to the new point $\mathbf{y}(\mathbf{x}, t)$. We demand $\mathbf{y}$ to be local, continuous, and invertible. For this proof, only the invertibility is relevant - it means, $\mathbf{y}^{-1}$ is well defined. The modified flow map $\widetilde{\phi}$ can be computed from ϕ and $\mathbf{y}$ by:

$$\widetilde{\phi}(\mathbf{x}, t, \tau) = \mathbf{y}(\phi(\mathbf{y}^{-1}(\mathbf{x}, t), t, \tau), t + \tau)$$

We want to show that $\widetilde{\phi}$ is also a flow map. Hence the deformation must keep all flow map properties. The inversion is a particular case of the additivity. Therefore it is sufficient to show that $\widetilde{\phi}$ fulfills the identity and the additivity.

Lemma 1. *If ϕ is a flow map, then $\widetilde{\phi}$ is a flow map as well.*

Proof.
We show the identity of $\widetilde{\phi}$:

$$\begin{aligned}
\widetilde{\phi}(\mathbf{x}, t, 0) &= \mathbf{y}(\phi(\mathbf{y}^{-1}(\mathbf{x}, t), t, 0), t) \\
&= \mathbf{y}(\mathbf{y}^{-1}(\mathbf{x}, t), t) \\
&= \mathbf{x}\,.
\end{aligned}$$

We show the additivity of $\widetilde{\phi}$:

$$\begin{aligned}
&\widetilde{\phi}(\widetilde{\phi}(\mathbf{x}, t, \tau_1), t + \tau_1, \tau_2) \\
&= \mathbf{y}(\phi(\mathbf{y}^{-1}(\mathbf{y}(\phi(\mathbf{y}^{-1}(\mathbf{x}, t), t, \tau_1), t{+}\tau_1), t{+}\tau_1), t{+}\tau_1, \tau_2), t{+}\tau_1{+}\tau_2) \\
&= \mathbf{y}(\phi(\mathbf{y}^{-1}(\mathbf{x}, t), t, \tau_1 + \tau_2), t + \tau_1 + \tau_2) \\
&= \widetilde{\phi}(\mathbf{x}, t, \tau_1 + \tau_2)\,.
\end{aligned}$$

□

B

ELEMENTS OF MATRIX H FOR DRIFT FIELDS

Chapter 5 introduces a numeric approach to compute drift fields. The pathline extraction from drift fields works best when the spatial gradients of the scalar field components $\nabla a(\mathbf{x}, t)$ and $\nabla b(\mathbf{x}, t)$ are orthogonal to each other and have unit length. ∇a and ∇b combined build the spatial drift field gradient $\nabla \mathbf{d} = (\nabla a, \nabla b)$. In the ideal case, $\nabla \mathbf{d}$ describes a rotation matrix, i.e., the following equations hold:

$$\nabla a^{\mathrm{T}} \nabla a = 1\,, \quad \nabla b^{\mathrm{T}} \nabla b = 1\,, \quad \nabla a^{\mathrm{T}} \nabla b = 0\,.$$

The greater the difference to these values, the more $\nabla \mathbf{d}$ differs from a rotation matrix. The behavior of $\nabla \mathbf{d}$ under advection is described by:

$$\widehat{\nabla \mathbf{d}} \;:=\; \nabla \mathbf{d} \nabla \phi^{-1} \;=\; \nabla \mathbf{d}(\phi, t_0 + \tau)\,.$$

For the drift field computation, we utilize the following matrix:

$$\mathbf{H}(\mathbf{x}, \tau) := \widehat{\nabla \mathbf{d}} \widehat{\nabla \mathbf{d}}^{\mathrm{T}} = \begin{pmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{pmatrix}\,.$$

Remarks regarding the notation: For better readability, we denote the inverted flow map gradient $\nabla \phi^{-1}$ as $(\bar{\phi}_1, \bar{\phi}_2)$. Furthermore, we use a second index for the inverted flow map gradient, that indicates the x-, or y-component, e.g., $\bar{\phi}_{1y}$ is the y-component of the first column vector of $\nabla \phi^{-1}$. $\widehat{\nabla \mathbf{d}}$ denotes the drift field gradient under advection.

Proof.
We show that entries in $\mathbf{H}(\mathbf{x},\tau)$ correspond to:

$$h_{11} = \nabla a^{\mathrm{T}} \nabla a\,, \quad h_{22} = \nabla b^{\mathrm{T}} \nabla b\,, \quad h_{12} = \nabla a^{\mathrm{T}} \nabla b\,.$$

$$\begin{aligned}
\nabla \mathbf{d} = (\nabla a, \nabla b)^{\mathrm{T}} &= \begin{pmatrix} a_x & a_y \\ b_x & b_y \end{pmatrix}, \\
\nabla \phi^{-1} = \bar{\phi} &= \begin{pmatrix} \bar{\phi}_{1x} & \bar{\phi}_{2x} \\ \bar{\phi}_{1y} & \bar{\phi}_{2y} \end{pmatrix}, \\
\widehat{\nabla \mathbf{d}} = \nabla \mathbf{d} \nabla \phi^{-1} = \nabla \mathbf{d} \bar{\phi} &= \begin{pmatrix} a_x & a_y \\ b_x & b_y \end{pmatrix} \begin{pmatrix} \bar{\phi}_{1x} & \bar{\phi}_{2x} \\ \bar{\phi}_{1y} & \bar{\phi}_{2y} \end{pmatrix} \\
&= \begin{pmatrix} a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y} & a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y} \\ b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y} & b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y} \end{pmatrix} \\
&= \left(\widehat{\nabla a}, \widehat{\nabla b}\right)^{\mathrm{T}},
\end{aligned}$$

$$\begin{aligned}
&\mathbf{H}(\mathbf{x},\tau) = \widehat{\nabla \mathbf{d}} \widehat{\nabla \mathbf{d}}^{\mathrm{T}} \\
&= \begin{pmatrix} a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y} & a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y} \\ b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y} & b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y} \end{pmatrix} \begin{pmatrix} a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y} & b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y} \\ a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y} & b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y} \end{pmatrix} \\
&= \begin{pmatrix} \begin{matrix} (a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y})^2 \\ + \\ (a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y})^2 \end{matrix} & \begin{matrix} (a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y})(b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y}) \\ + \\ (a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y})(b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y}) \end{matrix} \\ \begin{matrix} (a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y})(b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y}) \\ + \\ (a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y})(b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y}) \end{matrix} & \begin{matrix} (b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y})^2 \\ + \\ (b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y})^2 \end{matrix} \end{pmatrix} \\
&= \begin{pmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{pmatrix},
\end{aligned}$$

$$\widehat{\nabla a}^{\mathrm{T}}\widehat{\nabla a} = \begin{pmatrix} a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y} \\ a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y} \\ a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y} \end{pmatrix}$$
$$= (a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y})^2 + (a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y})^2 = h_{11}\,,$$

$$\widehat{\nabla b}^{\mathrm{T}}\widehat{\nabla b} = \begin{pmatrix} b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y} \\ b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y} \\ b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y} \end{pmatrix}$$
$$= (b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y})^2 + (b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y})^2 = h_{22}\,,$$

$$\widehat{\nabla a}^{\mathrm{T}}\widehat{\nabla b} = \begin{pmatrix} a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y} \\ a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y} \\ b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y} \end{pmatrix}$$
$$= (a_x\bar{\phi}_{1x} + a_y\bar{\phi}_{1y})(b_x\bar{\phi}_{1x} + b_y\bar{\phi}_{1y}) +$$
$$(a_x\bar{\phi}_{2x} + a_y\bar{\phi}_{2y})(b_x\bar{\phi}_{2x} + b_y\bar{\phi}_{2y}) = h_{12}\,.$$

□

C

RELATION BETWEEN VECTOR FIELD & FLOW MAP DERIVATIVES

H. Theisel and C. Rössl have developed the following proof in [177].

In Chapter 7 we introduce recirculation surfaces. We use the vector function

$$\mathbf{s}(\mathbf{x}, t, \tau) = \frac{\phi - \mathbf{x}}{\tau}. \tag{C.1}$$

Each location $(\mathbf{x}, t, \tau)^{\mathrm{T}}$ with $\mathbf{s}(\mathbf{x}, t, \tau) = \mathbf{0}$ is a point on the recirculation surface. To study the properties of $\mathbf{s}$ we use partial derivative

$$\frac{\partial \mathbf{s}}{\partial \tau} = \frac{\mathbf{v}_2 - \mathbf{s}}{\tau}.$$

The abbreviation

$$\mathbf{v}_2 := \mathbf{v}(\phi, t + \tau) = \nabla\phi \cdot \mathbf{v} + \phi_t$$

uses a particular property of the flow map, that is not obvious.

Proof.
We show

$$\nabla\phi \cdot \mathbf{v} + \phi_t = \mathbf{v}(\phi, t + \tau)$$

for $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$ and $\phi = \phi(\mathbf{x}, t, \tau)$.

We consider the 3D time-dependent velocity field $\mathbf{v}$ as 4D steady velocity field

$$\widetilde{\mathbf{p}} = \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}.$$

This gives the 4D flow map $\widetilde{\phi}$ for $\widetilde{\mathbf{p}}$ as

$$\widetilde{\phi} = \begin{pmatrix} \phi \\ t + \tau \end{pmatrix},$$

and its (space-time) gradient $\nabla\widetilde{\phi}$ as

$$\nabla\widetilde{\phi} = \begin{pmatrix} \nabla\phi & \phi_t \\ \mathbf{0} & 1 \end{pmatrix}.$$

Consider a point $(\mathbf{x}_0, t_0)$ under integration of $\widetilde{\mathbf{p}}$. Further, consider a point $(\mathbf{x}_1, t_1)$ in a small linear neighborhood of $(\mathbf{x}_0, t_0)$:

$$(\mathbf{x}_1, t_1) = (\mathbf{x}_0, t_0) + \epsilon\widetilde{\mathbf{r}}$$

for a small ϵ. Then the definition of the flow map gradient gives

$$\frac{1}{\epsilon}\left(\widetilde{\phi}(\mathbf{x}_1, t_1, \tau) - \widetilde{\phi}(\mathbf{x}_0, t_0, \tau)\right) = \nabla\widetilde{\phi}(\mathbf{x}_0, t_0, \tau)\,\widetilde{\mathbf{r}}\,.$$

If we place $(\mathbf{x}_1, t_1)$ on the pathline through $(\mathbf{x}_0, t_0)$, i.e., $\widetilde{\mathbf{r}} = \widetilde{\mathbf{p}}(\mathbf{x}_0, t_0)$, it remains on the same pathline during integration. This gives:

$$\frac{1}{\epsilon}\left(\widetilde{\phi}(\mathbf{x}_1, t_1, \tau) - \widetilde{\phi}(\mathbf{x}_0, t_0, \tau\right) = \nabla\widetilde{\phi}(\mathbf{x}_0, t_0, \tau)\,\widetilde{\mathbf{p}}(\mathbf{x}_0, t_0) = \widetilde{\mathbf{p}}(\widetilde{\phi}(\mathbf{x}_0, t_0, \tau))\,.$$

We expand the last two elements for the general case and receive

$$\begin{array}{ccc} \nabla\widetilde{\phi}(\mathbf{x}, t, \tau)\,\widetilde{\mathbf{p}}(\mathbf{x}, t) & = & \widetilde{\mathbf{p}}(\widetilde{\phi}(\mathbf{x}, t, \tau)) \\ = & & = \\ \begin{pmatrix} \nabla\phi & \phi_t \\ \mathbf{0} & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix} & & \widetilde{\mathbf{p}} \begin{pmatrix} \phi \\ t+\tau \end{pmatrix} \\ = & & = \\ \begin{pmatrix} \nabla\phi \cdot \mathbf{v} + \phi_t \\ 1 \end{pmatrix} & = & \begin{pmatrix} \mathbf{v}(\phi, t+\tau) \\ 1 \end{pmatrix} \end{array}$$

Rewriting this condition in the spatial coordinates yields the postulated equation. □

BIBLIOGRAPHY

[1] M. D. Ament, "Computational Visualization of Scalar Fields," PhD thesis, University of Stuttgart, 2014 (cit. on p. 35).

[2] S. Anand, M. Hooshyar, J. Nordbotten, and A. Porporato, "A Minimalist Model for Co-Evolving Supply and Drainage Networks," 2020, Adaptation and Self-Organizing Systems, e-print on arXiv.org (cit. on p. 16).

[3] D. Asimov, "Notes on the Topology of Vector Fields and Flows," NASA Ames Research Center, Tech. Rep., 1993 (cit. on p. 113).

[4] E. Aurell, G. Boffetta, A. Crisanti, G. Paladin, and A. Vulpiani, "Predictability in the large: an extension of the concept of Lyapunov exponent," *Journal of Physics A: Mathematical and General*, vol. 30, no. 1, pp. 1–26, 1997 (cit. on p. 49).

[5] *Auto-generated line-plot of the DAX Performance Index*, https://www.google.com/search?q=dax+performance+index, visited on 23-10-2020 (cit. on p. 14).

[6] S. S. Barakat and X. Tricoche, "Adaptive Refinement of the Flow Map Using Sparse Samples," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2753–2762, 2013 (cit. on pp. 42, 90).

[7] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The Ball-Pivoting Algorithm for Surface Reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999 (cit. on p. 123).

[8] J. Blazek, *Computational Fluid Dynamics*, 3rd. Butterworth-Heinemann, 2015, p. 466 (cit. on pp. 34, 35).

[9] H. Blum and R. N. Nagel, "Shape Description Using Weighted Symmetric Axis Features," *Pattern Recognition*, vol. 10, no. 3, pp. 167–180, 1978, The Proceedings of the IEEE Computer Society Conference (cit. on p. 89).

[10] G.-P. Bonneau, T. Ertl, and G. M. Nielson, *Scientific Visualization: The Visual Extraction of Knowledge from Data*, 1st. Springer, 2006, p. 434 (cit. on p. 11).

[11] R. Borgo, J. Kehrer, D. H. S. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen, "Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications," in *Eurographics 2013 - State of the Art Reports*, The Eurographics Association, 2013 (cit. on p. 19).

[12] M. Botsch and L. Kobbelt, "An Intuitive Framework for Real-Time Freeform Modeling," *Association for Computing Machinery Transactions on Graphics*, vol. 23, no. 3, pp. 630–634, 2004 (cit. on p. 53).

[13] W. E. Boyce and R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, 10th. Wiley Abridged, 2012, p. 832 (cit. on p. 23).

[14] A. Brambilla, R. Carnecky, R. Peikert, I. Viola, and H. Hauser, "Illustrative Flow Visualization: State of the Art, Trends and Challenges," in *Eurographics 2012 - State of the Art Reports*, Eurographics Association, 2012, pp. 75–94 (cit. on p. 18).

[15] C. A. Brewer, "Color Use Guidelines for Mapping and Visualization," in *Visualization in Modern Cartography*, ser. Modern Cartography Series, vol. 2, Academic Press, 1994, pp. 123–147 (cit. on p. 14).

[16] C. A. Brewer, "Guidelines for Use of the Perceptual Dimensions of Color for Mapping and Visualization," in *Color Hard Copy and Graphic Arts III*, vol. 2171, SPIE, 1994, pp. 54–63 (cit. on p. 14).

[17] K. W. Brodlie, J. R. Gallop, C. D. Osland, L. A. Carpenter, R. J. Hubbold, P. Quarendon, R. A. Earnshaw, and A. M. Mumford, Eds., *Scientific Visualization*. Springer, 1992, p. 284 (cit. on pp. 11, 15).

[18] S. L. Brunton and B. R. Noack, "Closed-Loop Turbulence Control: Progress and Challenges," *Applied Mechanics Reviews*, vol. 67, no. 5, 2015 (cit. on p. 113).

[19] R. Bujack and A. Middel, "State of the Art in Flow Visualization in the Environmental Sciences," *Environmental Earth Sciences*, vol. 79, no. 2, pp. 1–10, 2020 (cit. on pp. 18, 19, 30, 49).

[20] R. Bujack, T. L. Turton, F. Samsel, C. Ware, D. H. Rogers, and J. Ahrens, "The Good, the Bad, and the Ugly: A Theoretical Framework for the Assessment of Continuous Colormaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 923–933, 2018 (cit. on p. 14).

[21] B. Cabral and L. C. Leedom, "Imaging Vector Fields Using Line Integral Convolution," in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, Association for Computing Machinery, 1993, pp. 263–270 (cit. on p. 19).

[22] S. Camarri, M. V. Salvetti, M. Buffoni, and A. Iollo, "Simulation of the Three-Dimensional Flow Around a Square Cylinder Between Parallel Walls at Moderate Reynolds Numbers," in *XVII Congresso AIMeTA di Meccanica Teorica e Applicata*, vol. 1, 2005, pp. 23–34 (cit. on p. 133).

[23] E. Caraballo, M. Samimy, and J. DeBonis, "Low Dimensional Modeling of Flow for Closed-Loop Flow Control," in *Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2003 (cit. on p. 78).

[24] G. F. Carrier and A. R. Robinson, "On the Theory of the Wind-Driven Ocean Circulation," *Journal of Fluid Mechanics*, vol. 12, no. 1, pp. 49–80, 1962 (cit. on p. 40).

[25] G. Chen, V. Kwatra, L.-Y. Wei, C. D. Hansen, and E. Zhang, "Design of 2D Time-Varying Vector Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 10, pp. 1717–1730, 2012 (cit. on p. 49).

[26] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang, "Vector Field Editing and Periodic Orbit Extraction Using Morse Decomposition," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 4, pp. 769–785, 2007 (cit. on p. 113).

[27] G. Chen, K. Mischaikow, R. S. Laramee, and E. Zhang, "Efficient Morse Decompositions of Vector Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 848–862, 2008 (cit. on p. 113).

[28] F. Crameri, *Scientific Colour Maps*, 2020 www.fabiocrameri.ch/lajolla.php, visited on 23-10-2020 (cit. on p. 73).

[29] F. Crameri, G. E. Shephard, and P. J. Heron, "The Misuse of Colour in Science Communication," *Nature Communications*, vol. 11, no. 1, 2020 (cit. on p. 14).

[30] J. Damon, "Generic Structure of Two-Dimensional Images Under Gaussian Blurring," *SIAM Journal on Applied Mathematics*, vol. 59, pp. 97–138, 1998 (cit. on p. 89).

[31] A. Debien, K. A. F. F. von Krbek, N. Mazellier, T. Duriez, L. Cordier, B. R. Noack, M. W. Abel, and A. Kourta, "Closed-loop Separation Control Over a Sharp Edge Ramp Using Genetic Programming," *Experiments in Fluids*, vol. 57, no. 3, 2016 (cit. on p. 113).

[32] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces*. Dover Publications, 1976, p. 503 (cit. on p. 116).

[33] M. P. Do Carmo and F. Flaherty, *Riemannian Geometry*, 1st. Birkhäuser, 1992, p. 315 (cit. on p. 116).

[34] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, "Ridges for Image Analysis," *Journal of Mathematical Imaging and Vision*, vol. 4, no. 4, pp. 353–373, 1994 (cit. on p. 89).

[35] D. Eberly, *Ridges in Image and Data Analysis*, ser. Computational Imaging and Vision. Springer, 1996, vol. 7 (cit. on pp. 15, 16).

[36] M. Edmunds, R. S. Laramee, G. Chen, N. Max, E. Zhang, and C. Ware, "Surface-based Flow Visualization," in *Computers & Graphics*, vol. 36, Elsevier, 2012, pp. 974–990 (cit. on p. 22).

[37] R. H. Enns and G. McGuire, "Forced Duffing Equation," in *Laboratory Manual for Nonlinear Physics with Maple for Scientists and Engineers*, Birkhäuser Boston, 1997, pp. 37–42 (cit. on p. 102).

[38] M. Falk and D. Weiskopf, "Output-Sensitive 3D Line Integral Convolution," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 820–834, 2008 (cit. on p. 19).

[39] M. Farazmand and G. Haller, "Computing Lagrangian Coherent Structures from their Variational Theory," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 1, 2012 (cit. on p. 90).

[40] R. A. Finkel and J. L. Bentley, "Quad Trees: A Data Structure for Retrieval on Composite Keys," *Acta Informatica*, vol. 4, pp. 1–9, 1974 (cit. on p. 95).

[41] B. Fornberg, "Generation of Finite Difference Formulas on Arbitrarily Spaced Grids," *Mathematics of Computation*, vol. 51, no. 184, pp. 699–706, 1988 (cit. on p. 37).

[42] S. F. Frisken and R. N. Perry, "Simple and Efficient Traversal Methods for Quadtrees and Octrees," *Journal of Graphics Tools*, vol. 7, p. 2002, 2002 (cit. on p. 95).

[43] G. Froyland and K. Padberg, "Almost-invariant Sets and Invariant Manifolds – Connecting Probabilistic and Geometric Descriptions of Coherent Structures in Flows," *Physica D: Nonlinear Phenomena*, vol. 238, no. 16, pp. 1507–1523, 2009 (cit. on p. 42).

[44] G. Froyland and N. Santitissadeekorn, "Optimal Mixing Enhancement," *SIAM Journal on Applied Mathematics*, vol. 77, no. 4, pp. 1444–1470, 2018 (cit. on p. 42).

[45] W. von Funck, H. Theisel, and H.-P. Seidel, "Vector Field Based Shape Deformations," *Association for Computing Machinery Transactions on Graphics*, vol. 25, no. 3, pp. 1118–1125, 2006 (cit. on pp. 49, 53).

[46] W. von Funck, H. Theisel, and H.-P. Seidel, "Explicit Control of Vector Field Based Shape Deformations," in *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, IEEE, 2007, pp. 291–300 (cit. on p. 49).

[47] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel, "Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1396–1403, 2008 (cit. on p. 133).

[48] J. Furst, S. Pizer, and D. Eberly, "Marching Cores: A Method for Extracting Cores from 3D Medical Images," in *Proceedings of the Workshop on Mathematical Methods in Biomedical Image Analysis*, IEEE Computer Society, 1996, pp. 124–130 (cit. on p. 89).

[49] J. D. Furst and S. M. Pizer, "Marching Ridges," in *Proceedings of the IASTED International Conference – Signal and Image Processing*, ACTA Press, 2001, pp. 22–26 (cit. on p. 89).

[50] C. Garth, G. Li, X. Tricoche, C. Hansen, and H. Hagen, "Visualization of Coherent Structures in Transient 2D Flows," in *Topology-Based Methods in Visualization*, Springer, 2007, pp. 1–13 (cit. on p. 90).

[51] C. Garth, F. Gerhardt, X. Tricoche, and H. Hagen, "Efficient Computation and Visualization of Coherent Structures in Fluid Flow Applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, 2007 (cit. on p. 90).

[52] C. Garth and K. I. Joy, "Fast, Memory-Efficient Cell Location in Unstructured Grids for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1541–1550, 2010 (cit. on p. 35).

[53] C. Garth, X. Tricoche, and G. Scheuermann, "Tracking of Vector Field Singularities in Unstructured 3D Time-Dependent Datasets," in *Proceedings of the Conference on Visualization '04*, IEEE Computer Society, 2004, pp. 329–336 (cit. on pp. 114, 122).

[54] J. Gauch, "Image Segmentation and Analysis via Multiscale Gradient Watershed Hierarchies," *IEEE Transactions on Image Processing*, vol. 8, no. 1, pp. 69–79, 1999 (cit. on p. 89).

[55] N. Gautier, J.-L. Aider, T. Duriez, B. R. Noack, M. Segond, and M. Abel, "Closed-Loop Separation Control Using Machine Learning," *Journal of Fluid Mechanics*, vol. 770, pp. 442–457, 2015 (cit. on p. 113).

[56] T. Germer, M. Otto, R. Peikert, and H. Theisel, "Lagrangian Coherent Structures with Guaranteed Material Separation," *Computer Graphics Forum (Proceedings EuroVis)*, vol. 30, no. 3, pp. 761–770, 2011 (cit. on pp. 42, 90).

[57] T. Gerrits, "Visualization of Second-Order Tensor Data and Vector Field Ensembles," PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2021 (cit. on p. 15).

[58] T. Gerrits, C. Rössl, and H. Theisel, "An Approximate Parallel Vectors Operator for Multiple Vector Fields," *Computer Graphics Forum (Proceedings EuroVis)*, vol. 37, no. 3, pp. 315–326, 2018 (cit. on p. 89).

[59] I. Goldhirsch, P.-L. Sulem, and S. A. Orszag, "Stability and Lyapunov Stability of Dynamical Systems: A Differential Approach and a Numerical Method," *Physica D: Nonlinear Phenomena*, vol. 27, no. 3, pp. 311–337, 1987 (cit. on p. 32).

[60] T. Günther, "Opacity Optimization and Inertial Particles in Flow Visualization," PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2016 (cit. on p. 25).

[61] T. Günther, A. Kuhn, and H. Theisel, "MCFTLE: Monte Carlo Rendering of Finite-Time Lyapunov Exponent Fields," *Computer Graphics Forum*, vol. 35, no. 3, pp. 381–390, 2016 (cit. on pp. 42, 90).

[62] T. Günther and H. Theisel, "The State of the Art in Vortex Extraction," *Computer Graphics Forum*, vol. 37, no. 6, pp. 149–173, 2018 (cit. on pp. 21, 65).

[63] A. Hadjighasem, M. Farazmand, D. Blazevski, G. Froyland, and G. Haller, "A Critical Comparison of Lagrangian Methods for Coherent Structure Detection," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 5, 2017 (cit. on p. 31).

[64] G. Haller and T. Sapsis, "Lagrangian Coherent Structures and the Smallest Finite-Time Lyapunov Exponent," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 21, no. 2, p. 023 115, 2011 (cit. on pp. 90, 102).

[65] G. Haller and G. Yuan, "Lagrangian Coherent Structures and Mixing in Two-Dimensional Turbulence," *Physica D: Nonlinear Phenomena*, vol. 147, no. 3–4, pp. 352–370, 2000 (cit. on pp. 32, 89).

[66] G. Haller, "Distinguished Material Surfaces and Coherent Structures in Three-dimensional Fluid Flows," *Physica D: Nonlinear Phenomena*, vol. 149, no. 4, pp. 248–277, 2001 (cit. on pp. 32, 89).

[67] G. Haller, "Lagrangian Coherent Structures from Approximate Velocity Data," *Physics of Fluids*, vol. 14, no. 6, pp. 1851–1861, 2002 (cit. on pp. 32, 89).

[68] G. Haller, "Lagrangian Coherent Structures," *Annual Review of Fluid Mechanics*, vol. 47, no. 1, pp. 137–162, 2015 (cit. on p. 31).

[69] R. W. Hamming, *Numerical Methods for Scientists and Engineers*. McGraw-Hill, Dover Publications, 1962 (1st), 1973 (2nd) (cit. on p. 11).

[70] Han-Wei Shen and D. Kao, "UFLIC: A Line Integral Convolution Algorithm for Visualizing Unsteady Flows," in *Proceedings of IEEE Visualization Conference '97*, IEEE, 1997, pp. 317–322 (cit. on p. 19).

[71] F. Hanisch, "Marching Square," in *CGEMS - Computer Graphics Educational Materials*, The Eurographics Association, 2004 (cit. on p. 70).

[72] R. Haralick, "Ridges and Valleys on Digital Images," *Computer Vision, Graphics and Image Processing*, vol. 22, pp. 28–38, 1983 (cit. on p. 89).

[73] H.-C. Hege and D. Stalling, "Fast LIC with Piecewise Polynomial Filter Kernels," in *Mathematical Visualization - Algorithms and Applications*, 1998, pp. 295–314 (cit. on p. 19).

[74] J. Helman and L. Hesselink, "Representation and Display of Vector Field Topology in Fluid Flow Data Sets," *Computer*, vol. 22, no. 8, pp. 27–36, 1989 (cit. on pp. 18, 21).

[75] J. Helman and L. Hesselink, "Visualizing Vector Field Topology in Fluid Flows," *IEEE Computer Graphics and Applications*, vol. 11, no. 3, pp. 36–46, 1991 (cit. on pp. 18, 21).

[76] M. Hlawatsch, F. Sadlo, and D. Weiskopf, "Hierarchical Line Integration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 8, 2011 (cit. on p. 50).

[77] L. Hofmann and F. Sadlo, "The Dependent Vectors Operator," *Computer Graphics Forum*, vol. 38, no. 3, pp. 261–272, 2019 (cit. on p. 42).

[78] M. Hummel, R. Bujack, K. I. Joy, and C. Garth, "Error estimates for Lagrangian flow field representations," *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers*, pp. 7–11, 2016 (cit. on pp. 42, 49).

[79] J. Kasten, J. Reininghaus, W. Reich, and G. Scheuermann, "Toward the Extraction of SaddlePperiodic Orbits," in *Topological Methods in Data Analysis and Visualization III*, Springer, 2014, pp. 55–69 (cit. on pp. 113, 125).

[80] G. L. Kindlmann, R. S. J. Estépar, S. M. Smith, and C. F. Westin, "Sampling and visualizing creases with scale-space particles," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1415–1424, 2009 (cit. on p. 119).

[81] G. Kindlmann, E. Reinhard, and S. Creem, "Face-based Luminance Matching for Perceptual Colormap Generation," in *Proceedings of the Conference on Visualization '02*, IEEE Computer Society Press, 2002, pp. 299–306 (cit. on p. 14).

[82] G. Kindlmann, X. Tricoche, and C.-F. Westin, "Delineating White Matter Structure in Diffusion Tensor MRI with Anisotropy Creases," *Medical Image Analysis*, vol. 11, no. 5, pp. 492–502, 2007 (cit. on p. 89).

[83] R. V. Klassen and S. J. Harrington, "Shadowed Hedgehogs: A Technique for Visualizing 2D Slices of 3D Vector Fields," in *Proceedings of the Conference on Visualization '91*, IEEE Computer Society Press, 1991, pp. 148–153 (cit. on p. 19).

[84] T. Klein and T. Ertl, "Scale-space Tracking of Critical Points in 3D Vector Fields," in *Topology-Based Methods in Visualization*, Springer, 2007, pp. 35–49 (cit. on p. 114).

[85] J. J. Koenderink and A. J. van Doorn, "Local Features of Smooth Shapes: Ridges and Courses," in *Geometric Methods in Computer Vision II*, International Symposium on Optics, Imaging, and Instrumentation, vol. 2031, 1993, pp. 2–13 (cit. on p. 89).

[86] A. Kuhn, W. Engelke, C. Rössl, M. Hadwiger, and H. Theisel, "Time Line Cell Tracking for the Approximation of Lagrangian Coherent Structures with Subgrid Accuracy," *Computer Graphics Forum*, vol. 33, no. 1, pp. 222–234, 2014 (cit. on p. 42).

[87] A. Kuhn, C. Rössl, T. Weinkauf, and H. Theisel, "A Benchmark for Evaluating FTLE Computations," in *2012 IEEE Pacific Visualization Symposium*, 2012, pp. 121–128 (cit. on pp. 67, 92, 99).

[88] M. Langbein, G. Scheuermann, and X. Tricoche, "An Efficient Point Location Method for Visualization in Large Unstructured Grids.," in *VMV 2003*, 2003, pp. 27–35 (cit. on p. 35).

[89] R. Laramee, D. Weiskopf, J. Schneider, and H. Hauser, "Investigating Swirl and Tumble Flow with a Comparison of Visualization Techniques," in *Proceedings of the Conference on Visualization '04*, IEEE Computer Society, pp. 51–58 (cit. on p. 113).

[90] R. S. Laramee, G. Erlebacher, C. Garth, T. Schafhitzel, H. Theisel, X. Tricoche, T. Weinkauf, and D. Weiskopf, "Applications of Texture-Based Flow Visualization," *Engineering Applications of Computational Fluid Mechanics*, vol. 2, no. 3, pp. 264–274, 2008 (cit. on p. 19).

[91] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf, "The State of the Art in Flow Visualization: Dense and Texture-Based Techniques," *Computer Graphics Forum*, vol. 23, no. 2, pp. 203–221, 2004 (cit. on pp. 19, 49).

[92] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post, "Topology-Based Flow Visualization, The State of the Art," in *Topology-based Methods in Visualization*, Springer, 2007 (cit. on p. 49).

[93] F. Lekien, C. Coulliette, A. J. Mariano, E. H. Ryan, L. K. Shay, G. Haller, and J. Marsden, "Pollution Release Tied to Invariant Manifolds: A Case Study for the Coast of Florida," *Physica D: Nonlinear Phenomena*, vol. 210, no. 1-2, pp. 1–20, 2005 (cit. on p. 89).

[94] F. Lekien and S. D. Ross, "The Computation of Finite-Time Lyapunov Exponents on Unstructured Meshes and for Non-Euclidean Manifolds," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 1, 2010 (cit. on p. 95).

[95] R. Li, L. Liu, L. Phan, S. Abeysinghe, C. Grimm, and T. Ju, "Polygonizing Extremal Surfaces with Manifold Guarantees," in *Proceedings of the 14th Association for Computing Machinery Symposium on Solid and Physical Modeling*, Association for Computing Machinery, 2010, pp. 189–194 (cit. on p. 89).

[96] T. Lindeberg, "Edge Detection and Ridge Detection with Automatic Scale Selection," in *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*, ser. CVPR '96, IEEE Computer Society, 1996, pp. 465– (cit. on p. 89).

[97] D. Lipinski and K. Mohseni, "A Ridge Tracking Algorithm and Error Estimate for Efficient Computation of Lagrangian Coherent Structures," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 1, 2010 (cit. on p. 90).

[98] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Association for Computing Machinery SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987 (cit. on pp. 15, 70, 138).

[99] G. Machado, S. Boblest, T. Ertl, and F. Sadlo, "Space-Time Bifurcation Lines for Extraction of 2D Lagrangian Coherent Structures," *Computer Graphics Forum*, vol. 35, no. 3, pp. 91–100, 2016 (cit. on p. 42).

[100] S. Mann and A. Rockwood, "Computing Singularities of 3D Vector Fields with Geometric Algebra," in *Proceedings of the Conference on Visualization '02*, IEEE Computer Society, 2002, pp. 283–289 (cit. on p. 114).

[101] S. Marschner and R. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering," in *Proceedings of the Conference on Visualization '94*, IEEE Computer Society, 1994, pp. 100–107 (cit. on p. 35).

[102] G. T. Mase, R. E. Smelser, G. E. Mase, and J. S. Rossmann, *Continuum Mechanics for Engineers*, 3rd. CRC Press, Computational Mechanics and Applied Analysis, 2009, p. 398 (cit. on p. 32).

[103] N. Max and T. Weinkauf, "Critical Points of the Electric Field from a Collection of Point Charges," in *Topology-Based Methods in Visualization II*, ser. Mathematics and Visualization, Springer, 2009, pp. 101–114 (cit. on p. 114).

[104] B. H. McCormick, T. A. Defanti, and M. D. Brown, "Visualization in Scientific Computing," in *Advances in Computers*, C, vol. 21, 1987, pp. 247–307 (cit. on p. 11).

[105] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen, "Over Two Decades of Integration-Based, Geometric Flow Visualization," *Computer Graphics Forum*, vol. 29, no. 6, pp. 1807–1829, 2010 (cit. on pp. 22, 30).

[106] M. Mittasch, P. Gross, M. Nestler, *et al.*, "Non-Invasive Perturbations of Intracellular Flow Reveal Physical Principles of Cell Organization," *Nature Cell Biology*, vol. 20, no. 3, pp. 344–351, 2018, ISSN: 1465-7392 (cit. on p. 113).

[107] H. Miura and S. Kida, "Identification of Tubular Vortices in Turbulence," *Journal of the Physical Society of Japan*, vol. 66, pp. 1331–1334, 1997 (cit. on p. 89).

[108] S. Musuvathy, E. Cohen, J. Damon, and J.-K. Seong, "Principal Curvature Ridges and Geometrically Salient Regions of Parametric B-spline Surfaces," *Computer-Aided Design*, vol. 43, pp. 756–770, 2011 (cit. on p. 89).

[109] B. T. Nadiga and B. P. Luce, "Global Bifurcation of Shilnikov Type in a Double-Gyre Ocean Model," *Journal of Physical Oceanography*, vol. 31, no. 9, pp. 2669–2690, 2001 (cit. on p. 40).

[110] P. Nardini, M. Chen, F. Samsel, R. Bujack, M. Bottinger, and G. Scheuermann, "The Making of Continuous Colormaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 8, 2020 (cit. on p. 14).

[111] R. Netzel and D. Weiskopf, "Texture-Based Flow Visualization," *Computing in Science & Engineering*, vol. 15, no. 6, pp. 96–102, 2013 (cit. on pp. 19, 20).

[112] B. R. Noack, I. Mezić, G. Tadmor, and A. Banaszuk, "Optimal Mixing in Recirculation Zones," *Physics of Fluids*, vol. 16, no. 4, pp. 867–888, 2004 (cit. on p. 113).

[113] T. Oster, "New Visualization Techniques for Engineering Simulations," PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2019 (cit. on p. 13).

[114] E. Ott, *Chaos in Dynamical Systems*. Cambridge University Press, 2002 (cit. on p. 31).

[115] R. Peikert and M. Roth, "The Parallel Vectors Operator - A Vector Field Visualization Primitive," in *Proceedings of the Conference on Visualization '99*, 1999, pp. 263–270 (cit. on p. 89).

[116] R. Peikert and F. Sadlo, "Topology-guided Visualization of Constrained Vector Fields," in *Topology-based Methods in Visualization*, ser. Mathematics and Visualization, Springer, 2007, pp. 21–34 (cit. on p. 113).

[117] R. Peikert and F. Sadlo, "Height Ridge Computation and Filtering for Visualization," in *IEEE Pacific Visualization Symposium*, 2008, pp. 119–126 (cit. on p. 16).

[118] R. Peikert and F. Sadlo, "Flow Topology Beyond Skeletons: Visualization of Features in Recirculating Flow," in *Topology-Based Methods in Visualization II*, Springer, 2009, pp. 145–160 (cit. on p. 113).

[119] S. M. Pizer, C. A. Burbeck, J. M. Coggins, D. S. Fritsch, and B. S. Morse, "Object Shape Before Boundary Shape: Scale-space Medial Axes.," *Journal of Mathematical Imaging and Vision*, vol. 4, no. 3, pp. 303–313, 1994 (cit. on p. 89).

[120] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matković, and H. Hauser, "The State of the Art in Topology-Based Visualization of Unsteady Flow," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1789–1811, 2011 (cit. on pp. 16, 31).

[121] A. Pobitzer, R. Peikert, R. Fuchs, H. Theisel, and H. Hauser, "Filtering of FTLE for Visualizing Spatial Separation in Unsteady 3D Flow," in *Topological Methods in Data Analysis and Visualization II*, Springer, 2012 (cit. on p. 90).

[122] S. Popinet, "Free Computational Fluid Dynamics," *ClusterWorld*, vol. 2, no. 6, 2004 (cit. on pp. 80, 103).

[123] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch, "Feature Extraction and Visualisation of Flow Fields," in *Eurographics 2002 State of the Art Reports*, Eurographics Association, 2002, pp. 69–100 (cit. on pp. 18, 19, 21).

[124] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-Time Hatching," in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, Association for Computing Machinery, 2001 (cit. on p. 49).

[125] B. Preim and C. Botha, Eds., *Visual Computing for Medicine*, 2nd. Elsevier, 2014 (cit. on p. 35).

[126] S. E. Reed, O. Kreylos, S. Hsi, L. H. Kellogg, G. Schladow, M. B. Yikilmaz, H. Segale, J. Silverman, S. Yalowitz, and E. Sato, "Shaping Watersheds Exhibit: An Interactive, Augmented Reality Sandbox for Advancing Earth Science Education," *American Geophysical Union Fall Meeting*, vol. ED34A-01, 2014 (cit. on p. 14).

[127] W. Reich, D. Schneider, C. Heine, A. Wiebel, G. Chen, and G. Scheuermann, "Combinatorial Vector Field Topology in Three Dimensions," in *Topological Methods in Data Analysis and Visualization II*, Springer, 2012, pp. 47–59 (cit. on pp. 113, 125).

[128] I. B. Rojo, M. Gross, and T. Günther, "Accelerated Monte Carlo Rendering of Finite-Time Lyapunov Exponents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 708–718, 2020 (cit. on p. 90).

[129] I. B. Rojo and T. Günther, "Vector Field Topology of Time-Dependent Flows in a Steady Reference Frame," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 280–290, 2020 (cit. on p. 80).

[130] F. Sadlo and D. Weiskopf, "Time-Dependent 2-D Vector Field Topology: An Approach Inspired by Lagrangian Coherent Structures," *Computer Graphics Forum*, vol. 29, no. 1, pp. 88–100, 2010 (cit. on p. 42).

[131] F. Sadlo and R. Peikert, "Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1456–1463, 2007 (cit. on pp. 49, 89, 90).

[132] F. Sadlo and R. Peikert, "Visualizing Lagrangian Coherent Structures and Comparison to Vector Field Topology," in *Topology-Based Methods in Visualization II*, ser. Mathematics and Visualization. Springer, 2009, pp. 15–29 (cit. on p. 90).

[133] F. Sadlo, A. Rigazzi, and R. Peikert, "Time-dependent Visualization of Lagrangian Coherent Structures by Grid Advection," in *Topology-Based Methods in Visualization II*, ser. Mathematics and Visualization, Springer, 2009, pp. 151–165 (cit. on p. 90).

[134] J. Sahner, T. Weinkauf, N. Teuber, and H.-C. Hege, "Vortex and Strain Skeletons in Eulerian and Lagrangian Frames," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 980–990, 2007 (cit. on p. 89).

[135] T. Salzbrunn, H. Jänicke, T. Wischgoll, and G. Scheuermann, "The State of the Art in Flow Visualization: Partition-based Techniques," in *Simulation and Visualization 2008*, SCS Publishing House, 2008, pp. 75–92 (cit. on p. 49).

[136] F. Samsel, T. L. Turton, P. Wolfram, and R. Bujack, "Intuitive Colormaps for Environmental Visualization," in *Workshop on Visualisation in Environmental Sciences (EnvirVis)*, The Eurographics Association, 2017 (cit. on p. 14).

[137] A. Sanderson, G. Chen, X. Tricoche, and E. Cohen, "Understanding Quasi-Periodic Fieldlines and their Topology in Toroidal Magnetic Fields," in *Topological Methods in Data Analysis and Visualization II*, ser. Mathematics and Visualization, Springer, 2012, pp. 125–140 (cit. on p. 113).

[138] G. Scheuermann, H. Hagen, H. Kruger, M. Menzel, and A. Rockwood, "Visualization of Higher Order Singularities in Vector Fields," in *Proceedings of the Conference on Visualization '97*, IEEE Computer Society, pp. 67–74 (cit. on p. 21).

[139] G. Scheuermann, H. Hagen, H. Kruger, and A. Rockwood, "Visualizing Critical Points of Arbitrary Poincaré -Index," in *Scientific Visualization Conference (Dagstuhl '97)*, IEEE Computer Society, 1997, pp. 277–277 (cit. on p. 21).

[140] B. Schindler, R. Fuchs, S. Barp, J. Waser, K. Matković, A. Pobitzer, and R. Peikert, "Lagrangian Coherent Structures for Design Analysis of Revolving Doors," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2159–2168, 2012 (cit. on p. 90).

[141] B. Schindler, R. Peikert, R. Fuchs, and H. Theisel, "Ridge Concepts for the Visualization of Lagrangian Coherent Structures," in *Topological Methods in Data Analysis and Visualization II*, ser. Mathematics and Visualization, Springer, 2012, pp. 221–235 (cit. on pp. 16, 42, 90).

[142] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit an Object-Oriented Approach to 3D Graphics*, 4.1. Kitware, 2018 (cit. on p. 19).

[143] A. Seveso. (2020, www.burdu976.com). Alberto Seveso - Illustration and Photography (cit. on p. 1).

[144] S. C. Shadden, *Lagrangian Coherent Structures - Analysis of Time-dependent Dynamical Systems Using Finite-time Lyapunov Exponents*, 2020
shaddenlab.berkeley.edu/uploads/LCS-tutorial, visited on 20-12-2020 (cit. on pp. 40, 42).

[145] S. C. Shadden, F. Lekien, and J. E. Marsden, "Definition and Properties of Lagrangian Coherent Structures from Finite-Time Lyapunov Exponents in Two-Dimensional aperiodic Flows," *Physica D: Nonlinear Phenomena*, vol. 212, no. 7, 2005 (cit. on pp. 32, 40, 42, 90, 94, 100).

[146] S. C. Shadden, F. Lekien, J. D. Paduan, F. P. Chavez, and J. E. Marsden, "The correlation between surface drifters and coherent structures based on high-frequency radar data in Monterey Bay," *Deep Sea Research Part II: Topical Studies in Oceanography*, vol. 56, no. 3—5, pp. 161–172, 2009 (cit. on p. 89).

[147] S. C. Shadden, M. Astorino, and J.-F. Gerbeau, "Computational Analysis of an Aortic Valve Jet with Lagrangian Coherent Structures," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 1, 2010 (cit. on p. 113).

[148] J. Shen, T. T. Medjo, and S. Wang, "On a Wind-driven, Double-gyre, Quasi-geostrophic Ocean Model: Numerical Simulations and Structural Analysis," *Journal of Computational Physics*, vol. 155, pp. 387–409, 1999 (cit. on p. 39).

[149] S. Shimokawa and T. Matsuura, "Chaotic Behaviors in the Response of a Quasigeostrophic Oceanic Double Gyre to Seasonal External Forcing," *Journal of Physical Oceanography*, vol. 40, no. 7, pp. 1458–1472, 2010 (cit. on p. 40).

[150] N. Smith and S. van der Walt, *MPL Colormaps*, 2015, bids.github.io/colormap, visited on 14-01-2021 (cit. on pp. 73, 126).

[151] D. Stalling and H.-C. Hege, "Fast and Resolution Independent Line Integral Convolution," in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, Association for Computing Machinery, 1995, pp. 249–256 (cit. on pp. 19, 23).

[152] G. Strang, *Computational Science and Engineering*. Wellesley-Cambridge Press, 2007, p. 750 (cit. on p. 37).

[153] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2016, p. 584 (cit. on p. 116).

[154] Y. K. Suh, "Periodic Motion of a Point Vortex in a Corner Subject to a Potential Flow," *Journal of the Physical Society of Japan*, vol. 62, no. 10, pp. 3441–3445, 1993 (cit. on p. 113).

[155] G. Tadmor and A. Banaszuk, "Observer-based Control of Vortex Motion in a Combustor Recirculation Region," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 5, pp. 749–755, 2002 (cit. on p. 113).

[156] A. C. Telea, *Data Visualization - Principles and Practice*, 2nd. CRC Press Taylor & Francis Group, 2015, p. 612 (cit. on pp. 15, 33–35, 49).

[157] H. Theisel and H.-P. Seidel, "Feature Flow Fields," in *Eurographics IEEE VGTC Symposium on Visualization*, G.-P. Bonneau, S. Hahmann, and C. D. Hansen, Eds., The Eurographics Association, 2003 (cit. on pp. 64, 66, 114).

[158] H. Theisel, "Designing 2D Vector Fields of Arbitrary Topology," *Computer Graphics Forum*, vol. 21, no. 3, pp. 595–604, 2002 (cit. on p. 48).

[159] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel, "Grid-Independent Detection of Closed Stream Lines in 2D Vector Fields.," in *VMV*, vol. 4, 2004, pp. 421–428 (cit. on p. 113).

[160] X. Tricoche, G. Kindlmann, and C.-F. Westin, "Invariant Crease Lines for Topological and Structural Analysis of Tensor Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1627–1634, 2008 (cit. on p. 89).

[161] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen, "Topology Tracking for the Visualization of Time-dependent Two-Dimensional Flows," *Computers & Graphics*, vol. 26, no. 2, pp. 249–257, 2002 (cit. on p. 114).

[162] G. Turk, "Texture Synthesis on Surfaces," in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, Association for Computing Machinery, 2001, pp. 347–354 (cit. on p. 49).

[163] M. Üffinger, F. Sadlo, and T. Ertl, "A Time-Dependent Vector Field Topology Based on Streak Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 3, pp. 379–392, 2013 (cit. on p. 90).

[164] A. Vaxman, M. Campen, O. Diamanti, D. Bommes, K. Hildebrandt, M. B.-C. Technion, and D. Panozzo, "Directional Field Synthesis, Design, and Processing," SIGGRAPH '17, 2017 (cit. on pp. 33, 49).

[165] W. Von Funck, H. Theisel, and H.-P. Seidel, "Implicit Boundary Control of Vector Field Based Shape Deformations," in *Proceedings of the 12th International Conference on Mathematics of Surfaces*, Springer, 2007, pp. 154–165 (cit. on p. 49).

[166] D. F. Watson, *A Guide to the Analysis and Display of Spatial Data*, 1st. Pergamon, 1992, p. 314 (cit. on p. 15).

[167] R. Wegenkittl, E. Groller, and W. Purgathofer, "Animating Flow fields: Rendering of Oriented Line Integral Convolution," in *Computer Animation, Conference Proceedings*, IEEE Computer Society, 1997, pp. 15–21 (cit. on p. 19).

[168] L.-Y. Wei and M. Levoy, "Texture Synthesis over Arbitrary Manifold Surfaces," in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, Association for Computing Machinery, 2001, pp. 355–360 (cit. on p. 49).

[169] T. Weinkauf, H.-C. Hege, and H. Theisel, "Advected Tangent Curves: A General Scheme for Characteristic Curves of Flow Fields," *Computer Graphics Forum*, vol. 31, pp. 825–834, 2012 (cit. on pp. 23, 25).

[170] T. Weinkauf and H. Theisel, "Streak Lines as Tangent Curves of a Derived Vector Field," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1225–1234, 2010 (cit. on pp. 23, 24, 50).

[171] T. Weinkauf, "Extraction of Topological Structures in 2D and 3D Vector Fields," PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2008 (cit. on pp. 18, 21, 114, 122).

[172] T. Weinkauf, J. Sahner, H. Theisel, and H.-C. Hege, "Cores of Swirling Particle Motion in Unsteady Flows," *Proceedings of the Conference on Visualization '07*, vol. 13, no. 6, pp. 1759–1766, 2007 (cit. on p. 113).

[173] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel, "Topological Construction and Visualization of Higher Order 3D Vector Fields," *Computer Graphics Forum*, vol. 23, no. 3, pp. 469–478, 2004 (cit. on p. 48).

[174] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel, "Feature Flow Fields in Out-of-Core Settings," in *Topology-based Methods in Visualization*, ser. Mathematics and Visualization, Springer, 2007, pp. 51–64 (cit. on p. 114).

[175] T. Weinkauf, H. Theisel, A. Van Gelder, and A. Pang, "Stable Feature Flow Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 770–780, 2011 (cit. on p. 114).

[176] M. Weldon, T. Peacock, G. B. Jacobs, M. Helu, and G. Haller, "Experimental and Numerical Investigation of the Kinematic Theory of Unsteady Separation," *Journal of Fluid Mechanics*, vol. 611, 2008 (cit. on p. 89).

[177] T. Wilde, C. Rössl, and H. Theisel, "Recirculation Surfaces for Flow Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 946–955, 2019 (cit. on pp. 42, 153).

[178] T. Wilde, C. Rössl, and H. Theisel, "Flow Map Processing by Space-Time Deformation," in *Advances in Visual Computing*, G. Bebis, Z. Yin, E. Kim, J. Bender, K. Subr, B. C. Kwon, J. Zhao, D. Kalkofen, and G. Baciu, Eds., Springer, 2020, pp. 236–247 (cit. on p. 147).

[179] T. Wischgoll and G. Scheuermann, "Detection and Visualization of Closed Streamlines in Planar Flows," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 165–172, 2001 (cit. on p. 113).

[180] S. Wolligandt, T. Wilde, C. Rössl, and H. Theisel, "A Modified Double Gyre with Ground Truth Hyperbolic Trajectories for Flow Visualization," *Computer Graphics Forum*, vol. to appear, no. to appear, to appear, 2020 (cit. on p. 42).

[181] H. Wright, *Introduction to Scientific Visualization*. Springer, 2007, p. 147 (cit. on p. 11).

[182] J. Xiang, V. Tutino, K. Snyder, and H. Meng, "CFD: Computational Fluid Dynamics or Confounding Factor Dissemination? The Role of Hemodynamics in Intracranial Aneurysm Rupture Risk Assessment," *American Journal of Neuroradiology*, vol. 35, no. 10, pp. 1849–1857, 2014 (cit. on p. 129).

[183] E. Zhang, J. Hays, and G. Turk, "Interactive Tensor Field Design and Visualization on Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 94–107, 2007 (cit. on p. 49).

[184] E. Zhang, K. Mischaikow, and G. Turk, "Vector Field Design on Surfaces," *Association for Computing Machinery Transactions on Graphics*, vol. 25, no. 4, pp. 1294–1326, 2006 (cit. on p. 49).