Technische Universität Wien

Dissertation

Visualizing Local Properties and Characteristic Structures of Dynamical Systems

ausgeführt
zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Leitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller,
Institut 186 für Computergraphik,

eingereicht
an der Technischen Universität Wien,
Technisch-Naturwissenschaftliche Fakultät,

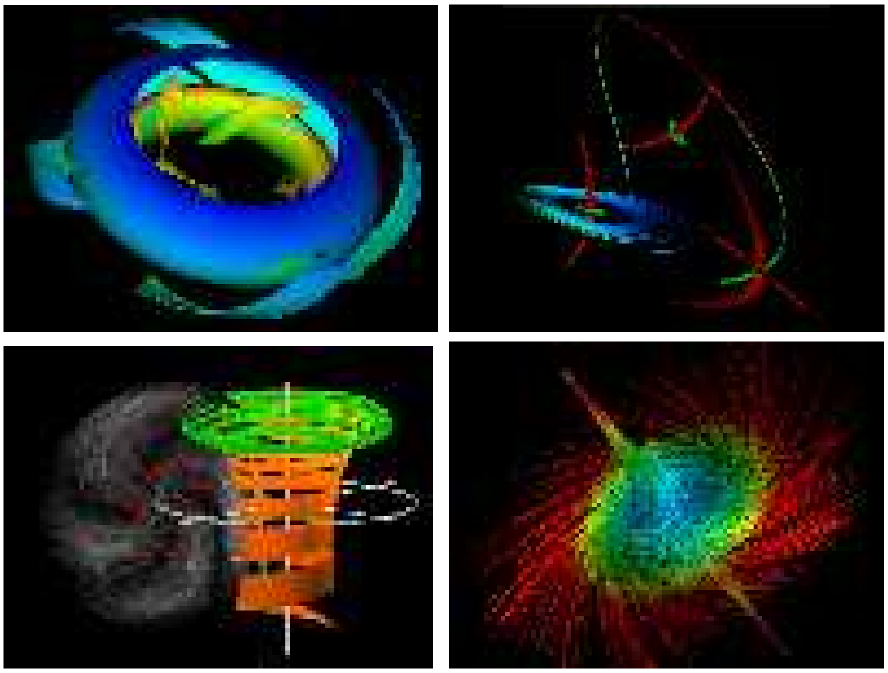
von
Dipl.-Ing. Helwig Löffelmann,
Matrikelnummer 8926287,
Seitenstettengasse 5/21,
A-1010 Wien, Österreich,
geboren am 16. April 1971 in Wien.

Wien, im November 1998.

Helwig Löffelmann

Visualizing Local Properties and Characteristic Structures of Dynamical Systems

(PhD Thesis)



http://www.cg.tuwien.ac.at/~helwig/diss/
http://www.cg.tuwien.ac.at/research/vis/dynsys/
mailto:helwig@cg.tuwien.ac.at

Abstract, Kurzfassung

(engl.) One of the most important technological developments in the 20$^{\text{th}}$ century is the use of computers in many areas of daily life, work, and research. Due to the new abilities in processing data, the amount of information that is handled, is much larger compared to previous times. As a consequence, the graphical representation, i.e., the *visualization* of data, established itself as a useful approach to investigate large data-sets. Among other application areas, the visualization of *dynamical systems* (together with flow visualization) is a very important research field. Either sampled data like a simulation of a wind tunnel or analytic models of real world phenomena, e.g., models of chemical reactions or food chains, are visualized for further investigation. Many useful techniques have been developed in this area during the past few years. In this thesis some additional contributions to this research field are presented.

After a brief introduction to the visualization of dynamical systems and a short overview over the state of the art, the new contributions to this research field are presented: *Stream arrows* improve the use of stream surfaces for the visualization of three-dimensional flow data. Additional (local) information is displayed and the problem of occlusion is diminished by transparency modulation. *Poincaré maps* are used for the visualization of periodic or quasi-periodic flows. Extracting a 2D map as essential information and combining it with selective cues from the 3D flow, allows to efficiently investigate even complex flows with periodic behavior. *Critical points* of a flow provide important information – usually mathematical analysis starts with their identification and investigation. A method featuring a direct representation of flow near critical points as well as the visualization of higher-order local information is presented. *Characteristic trajectories* are important components in an abstract description of dynamical systems. Similarly to the visualization of critical points, direct visualization of the vicinity of these curves in phase space is used to enhance the information provided through visualization.

(dt.) Eine der wichtigsten technologischen Entwicklungen im 20. Jahrhundert ist die Verwendung von Computern in vielen Bereichen des täglichen Lebens, der Arbeit und der Forschung. Aufgrund der neuen Möglichkeiten zur Datenverarbeitung, ist die Menge an Information, die behandelt wird, viel größer als in früheren Zeiten. Als eine Folge davon hat sich die graphische Repräsentation, also die Visualisierung von Daten, als ein nützlicher Ansatz zur Untersuchung von großen Datensätzen etabliert. Die Visualisierung von dynamischen Sys-

temen (gemeinsam mit der Strömungsvisualiserung) macht unter anderen Anwendungsgebieten einen sehr wichtigen Forschungsbereich aus. Es müssen entweder erhobene Daten wie die einer Simulation eines Windtunnels oder analytische Modelle von Phänomenen der realen Welt, beispielsweise Modelle chemischer Reaktionen oder Nahrungsketten, visualisiert werden, um sie zu untersuchen. In den letzten Jahren wurden viele nützliche Techniken in diesem Bereich entwickelt. In dieser Dissertation werden ein paar zusätzliche Beiträge zu diesem Forschungsfeld vorgestellt.

Nach einer Einführung in die Visualisierung dynamischer Systeme und einem kurzen Überblick über den aktuellen Stand der Wissenschaft werden die neuen Beiträge zu diesem Forschungsfeld präsentiert: Strömungspfeile verbessern die Verwendung von Strömungsflächen für die Visualisierung von drei-dimensionalen Strömungsdaten. Zusätzliche (lokale) Information wird dargestellt und das Verdeckungsproblem wird durch die lokale Variation von Transparenz reduziert. Poincaréabbildungen werden für die Visualisierung von periodischem bzw. quasi-periodischem Flüssen verwendet. Komplexe Flüsse mit periodischem Charakter werden effizient untersucht, indem eine 2D Abbildung als essentielle Information extrahiert und diese mit einzelnen Merkmalen des 3D Flusses kombiniert dargestellt wird. Fixpunkte eines Flusses bieten wichtige Informationen – gewöhnlich startet die mathematische Analyse mit deren Identifikation und Untersuchung. Es wird eine Methode vorgestellt, welche sowohl die direkte Darstellung der Strömungsdaten in der Nähe der Fixpunkte als auch die Visualisierung von lokaler Information höherer Ordnung beinhaltet. Charakteristische Trajektorien sind wichtige Komponenten abstrakter Beschreibungen von dynamischen Systemen. Ähnlich zur Visualisierung von Fixpunkten, wird die direkte Visualisierung der Nachbarschaft dieser Kurven im Phasenraum verwendet, um das lokale Verhalten besser veranschaulichen zu können.

Related publications

This thesis is based on the following publications:

- H. Löffelmann, Z. Szalavári, and E. Gröller: **Local Analysis of Dynamical Systems – Concepts and Interpretation**. Published in *Proc. of the 4^{th} International Conference in Central Europe on Computer Graphics and Visualization '96*, pp. 170-180, Univ. of West Bohemia, Plzen, Czech Republic, February, 1996.
- H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer: **Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems**. Published in *The Visual Computer* **13**(8), pp. 359-369, 1997.
- H. Löffelmann and E. Gröller: **DynSys3D: A workbench for developing advanced visualization techniques in the field of three-dimensional dynamical systems**. Published in *Proc. of the 5^{th} International Conference in Central Europe on Computer Graphics and Visualization '97*, pp. 301-310, Univ. of West Bohemia, Plzen, Czech Republic, February, 1997.
- H. Löffelmann, L. Mroz, and E. Gröller: **Hierarchical Streamarrows for the Visualization of Dynamical Systems**. Published in *Proc. of the 8^{th}* EUROGRAPHICS *Workshop on Visualization in Scientific Computing*, pp. 203-211, Boulogne sur Mer, France, April, 1997. Republished in W. Lefer and M. Grave (eds.): Visualization in Scientific Computing '97, Springer, pp. 155-163.
- H. Löffelmann, T. Kučera, and E. Gröller: **Visualizing Poincaré Maps Together With the Underlying Flow**. Published in H.-C. Hege and K. Polthier (eds.): *Mathematical Visualization, Algorithms, Applications, and Numerics*, Springer, pp. 315–328.
- A. Milik, P. Szmolyan, H. Löffelmann, and E. Gröller: **Geometry of Mixed-Mode Oscillations in the 3-d Autocatalator**. Published in *International Journal of Bifurcation and Chaos* **8**(3), pp. 505-520, 1998.
- H. Löffelmann and E. Gröller: **Enhancing the Visualization of Characteristic Structures in Dynamical Systems**. Published in *Proc. of the 9^{th}* EUROGRAPHICS *Workshop on Visualization in Scientific Computing*, pp. 35-46, Blaubeuren, Germany, April, 1998. Republished in D. Bartz (ed.): Visualization in Scientific Computing '98, Springer, pp. 59–68.
- H. Löffelmann, H. Doleisch, and E. Gröller: **Visualizing Dynamical Systems near Critical Points**. Published in *Proc. of the Spring Conference on Computer Graphics and its Applications 1998*, pp. 175-184, Budmerice, Slovakia, April, 1998.

Contents

Chapter 1

Introduction

> The purpose of computing is insight, not numbers.
> *Richard W. Hamming (1915-1998)*

> The real voyage of discovery consists not in seeking new landscapes, but in having new eyes.
> *Marcel Proust (1871-1927)*

This thesis presents research work concerning the visualization of dynamical systems. The introduction is split into three parts: first (Sect. 1.1), visualization, scientific visualization, and some fields of applications are described. Afterwards, a brief introduction to dynamical systems is given (Sect. 1.2). Finally, the context of this work, namely the visualization of dynamical systems is discussed (Sect. 1.3).

1.1 Visualization, scientific visualization

> '**visualize**': to form a mental vision, image, or picture of (something not visible or present to the sight, or of an abstraction); to make visible to the mind or imagination.
> *Oxford English Dictionary, 2^{nd} edition, 1989*

In science often large and/or complex collections of data have to be processed. Usually it is not suitable for human researchers to investigate such data-sets by reading lists of numbers or other textual representations. The mapping of information into graphs or images, i.e., visualization, was identified as a powerful tool for data investigation already a long time ago. Leonardo da Vinci

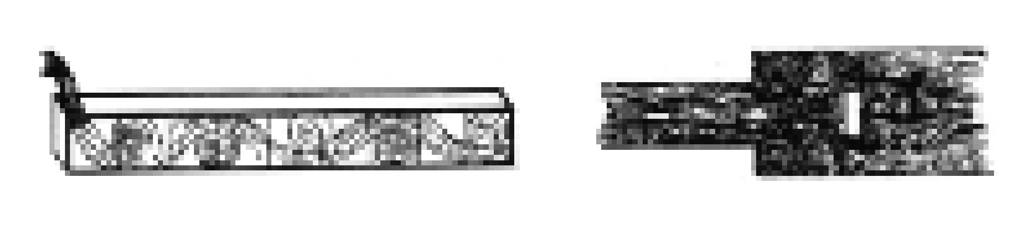


Figure 1.1: Two examples of early flow visualization by Leonardo da Vinci (images out of "Frontiers of Scientific Visualization" by Pickover and Teksbury [65]).

(1452–1519), for example, already used drawings to communicate scientific results. Fig. 1.1 shows two examples of his work. More recently, the extensive use of computers for data processing generated a new need for elaborated visualization techniques. In the early 1990s annual-conference series, solely focusing on visualization, e.g., the "Eurographics Workshop on Visualization in Scientific Computing" or the "Ieee Conference on Visualization", were established. Ten years later already a few compendia on visualization are available as comprehensive text books, for example, "Scientific Visualization" by Gregory Nielson, Hans Hagen, and Heinrich Müller [59]. To illustrate the role visualization is playing at the end of the first millennium, some of the most important application fields are listed below:

Medical data visualization – (anatomic) data is acquired through measurement devices, e.g., MRI or CT, which is then presented using volume visualization techniques, e.g., direct volume rendering or isosurface extraction.

Flow visualization – vector data, either computed by flow simulation, or measured data using experimental setups, is plotted for the purpose of data investigation. For example, the design of new aircrafts can be checked using simulation and visualization without constructing expensive prototypes.

Geographic information systems (GIS) and visualization – for hundrets of years up to now maps are used as visualization of geographic data. Techniques like color coding, height fields, iso-lines, and icons, are used to show topographic information like mountains, rivers, etc., together with additional information, for example, temperature.

Information visualization – big databases, multi-modal data, and abstract data (usually non-scientific data) increasingly require appropriate visualization techniques. Business data visualization (charts, diagrams, and graphs) is already widly used to illustrate economic data and relationships.

Visualization of microscopic data – molecules and/or atomic structures investigated in biology, chemistry, and physics, increasingly are visualized for analysis. Also data acquired by non-optical microscopes usually needs to be visualized before investigation can start.

Large-scale data and visualization – astronomy, for instance, deals with data that is simulated or measured at a scale that prohibits direct investigation in most cases. Again, visualization can help to "fit", for example, the entire universe into the study room of an astronomer.

Architectural visualization – Planning of urban regions as well as buildings is enhanced by visualization methods. New buildings, are visualized on the basis of computer aided design (CAD) data together with existing context. This allows to evaluate plans before actual construction.

Archeology and visualization – to investigate archaic cultures, for instance, visualization enables researchers to imagine life, habits, rites, etc., in former ages. Reconstruction of historic buildings using visualization is an area of increasing importance.

Visualization aims to maximally exploit the visual channel to the human user for information communication. Visual resolution, spatial as well as temporal, and 'resolution' of the human perceptional capabilities restrict the content of information that can be conveyed to the human user through visualization.

1.2 Dynamical systems, vector fields

Vector fields typically represent flows on discrete locations in space. Various grid structures (regular grid, curvilinear grid, etc.) are in use. Dynamical systems on the other hand are usually defined analytically, for example, by a set of differential equations.

In the previous section various fields of applications of visualization briefly have been presented. The work presented here closely fits into the flow visualization area, since flow data and dynamical systems match up quite good with respect to visualization – many techniques developed for flow visualization are useful for dynamical system visualization and vica versa.

Dynamical systems are a description of the evolution of some (usually inter-dependent) entities within a common system. A food chain, for instance, describing the who-eats-whom relation between several species, sharing some common place of living, is modeled as a dynamical system. The predator-prey model by Lotka and Volterra [72] is an example of such a food chain. It describes the evolution of a system consisting of one species of consumers (predators) and another one of resource (prey). Basically, it consists of two numbers representing the amount of both species present in the system at a certain time, and a description of the temporal change of these numbers due to the given setting of the system.

More general, a *dynamical system* is a set of n numbers $\mathbf{x}[i]$ – usually n is called the *dimensionality* of the system – that vary according to specific set of rules. These *system variables* $\mathbf{x}[i]$ build up the *state* $\mathbf{x} \in \Omega \subseteq \mathbf{R}^n$ of the system, where Ω usually is called the *phase space* of the dynamical system. Some specific value $\mathbf{x}(t)$ represents the actual configuration of the system at a specific point in *time* t. In addition to system variables and time, usually *parameters* $\mathbf{p}[j]$ are part of the rules of evolution. Their different values span a *class* of dynamical systems over $\Pi \subseteq \mathbf{R}^m$, called *parameter space*.

A *continuous* dynamical system usually is given by a set of ordinary differential equations (ODEs) [4], whereas a *discrete* dynamical system is specified by difference equations:

$$\begin{array}{rcccll} \dot{\mathbf{x}}(t) & = & \mathrm{d}\mathbf{x}(t)/\mathrm{d}t & = & \mathbf{f_p}(\mathbf{x}(t), t) & \text{(continuous case, } t \in \mathbf{R}) \\ \Delta\mathbf{x}(t) & = & \mathbf{x}(t{+}1) - \mathbf{x}(t) & = & \mathbf{f_p}(\mathbf{x}(t), t) & \text{(discrete case, } t \in \mathbf{Z}) \end{array} \quad (1.1)$$

There are other possibilities to describe the dynamics of a dynamical system, for instance, discrete dynamical systems are sometimes written as $\mathbf{x}(t{+}1) = \mathbf{f_p}(\mathbf{x}(t), t)$. Usually most of the alternatives are either compatible to the notation presented above, or can be transformed such that they match the above definition.

A dynamical system is called *time-dependent*, if the rules determining the dynamics depend on time, i.e., $\mathbf{f_p}$ itself depends on time t (see Eqs. 1.1). If, on the other hand, these rules are static over time, a *steady*, i.e., a *time-independent* system is given. In this case $\mathbf{f_p}$ only depends on the present state of the system $\mathbf{x}(t)$ and parameters $\mathbf{p}$.

In the case of the Lotka and Volterra model, a two-dimensional, continuous, and steady dynamical system is given: the state $\mathbf{x} \in \Omega = [0, \infty)^2$ of the system is composed of x (amount of prey) and y (predators), and two ODEs including four parameters that represent the rules of evolution:

$$\begin{array}{rcll} \dot{x} & = & r \cdot x - p \cdot x \cdot y & \text{(evolution of prey)} \\ \dot{y} & = & e \cdot p \cdot x \cdot y - m \cdot y & \text{(evolution of predators)} \end{array} \tag{1.2}$$

In this rather simple model prey is assumed to grow exponentially at a rate r ($\dot{x} = r{\cdot}x$) if no predators are present. Predators hunt a certain percentage p of prey, thereby decreasing the amount of prey proportionally ($\dot{x} = \ldots - p{\cdot}x{\cdot}y$). Hunted prey is 'used' for reproduction of predators on the basis of a certain efficiency e ($\dot{y} = e \cdot p \cdot x \cdot y - \ldots$). Opposed to reproduction of predators there is a certain rate of mortality m ($\dot{y} = \ldots - m \cdot y$).

Solutions of a dynamical system, i.e., solutions to the differential or difference equations, are called *trajectories* or *orbits*. For continuous and steady dynamical systems a trajectory $\mathcal{T}_{\mathbf{s}}(t)$ starts at a specific seed value $\mathbf{s}$ and evolves over time according to the following equation:

$$\mathcal{T}_{\mathbf{s}}(t) = \mathbf{s} + \int_{u=0}^{t} \mathbf{f}_{\mathbf{p}}(\mathcal{T}_{\mathbf{s}}(u))\,\mathrm{d}u$$

Dynamical systems usually are depicted in phase space Ω. Sometimes other spaces, e.g., $\Omega \times \mathbf{R}$ – time is added as an additional dimension – or $\Omega \times \Pi$, i.e., investigating the dynamics of an entire class of dynamical systems, are used. If the dimensionality of such a space gets too large, sub-spaces are examined instead. Returning to the Lotka and Volterra example again, one could investigate the dynamics of this model in the phase plane assuming a

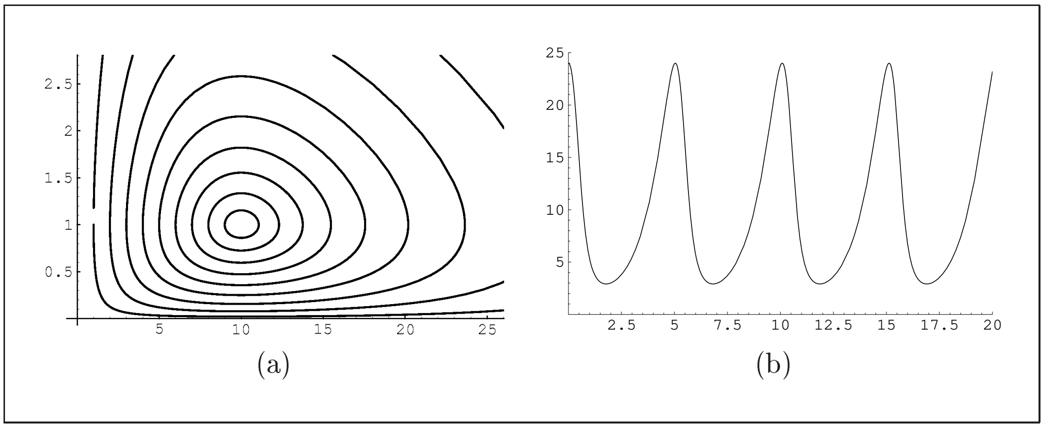


Figure 1.2: (a) Cycles of evolution in 2D phase space, and (b) evolution of variable x over time t (both computed for a predator-prey model by Lotka/Volterra).

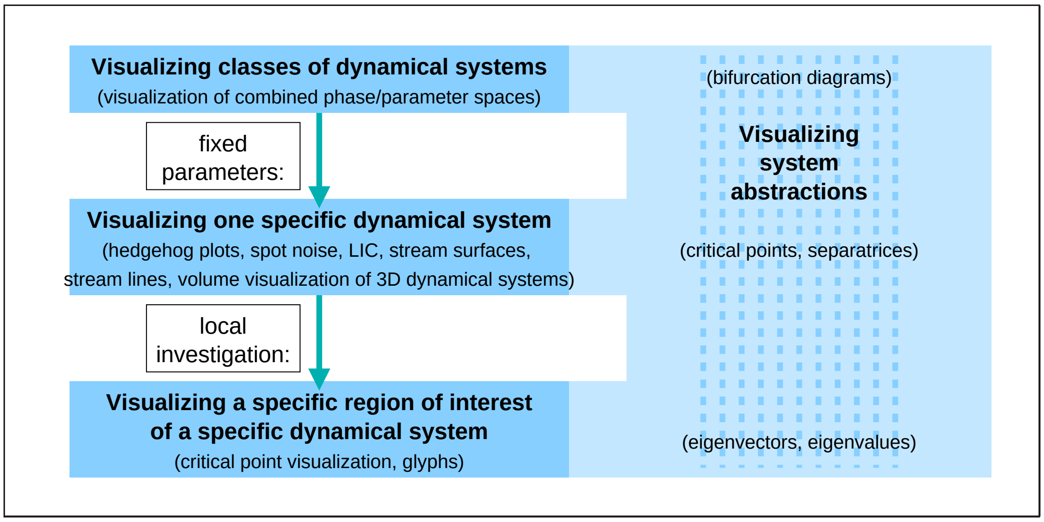


Figure 1.3: Different ways of viewing dynamical systems.

fixed set of parameter values (see Fig. 1.2(a)). Another possibility is to plot one of the state variables against time, again in the plane – in Fig. 1.2(b) the amount of prey (x) is plotted over time t.

1.3 Visualization of dynamical systems

As a dynamical system usually is a very dense representation of a multi-dimensional amount of complex information, the need for visualization is obvious. Many useful techniques are already available. Especially for two- and three-dimensional flow fields many visualization techniques have been developed in the past years [67].

However, the entirety of all kind of dynamical systems is much too diverse to be addressed by a single visualization technique. There is too much difference between, for instance, a discrete and a continuous dynamical system. In general, a proper visualization technique is dependent on the kind of data to be visualized, and the specific goal of investigation. Thus, a separation of techniques according to the specific sub-class of dynamical systems addressed, is necessary.

One possible way of classifying visualization techniques for dynamical systems is to look at the data scale they focus on. Stressing the aim of maximizing information transmission through the visual channel, it becomes clear that different visualization techniques are necessary for different scales. Investigating a specific dynamical system locally allows to view many more

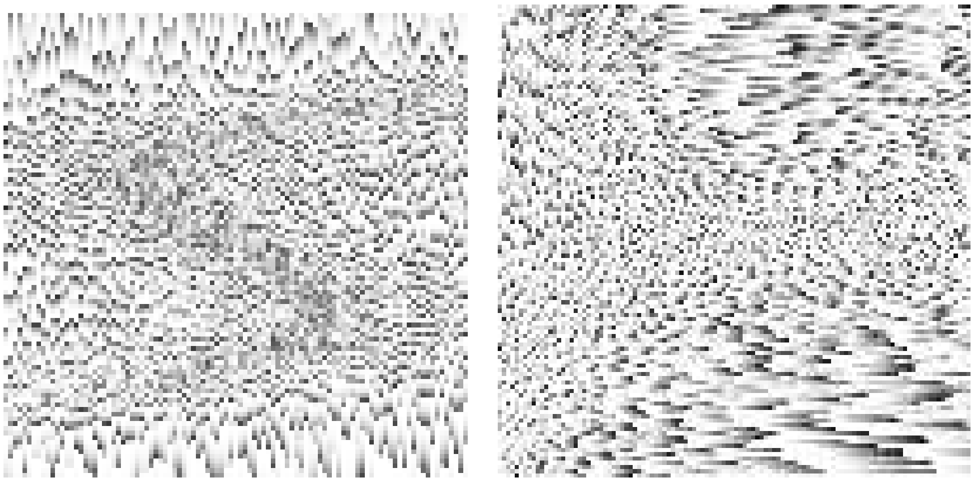


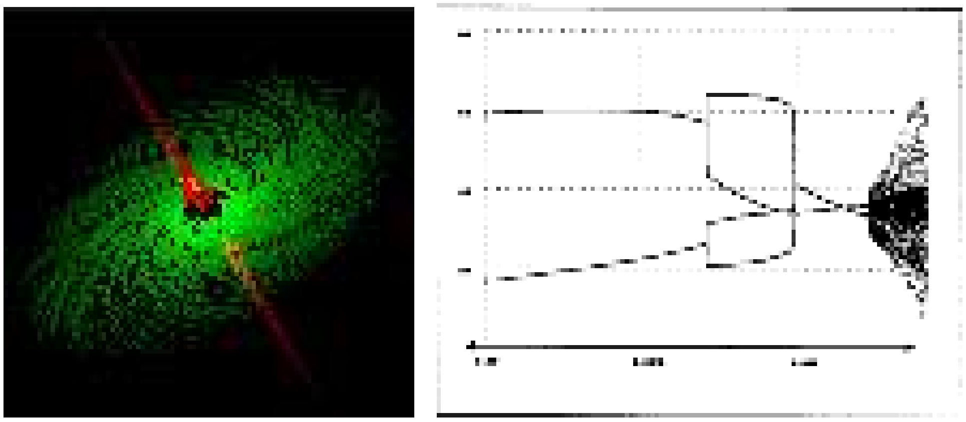
Figure 1.4: (a) Visualizing a 1D class of 1D dynamical systems – the parameter is varied along the horizontal axis. (b) Visualizing one specific 2D dynamical system.

details simultaneously than analyzing an entire class of dynamical systems. A separation into three levels of data scale is useful for identifying different kinds of visualization techniques (see Fig. 1.3):

Visualizing classes of dynamical systems – dynamical systems as defined in Eqs. 1.1 are dependent on phase space and parameter space, i.e., $\Omega \times \Pi \subseteq \mathbf{R}^{n+m}$, in case steady dynamical systems are considered. A visualization of $\mathbf{f_p}$ encoding all the dynamics is only possible, if phase space as well as parameter space are of quite low dimensionality, for example, if $n+m=2$.

In Fig. 1.4(a) a visualization of a one-dimensional class of one-dimensional dynamical systems ($\dot{x} = x^3 - x - p$) is shown. State variable x is associated with the vertical axis, whereas the only parameter p is mapped to the horizontal axis. Similarities as well as differences between systems with different parameter value p can directly be inferred from the image.

Visualizing one specific dynamical system – fixing parameters to a specific value, one member out of a class of dynamical systems is depicted. All the available dimensions of the visualization channel can be



(a) (b)

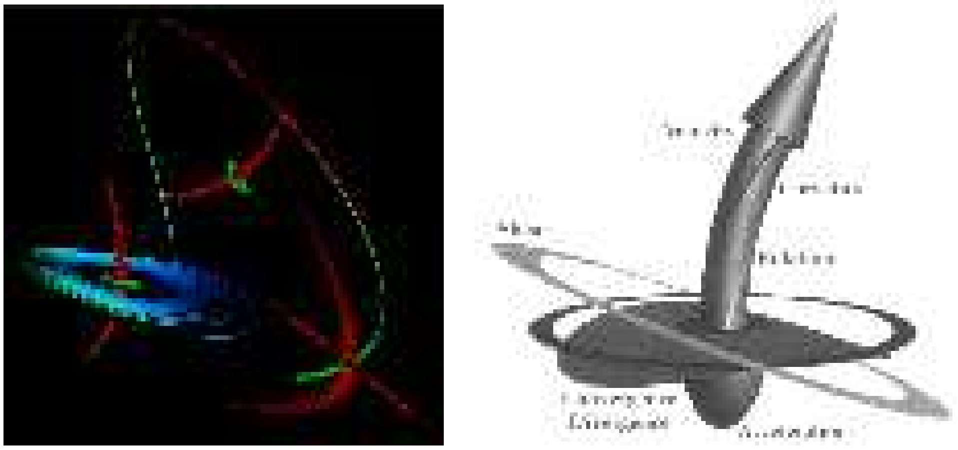
Figure 1.5: (a) Visualizing a local sub-space of interest [44]. (b) Typical bifurcation diagram.

used to represent information about the single selected system. Techniques belonging to this scale level of visualizing dynamical systems usually map the dynamics in phase space directly into visual properties. Many techniques can be found in this area, especially for system dimensions up to three. There are, however, approaches to the visualization of higher dimensional dynamical systems also [91].

Fig. 1.4(b) shows a two-dimensional visualization of a specific member out of the class of Lotka-Volterra models (cf. Eq. 1.2, $\mathbf{p} = (r\ p\ e\ m)^{\mathrm{T}} = (1\ 1\ 0.2\ 2)^{\mathrm{T}}$). The dynamics caused by this dynamical system is directly encoded by the used visualization technique.

Visualizing a specific region of interest – restricting spatial resolution to specific sub-spaces of interest enables the visualization to communicate more details. Data locally available can also be included within the visualization, while neighboring information is omitted.

Fig. 1.5(a) shows a visualization of a three-dimensional dynamical system, restricted to a spherical sub-space around the critical point of this system. Although phase space is three-dimensional this local technique avoids visual overloading while still preserving direct visualization of the system dynamics.



(a) (b)

Figure 1.6: (a) Visualizing an abstraction of a three-dimensional dynamical system [44]. (b) Visualizing the results of local analysis [19].

In general, there are two principal possibilities for designing a visualization technique: direct visualization means to directly map principal flow properties like direction and velocity to a visual representation. All the three classes of visualization mentioned above (more or less) belong to this kind of approach.

The **visualization of system abstractions**, on the other hand, means to first derive second-level properties of the flow like critical points and separatrices, and then visualize the abstract information. At any scale of the underlying data, analysis can be done first, and visualization used afterwards to convey the results. Characteristic structures like, e.g., critical points (system states where there is no motion at all) or cycles (states of a dynamical systems which reoccur after a certain period of evolution), may be extracted using dynamical system analysis, and mapped to visualization cues afterwards.

Bifurcation diagrams like the one shown in Fig. 1.5(b) depict (an approximation of) the stable sub-set for each (discrete) dynamical system (1D, vertical axis) in a one-dimensional class (horizontal axis). Bifurcations occur at parameter-value changes, where the stable sub-set changes qualitatively, e.g., at points of a phase doubling or a torus beak-down [77].

A typical result of visualizing a dynamical system after doing some analysis first, can be seen in Fig. 1.6(a). The critical points are visualized together with the results of an eigenvector and eigenvalue analysis of the system's Jacobian matrix at these points.

A sample result of visualizing derived data at specific sub-sets of phase space [19] is shown in Fig. 1.6(b). At a specific location in 3D phase space the Jacobian matrix of the dynamical system is analyzed and the derived (local) properties like, direction of flow, velocity, acceleration, rotation, etc., are visualized using a glyph.

An overview of the state of the art in visualizing dynamical systems and related fields is given in Chapter 2. Notes about terms and the local analysis of dynamical systems are given afterwards. Then, four techniques, namely, visualization by the use of stream arrows, visualization based on Poincaré maps, visualizing critical points, and the visualization of characteristic trajectories, are described in Chapters 4, 5, 6, and 7, respectively. A note on the implementation of these visualization methods is appended (Chapt. 8). Finally, a short summary is given, and conclusions are drawn. After the bibliography, a glossary of some important terms related to dynamical systems is given. The thesis concludes with appendices on the notation used and descriptions of the sample dynamical systems used.

Chapter 2

State of the art

> Lord grant me the serenity to accept the things I cannot change, the courage to change the things I can, and the wisdom to know the difference.
>
> *St. Francis of Assisi (1181-1226)*

The visualization of dynamical systems must be viewed in the context of a few related fields. Flow visualization, for example, is tightly related to it, as flow data can be seen as a special class of dynamical systems – flow, for example, usually is considered to be compressible only up to a certain level. No attracting nodes can be found in such a system. Dynamical systems, on the other hand, principally do not have any restrictions and, thus, can be considered as a super-class of flows.

Nevertheless, it is useful to distinguish between flows and dynamical systems, since often different aspects of interest are investigated through visualization. Another difference between flows and dynamical systems, which significantly influences visualization, is that flows usually are given discretely on large-sized grids whereas dynamical systems usually are given analytically by a few equations. Next to computational methods, experimental flow visualization techniques are also of interest. They provide the possibility to evaluate computational methods. Furthermore they have been inspirational for quite a few computer-based visualization techniques.

Another field related to the visualization of dynamical systems is visualizing tensor fields. In this case the type of data (vectors vs. tensors) is not compatible. However, the methods used to visualize either vector or tensor fields have several concepts in common, e.g., the use of integral curves. The aim for extracting the field topology in order to condense the content of information transported via visualization is also common to both fields.

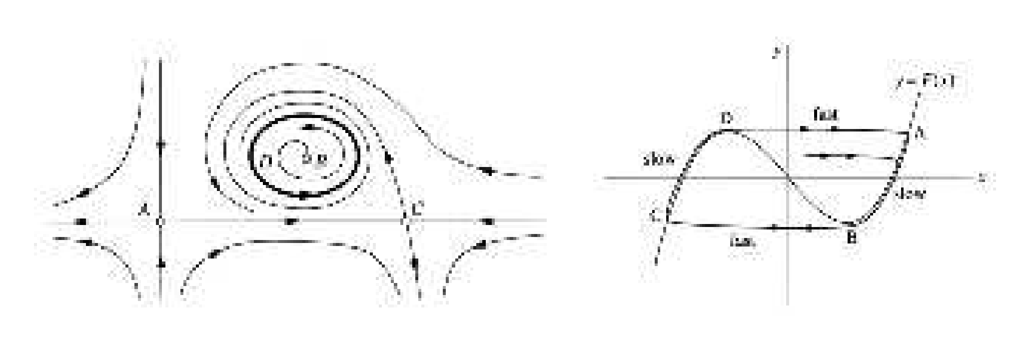


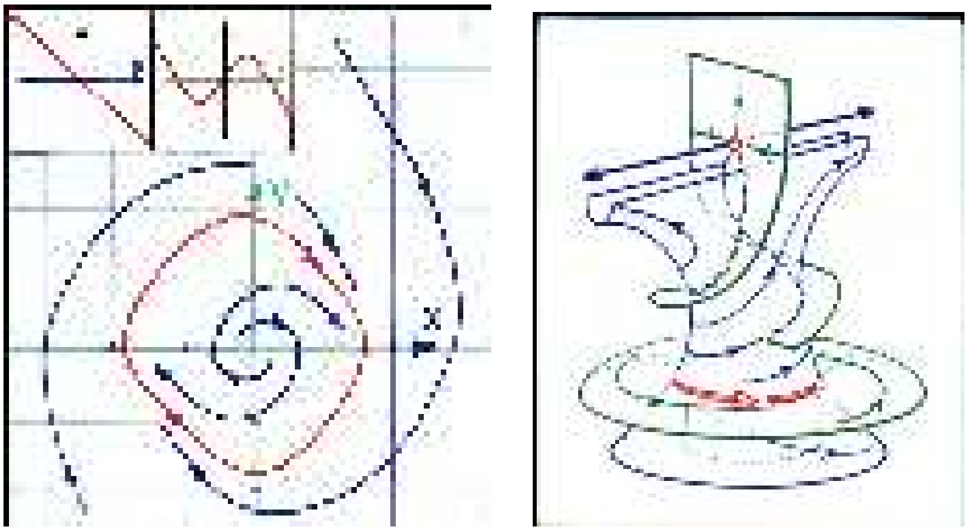
Figure 2.1: Two examples of visualization used for the communication of results gained from in-depth analysis of two-dimensional dynamical systems (images by Strogatz [80]).

The mathematical theory about ordinary differential equations (ODEs) is another important related field. It provides a common language to effectively describe dynamical systems. Furthermore, the field of computational fluid dynamics (CFD) provides a number of useful terms to characterize and describe dynamical systems. Its main focus is the simulation of flows. Both fields are kind of a basis the visualization of dynamical systems is built on.

2.1 Visualizing dynamical systems

The investigation of dynamical systems spans a wide research area. Models of systems with a state that varies over time, often are formalized using dynamical systems. Examples are food chains, econometric models, chemical reaction systems, meteorologic models, and stock market models. Usually researchers start with an in-depth analysis of the dynamical system. Afterwards visualization is used to communicate the results of the analysis. Critical sub-sets, e.g., critical points, separatrices, or bifurcation lines, are combined with additional integral cues as, for example, stream lines, stream surfaces, etc. In many text books about dynamical systems one can find images accompanying the theoretical analysis. Mapping phase space to image space is an intuitive way to visualize specific sub-sets of a two-dimensional phase space. Critical points and characteristic trajectories usually make up an important part of such illustrations. See Fig. 2.1 for two examples out of "Nonlinear Dynamics and Chaos" by Strogatz [80].

In the left graph three critical points ('A', 'B', and 'C') are denoted. Characteristic trajectories coinciding with the eigenvectors of the Jacobian



(a) (b)

Figure 2.2: Two examples of hand-drawn flow visualization (images by Abraham and Shaw [1]).

matrices associated with the critical points are added. Small arrows indicate the orientation of flow. In addition to the critical points also a cycle ('D') appears. Finally, a few additional trajectories are plotted to give an impression about the important features of the dynamical system being visualized. This type of sketch is quite usual for illustrating the most important structures of low-dimensional dynamical systems.

Another interesting book on dynamical systems is "Dynamics – The Geometry of Behavior" [1] by Abraham, a mathematician, and Shaw, an artist. Hand-drawn images are used to visualize certain characteristics of special dynamical systems. The mathematician provides knowledge concerning the important structures of the dynamical systems. The artist has an ability to clearly convey complex spatial arrangements through only a small set of visual cues. The cooperation of both results in effective depictions of dynamical systems. Two examples out of this book can be seen in Fig. 2.2.

Fig. 2.2(a) gives a sketch of a two-dimensional system. A cycle (red trajectory) around a critical point in the center is shown together with a few accompanying trajectories. Fig. 2.2(b) visualizes a dynamical system with three variables. A saddle critical point and a saddle cycle are shown in red and white. The surface structures in-between these two characteristic subsets make up the main part of the image. The sketch illustrates a rather complex relation between the critical point and the saddle cycle.

2.2 Flow visualization

Compared to visualization of dynamical systems, much more work has been done in the field of flow visualization. Experimental and/or empirical techniques have already been used for quite a long time.

In recent time flow visualization increasingly is done one a computational basis. Fluid flows as well as gaseous flows are simulated in the research field of computational fluid dynamics (CFD). Often finite element methods are used to handle complex flow structures, for instance, local solvers of the Navier-Stokes equations, which work on various kinds of grids. Usually data sets are computed that provide a huge amount of sampled vector information spread over a two- or three-dimensional domain.

Without visualization it is usually impossible to reasonably investigate such data sets. At this point flow visualization comes into play. It already provides numerous techniques to view various properties of such huge data sets, e.g., turbulences, vortical structures, separation lines, etc.

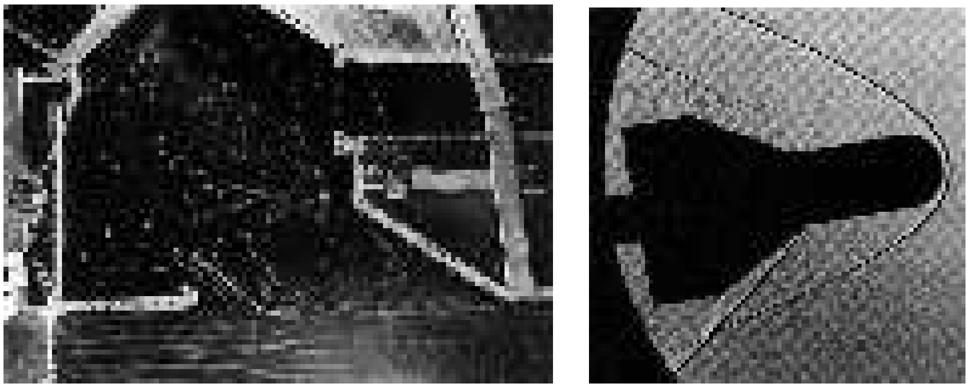
Experimental and empirical approaches

For quite a long time, researchers who deal with flows are using experimental setups to get an impression of its properties and structures, to get ideas about improvements to their work, and/or to evaluate their models. Three basic types of experimental techniques can be distinguished [67]:

Adding foreign material – Dye or magnesium powder is injected into liquid flow to visualize flow dynamics. See Fig. 2.3(a) for an example, where a model of a harbor was visualized using particles within the flow. In gaseous flows smoke or oil droplets are injected.

A problem with injecting material is that the injection process and the injected material may influence the flow. Using electrolytic techniques for generating hydrogen bubbles within the flow decreases these problems to a certain extent. Also photochemical methods are used, for instance, generating dye within the flow using a laser beam.

Applying tufts to the walls of a flow simulation, or coating certain border surfaces of interest with some viscous material like oil, visualizes



(a) (b)

Figure 2.3: Experimental flow visualization, two examples [67]: (a) particles within the flow (image by Delft Hydraulics) and (b) shadow graph technique (image by High-Speed Lab, Dept. of Aerospace Eng., Delft Univ. of Techn.).

flow behavior near objects within the flow, for example, flow close to aircraft wings in a wind tunnel.

Optical techniques – less disturbence of the flow can be achieved using optical methods. Optical properties like light refraction change at places within the flow where there are big local differences in flow density. Working with a light beam, images are generated with shadows and caustics. See Fig. 2.3(b) for an example, where shadows in the image denote shock waves within the flow.

Another visual property which changes in regions of high density gradients, is the phase of light rays. Interferometry is an example of a technique which exploits such phase shifts.

Adding heat/energy – heat can be applied to flows to artificially increase the density variation – optical techniques are then used for visualization. Shooting electrons into the flow volume is used to excite gas molecules. After being excited the molecules emit their extra energy as light particles, which visualizes flow patterns.

Although experimental methods have advantages – feedback is intuitive, no numerical errors, immediate response, etc. – there are some significant disadvantages: most severe is the fact, that experimental methods influence the flow itself. Next, experimental setups usually are time consuming and very

(a) (b) (c)

Figure 2.4: Three basic visualization techniques used for the Lotka-Volterra model: (a) hedgehog plot, (b) streamlets, and (c) LIC.

expensive. Finally, there is just a limited set of flow properties that can be visualized using experimental techniques.

In addition to experimental methods, empirical techniques – flow patterns are drawn by hand after investigation – also have a long tradition. Leonardo da Vinci used hand drawings to communicate his research results on fluid flows. More recently, Abraham and Shaw came up with visualizing flow structures by using hand-drawn images [1].

For in-depth information about experimental flow visualization techniques, see Merzkirch [55], Yang [93], and van Dyke [85].

Methods for two-dimensional flow fields

The less dimensions a dynamical system has, the easier visualization is. Techniques for the visualization of two-dimensional dynamical systems (or vector fields) already have quite a tradition in flow investigation. Hedgehog plots, also called arrow plots, usually show a large number of small arrows that indicate the flow direction at many (regularly spaced) points of the two-dimensional domain. Often arrows are normalized, so flow velocity is not encoded. This is, to prevent the display from overloading due to very long and overlapping arrows. See Fig. 2.4(a) for a hedgehog plot of the Lotka-Volterra model (see also Eq. 1.2).

More elaborated are stream line graphs. The dynamical system or flow field is integrated numerically for some specific initial points. Depending on whether the flow is time-dependent or not, streak lines, path lines, or stream lines are generated [29]. Temporal correlation of virtual particles that are moved by the flow are intuitively depicted, such that an impression of the embedded dynamics can be gained quite intuitively. See Fig. 2.4(b) for a set of streamlets for the Lotka-Volterra model.

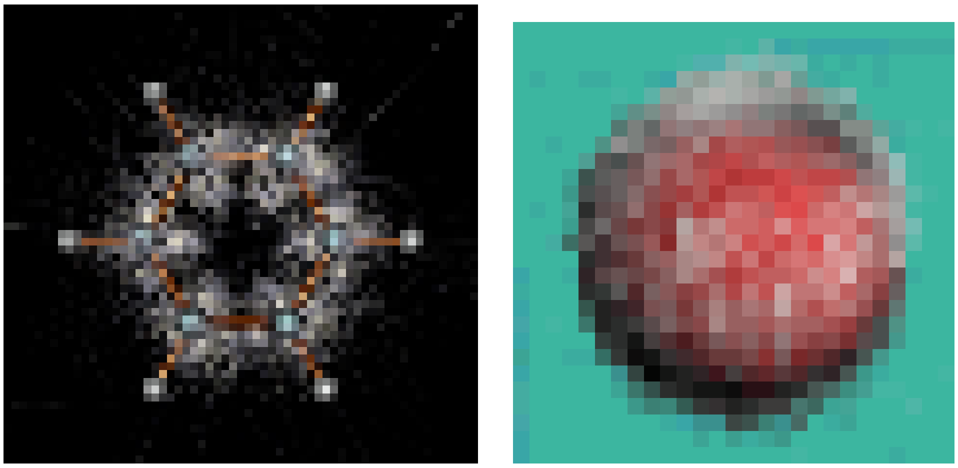
One problem with integral curves used in the visualization of continuous dynamical systems is the choice of the initial conditions. Evenly spaced seed points usually do not generate evenly spaced integral curves. Turk and Banks [84] and Jobard and Lefer [36, 37] propose methods to cope with this problem and generate evenly spaced stream lines for two-dimensional flows.

Instead of placing many integral lines over the flow domain, texture-based methods, also provide very useful results. Spot noise by van Wijk [87, 20] is generated by placing many small 'spots', for example, elongated ellipses, on the flow domain and orienting them according to the local flow direction. Different intensities are chosen for the spots. Thereby a noise texture is generated which locally is correlated with flow direction. Another texture-based technique, called line integral convolution (LIC) by Cabral and Leedom [14, 23, 24, 78, 79, 90, 89], generates similar results. A white noise texture is locally convoluted along flow trajectories. Again, a visual correlation along the flow is generated. Both techniques, spot noise and LIC, are capable of generating an overview of all the dynamics in a dynamical system. See Fig. 2.4(c) for a LIC image of the Lotka-Volterra system. Other techniques in this area are texture splats by Crawfis and Max [17], line bundles [16] and virtual ink droplets [48].

Direct visualization of 3D flows

In 3D the situation is more difficult than in 2D, since image synthesis involves a process of condensing visualization cues populating the three-dimensional domain. For example, arrow plots are usually not useful for three-dimensional flow, since depth perception of one-dimensional objects (the arrows) is poor compared to surface objects, and populating three-space with arrows easily produces overloaded images.

Representing stream lines as 1D curves, the use of stream lines or similar integral curves is difficult for the same reasons. Nevertheless, the intuitive understanding of this kind of direct representation of flow trajectories, i.e., of streamlets or stream lines, resulted in some interesting techniques, e.g., illuminated stream lines (cf. Fig. 2.5(a)) by Zöckler et al. [94] and vector field rendering (cf. Fig. 2.5(b)) by Banks [8].



(a) (b)

Figure 2.5: Two examples for rendering stream lines in 3D: (a) illuminated stream lines by Zöckler et al. [94], and (b) vector field rendering by Banks [8].

Stream ribbons show, additional to flow paths, flow rotation around trajectories [29]. A stream line is integrated through the vector field. Additionally local surface elements are used to encode local flow rotation. Either a second trajectory is connected or differential analysis of the flow is used to compute the ribbon twist. Fig. 2.6 shows a visualization of oil flow patterns at the contact surface of the flow and stream ribbons within the flow used for vortex core visualization. In this image results of an experimental setup are overlaid with results from a CFD simulation of the same flow.

In addition to stream lines and stream ribbons, stream surfaces make up an important part of flow visualization in 3D. Instead of a point, one-dimensional sets of initial conditions are used in the vector field integration step. Hultquist described, how to compute stream surfaces for dynamics over a three-dimensional domain [33]. Problems with stream surfaces are, that extensive surface parts easily occlude other parts of the visualization, and missing information about flow direction and velocity within the stream surface. Chapter 4 describes stream arrows that might be used to decrease most of the problems apparent with stream surfaces. Fig. 2.7(a) shows an example of a stream surface.

Other integral objects than streamlets or stream lines are used for flow visualization also. Stream balls by Brill et al. [12] are based on the meta balls concept. A set of initial points (seed points) is used to define an iso-surface,

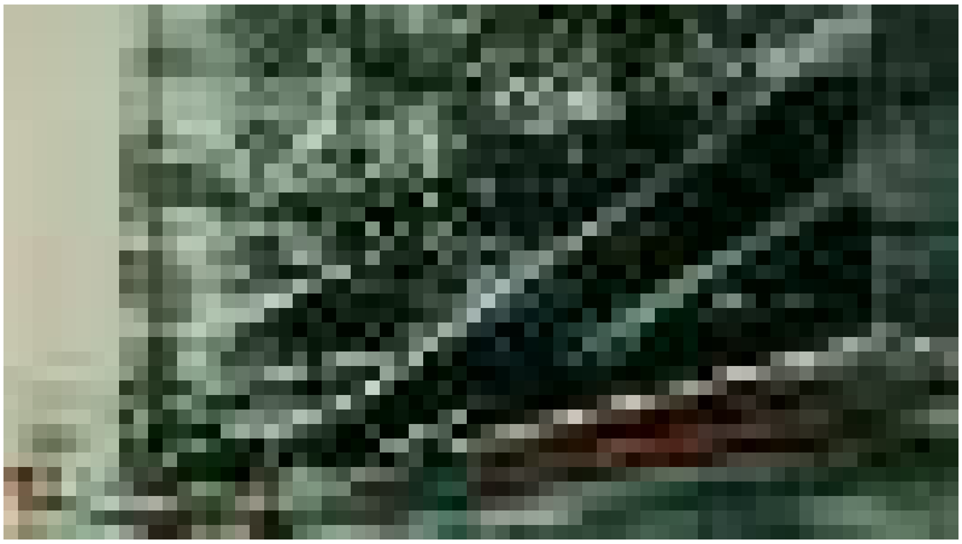
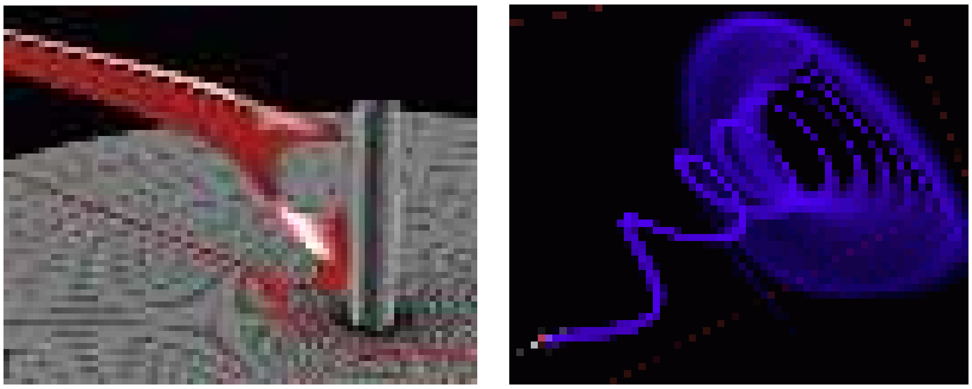


Figure 2.6: Stream ribbons show the rotation around stream lines (image by Hans-Georg Pagendarm) [62].

i.e., the stream balls, using a potential field proportional to the distance from these seed points and some user-specified threshold value. Consequently the points are moved following the underlying vector field. Thereby new points are added to the initial set and a surface-like meta object is generated. In regions of local divergence the iso-surface separates into distinct sub-surfaces, whereas in regions of local convergence the iso-surfaces related to multiple points merge and built up a coherent meta object.

To investigate flow near boundary surfaces, virtual tufts are used. Short integral objects are computed with initial conditions next to boundary surfaces. Instead of stream surfaces, a particle system can be used for flow visualization [88]. Particles are modeled as small surface parts spread over the locus of a stream surface. Transparent areas in-between the particles reduce the problem of occlusion, while the particles still give a good impression of the stream surface. Particles are drawn as small ellipses. A normal vector assigned to each surface particle is used in shading calculations.

Contrary to one- and two-dimensional visualization cues, flow volumes model the temporal evolution of an initial three-dimensional set [54]. This approach models the injection and propagation of smoke particles through the flow advection. Volume rendering is necessary to compute an image showing 3D flow visualized by the use of flow volumes. Local convergence or



(a) (b)

Figure 2.7: (a) Stream surface (image by the Data Visualization Group, NAS, NASA) [58]. (b) Flow volume (image by the Visualization group at LLNL) [18].

divergence is encoded by the density of the flow volume. Fig. 2.7(b) gives an example of a flow volume.

Visualizing local properties in 3D

Similar to dynamical systems flow data-sets also contain sub-sets of special interest. Vortex cores, for example, are very important structures in the simulation of flow around objects like airplanes, ships, turbines, etc [9, 39]. In the case of dynamical systems separatrices are especially interesting. Because of their special importance these lower-dimensional structures often are investigated in more detail. Local properties that are derived from flow derivatives, i.e., the Jacobian matrices at states of interest, are also visualized.

One approach to the visualization of special trajectories together with their local properties are the stream polygon and stream tube techniques by Schröder et al. [75]. For a certain number of sample points along the stream line of interest, the Jacobian matrix is examined. A decomposition into a symmetric and an asymmetric part yields local rotational and shear information about the flow near the investigated trajectory. This information is mapped to the geometrical properties of polygons which are assumed to be normal to the flow direction. Size, shape, and rotation of the polygons illustrate local flow properties. By connecting the edges of adjacent stream

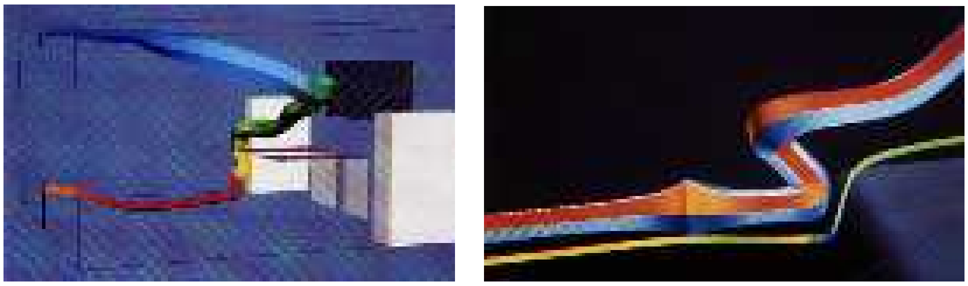


Figure 2.8: Two examples for flow visualization by the use of stream tubes (images by Schröder et al. [75]).

polygons a stream tube is generated. In Fig. 2.8 two examples of stream tubes are shown.

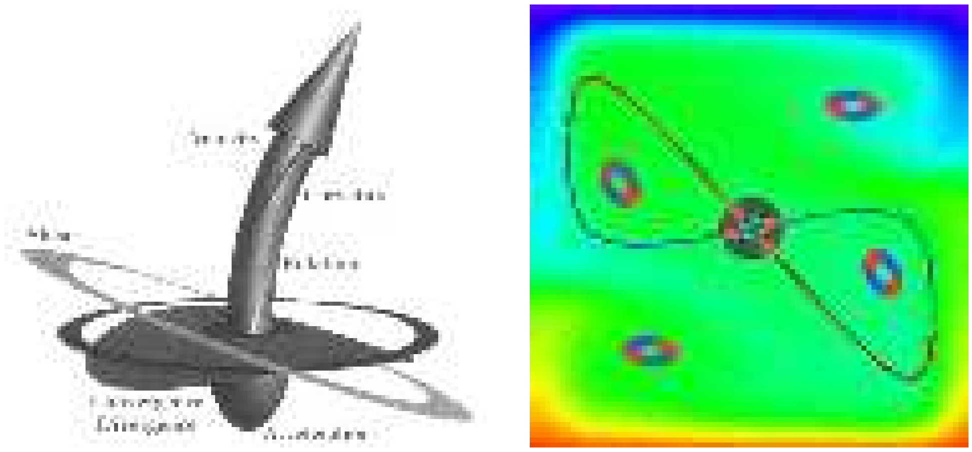
Even more 'verbose' than stream polygon and stream tube, the local flow probe [19] by de Leeuw and van Wijk represents local flow properties also derived from the Jacobian matrix. Direction and orientation, velocity, acceleration, curvature, rotation, shear, and convergence/divergence of the flow near a special state of interest are mapped to distinct geometrical properties of a rather complex glyph. See Fig. 2.9(a) for a sample glyph generated with this technique. Placing several of these glyphs, for example, along an especially important trajectory, an intuitive visualization of local properties is provided.

Happe and Rumpf [74] extended the use of icons for representing local flow characteristics near critical points of the system. See Fig. 2.9(b) for a sample image generated using this technique. Post et al. also present advanced visualization techniques on the basis of icons [68].

Visualizing the topology of vector fields

Helman and Hesselink [31] proposed to visualize the geometry of the topological structure of flow dynamics. Stream lines along the eigenvectors of critical points are used to show separatrices. Icons composed of line segments and small disks encode the Jacobian matrix near critical points. Globus et al. [27] came up with a tool to identify topological elements within data that is given at a discrete grid. Two sample images are shown in Fig. 2.10.

Overviews of work in this area are given by Levit in 1992 [43] and Asimov et al. in 1995 [7].



(a) (b)

Figure 2.9: (a) Local flow probe by de Leeuw and van Wijk [19]. (b) Icons for the visualization of local properties by Happe and Rumpf [74].

2.3 Related fields

There are quite some important fields related to flow visualization. Similar techniques, for example, are used in the area of tensor field visualization. A brief overview is given below:

Tensor field visualization – beside flow data-sets also tensor data is examined (tensor fields provide multi-dimensional data usually represented by the use of matrices). Stress propagation within certain objects like engines, turbines, etc., produce tensor data. Simulation techniques that are similar to methods known from CFD are used to compute dense data-sets of volume tensors.

One way to intuitively describe a matrix in 3D is to represent it in terms of its eigenvectors and eigenvalues. Depending on whether the eigenvalues are real or complex, all different from each other or not, the eigenvectors build up either three characteristic directions, or one surface of rotational dynamics additional to one characteristic direction.

Hyper stream lines by Delmarcelle and Hesselink [21] are a visualization concept for tensor data based on the decomposition described above. Certain characteristic curves are integrated like stream lines for flow data. Such curves follow, for example, the direction of maximum stress propagation. See Fig. 2.11 of a typical image in this area.

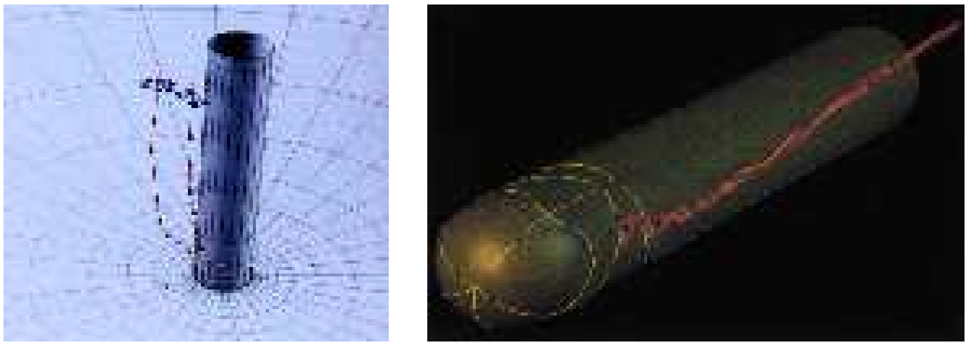


Figure 2.10: Two examples of the visualization of vector field topology (images by Helman et al. [31]).

Other related fields are:

Computational fluid dynamics (CFD) – as data to be visualized often originates from flow simulations, techniques for simulating dynamics are strongly related to the field of flow visualization. Usually the domain of flow is subdivided into a grid of many small cells. Then, the equations of pressure, motion, etc., are solved locally. Various grid structures are used, e.g., regular grids, curvilinear grids, etc.

Mathematics / ordinary differential equations (ODEs) – Lots of mathematical theory is available for the analysis of dynamical systems. The extraction of the topology of behavior is just one example. Finding critical points usually is simple compared to finding cycles or characteristic sub-sets of dimension one or higher. Advanced techniques like trapping regions must be used.

Numerics – simulating the dynamics of flow requires careful computations and advanced numerical techniques. Especially numerical integration and numerical derivation of flow characteristics are crucial components within flow visualization techniques.

Sampling and reconstruction – often flow data is given as a huge set of samples. Reconstructing the continuous solution from the discrete data is usually non-trivial and must be done carefully. Advanced interpolation and approximation techniques, for example, working on arbitrary grids, are necessary.

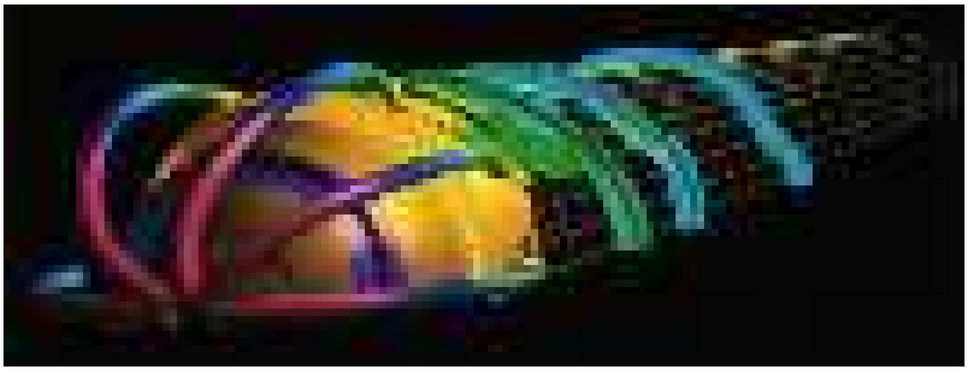


Figure 2.11: An example for tensor field visualization by Delmarcelle and Hesselink [21].

After some notes on terms and the local analysis of dynamical systems (Chapter 3), in Chapters 4, 5, 6, and 7 four approaches to the visualization of dynamical systems are described in detail. In each case a set of specific goals has to be met and thus rather different approaches emerged.

Chapter 3

Notes on the local analysis of dynamical systems

> Confusion is a word we have invented for an order which is not understood.
>
> *Henry Miller (1891-1980)*

Several terms and definitions related to the local analysis of dynamical systems are presented. Multiple terms for one and the same concept that were found in literature are put together to provide a "dictionary" of terms and to avoid potential confusion due to misleading definitions. Additionally, some important concepts which are necessary to analyze a dynamical system are briefly discussed and a new procedure to locally analyze a dynamical system's behavior near trajectory points is proposed. This chapter should give computer graphics specialists, who work on the visualization of analytically defined dynamical systems – but are not experts on the field of dynamical systems – a set of mathematical tools for a thorough investigation of the local behavior of such systems.

3.1 Introduction

Dynamical systems are found in various fields of research [5]. Usually they are given by an analytical specification or as sampled data. There are many possible ways to analyze such a system, for example, analyzing its long term behavior. An important branch of the analysis of dynamical systems is local analysis. For certain applications it is crucial to know, how initially close states will evolve with respect to each other. Flow field analysts, for example, are often interested in vortices, that may be detected by local analysis of the underlying dynamical system. Therefore this chapter concentrates on the local analysis of dynamical systems.

Scientists that are interested in dynamical systems (and the local analysis of these systems) are confronted with a lot of terms, formulas, and definitions. Non-mathematicians get easily confused by studying some of the relevant literature in the beginning. Differing terms for the same object do not help to clear up the situation as well as subtle differences in the interpretation of mathematical symbols do not simplify the understanding. This was one of the reasons to compile relevant terms that occur often in literature and to assemble the different definitions. For example, the curvature of a 3D curve can either be calculated from the Frenét formulas or by analyzing the Jacobian matrix of the dynamical system.

On the other hand it is interesting to see how some (local) attributes of a dynamical system can be derived by rather different approaches. This seems to be especially useful when some of the straightforward techniques are not possible due to incomplete or insufficient specifications. One example is the analysis of dynamical systems that are given as sampled data which do not allow the use of straightforward analytical approaches in most cases.

Before terms and definitions that are relevant for the local analysis of dynamical systems are discussed, some high-level classifications of dynamical systems are listed. Thereafter an arc from differential geometry aspects when analyzing trajectories of dynamical systems is spanned to the analysis of linear dynamical systems and its interpretation. In this sections we present well known concepts but concentrate on giving a unifying view of various terms and definitions, which are sometimes used ambiguously and interchangeably in literature. Then we discuss dynamical system analysis near special subsets of the topology of behavior to end up with a new approach to locally analyze points on trajectories.

3.2 Classifications of dynamical systems

Dynamical systems are mainly represented by a state that evolves in time. The input as well as the current state of a dynamical system determine the evolution of the system. Typically an output is generated from the state of the system [72]. This is a rather general definition of a dynamical system, where many different systems fit into. For investigating dynamical systems it is necessary to specify some characteristics that provide a subdivision into special classes of dynamical systems. Specific methods are available for some of these classes, thus such a classification can help to simplify the analysis.

An important characteristic of a dynamical system is whether it is continuous or discrete. Continuous systems (often called flows) are given by differential equations whereas discrete dynamical systems (often called maps) are specified by difference equations [83]. Autonomous systems are characterized by the fact that input and output are omitted from the definition [72].

An important criterion for the analysis of a dynamical system is whether it is time-dependent or not [41, 42]. For time-dependent dynamical systems the function that specifies $\dot{\mathbf{x}}$ (continuous case) or $\Delta\mathbf{x}_n$ (discrete case) depends on the time itself whereas for time-independent systems this function does not change over time.

For the analysis it is very important whether a dynamical system is linear or not. Linear dynamical systems are simple to analyze as opposed to non-linear systems, which typically do have intricate dynamical behavior [83]. Often linearization at specific locations is used to get insights into these complex non-linear dynamical systems.

Using linearization, another classification of dynamical systems is crucial to separate simple cases from more complex ones. Hyperbolic dynamical systems can be analyzed by linearization efficiently, whereas non-hyperbolic systems may cause major troubles in combination with linearization [1, 27]. Hyperbolic systems are structurally stable, i.e., small perturbations of the system parameters do not change the qualitative behavior of the system. Non-hyperbolic systems are difficult to investigate, occur rarely and can be considered the transitional phase between two hyperbolic systems of different nature [72].

3.3 Differential geometry and terms

The solution of a continuous dynamical system is a trajectory $\mathcal{T}_{\mathbf{s}}(t)$ as defined by Eq. 3.1 [40, 69]. Any point on the trajectory is given by its parameter t and an initial state $\mathbf{s}$ of the system. Parameter t can be interpreted as the time passed since the system evolved from $\mathbf{s}$. Note, that Eq. 3.1 is a "recursive" definition (integral equation) that cannot be expressed explicitly in most cases.

$$\mathcal{T}_{\mathbf{s}}(t) = \mathbf{s} + \int_{u=0}^{t} \mathbf{f}_{\mathbf{p}}(\mathcal{T}_{\mathbf{s}}(u))\,\mathrm{d}u \tag{3.1}$$

Differential geometry includes the analysis of curves and surfaces in higher dimensions. The construction of a local coordinate system (Frenét-Frame) helps to get insight into local characteristics of a spatial curve, e.g., curvature and torsion [10, 30]. Local analysis of trajectories requires a good working knowledge of various terms of differential geometry. They are shortly discussed in the following.

Given a parameterized curve $\mathcal{C}(t)$ in three-space a re-parameterization is possible such that the curve's new parameter s is equal to the arc length of curve $\mathcal{C}$ in the parameter interval $[0, s)$. In respect to these distinct parameters derivations of curve $\mathcal{C}$ are written differently:

$$\dot{\mathcal{C}} = \frac{\mathrm{d}\mathcal{C}}{\mathrm{d}t},\ \ddot{\mathcal{C}} = \frac{\mathrm{d}^2\mathcal{C}}{\mathrm{d}t^2},\ \text{etc.,} \qquad \mathcal{C}' = \frac{\mathrm{d}\mathcal{C}}{\mathrm{d}s},\ \mathcal{C}'' = \frac{\mathrm{d}^2\mathcal{C}}{\mathrm{d}s^2},\ \text{etc.} \tag{3.2}$$

By the use of these derivations a local coordinate system (Frenét-Frame) can be built at a curve point by the curve's tangent vector $\mathbf{t}_\mathcal{C} = \mathcal{C}'$, its principal normal $\mathbf{n}_\mathcal{C} = \mathcal{C}''/|\mathcal{C}''|$, and its binormal $\mathbf{b}_\mathcal{C} = \mathbf{t}_\mathcal{C} \times \mathbf{n}_\mathcal{C}$. These three vectors span an orthonormal basis at a curve point. Note, that $\mathbf{n}_\mathcal{C}$ and $\mathbf{b}_\mathcal{C}$ are ambiguous when the curve is locally equal to a straight line.

By building the Frenét-Frame at a point on the curve the curvature κ and the torsion τ of curve $\mathcal{C}$ at this point can be derived in a straightforward way from the orthonormal basis [10]:

$$\frac{\mathrm{d}}{\mathrm{d}s}\begin{pmatrix}\mathbf{t}_\mathcal{C}\\ \mathbf{n}_\mathcal{C}\\ \mathbf{b}_\mathcal{C}\end{pmatrix} = \begin{pmatrix}0 & -\kappa & 0\\ \kappa & 0 & -\tau\\ 0 & \tau & 0\end{pmatrix}\cdot\begin{pmatrix}\mathbf{t}_\mathcal{C}\\ \mathbf{n}_\mathcal{C}\\ \mathbf{b}_\mathcal{C}\end{pmatrix} \quad\Longrightarrow\quad \kappa = \left|\frac{\mathrm{d}\mathbf{t}_\mathcal{C}}{\mathrm{d}s}\right|, \quad \tau = \left|\frac{\mathrm{d}\mathbf{b}_\mathcal{C}}{\mathrm{d}s}\right|$$

Curvature κ and torsion τ of curve $\mathcal{C}$ can be described in other terms as well. For example, the curvature of a curve can be written as $1/r$, when r is the radius of the osculating circle [13]. As a third possibility, κ can be derived by the following procedure: assuming α to be the angle enclosed by the curve's tangent and the line running through $\mathcal{C}(s)$ and some point $\mathcal{C}(s+\Delta s)$, slightly ahead on the curve, the curvature κ can be calculated as $\kappa = \lim_{\Delta s\to 0} \alpha/\Delta s$.

Torsion can be similarly derived by a differential quotient. Assuming β to be the angle enclosed by a line through $\mathcal{C}(s)$ and $\mathcal{C}(s+\Delta s)$ the rectifying plane (spanned by $\mathbf{t}_\mathcal{C}$ and $\mathbf{b}_\mathcal{C}$), the torsion τ can be calculated as $\tau = \lim_{\Delta s\to 0} \beta/\Delta s$ [13].

3.4 Dynamical systems $\leftrightarrow$ Babylon of terms

This section discusses some of the often used terms in combination with dynamical system analysis. Most of the terms might be well-known to the reader, but often several differing terms are used in literature to denote the same concept or object. To avoid possible confusion about these sometimes interchangeably used terms a clarifying survey is appropriate.

We start with operator ∇, which is often used to define other important terms for the analysis of dynamical systems. It builds up a vector of the partial derivatives of its operand and is defined as shown in Eq. 3.3 [13]. If ∇'s operand $f(\mathbf{x})$ is a scalar function, then $\nabla f(\mathbf{x})$ is called the gradient of f [13]. If ∇'s operand $\mathbf{f}(\mathbf{x})$ is a vector function, then $\nabla\mathbf{f}$ is the Jacobian matrix $\mathbf{J} = \partial\mathbf{f}/\partial\mathbf{x}$ of $\mathbf{f}(\mathbf{x})$ [19].

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial \mathbf{x}[1]} \\ \frac{\partial}{\partial \mathbf{x}[2]} \\ \vdots \end{pmatrix}, \quad \operatorname{grad} f(\mathbf{x}) = \nabla f(\mathbf{x}), \quad \mathbf{J} = \nabla\mathbf{f}(\mathbf{x}) = \partial\mathbf{f}/\partial\mathbf{x} \tag{3.3}$$

An often used (scalar) term is the divergence of a flow $\operatorname{div}\mathbf{f}(\mathbf{x})$. It can be written as $\nabla \cdot \mathbf{f}(\mathbf{x})$ or as the trace Tr of $\mathbf{f}$'s Jacobian $\nabla\mathbf{f}$ [13]:

$$\operatorname{div}\mathbf{f}(\mathbf{x}) = \nabla \cdot \mathbf{f}(\mathbf{x}) = Tr(\nabla\mathbf{f}) = \sum_i (\partial\mathbf{f}/\partial\mathbf{x})_{i,i} \tag{3.4}$$

The divergence basically describes the local amount of outgoing or incoming flow at a specific location of the dynamical system. It is 0, if the amount of incoming flow is equal to the amount of outgoing flow.

Another important term for the local analysis of dynamical systems is the rotation vector of a flow: $\operatorname{rot}\mathbf{f}(\mathbf{x})$ [61, 75]. This attribute of a flow is often named vorticity instead of rotation and abbreviated by ω [29]. As a third term sometimes curl is used instead of rotation [29]. The vorticity/rotation/curl of a flow is defined as follows:

$$\omega = \operatorname{rot}\mathbf{f}(\mathbf{x}) = \operatorname{curl}\mathbf{f}(\mathbf{x}) = \nabla \times \mathbf{f}(\mathbf{x}) \tag{3.5}$$

Vector $\operatorname{rot}\mathbf{f}(\mathbf{x})$ describes the rotation axis and its length the rotation velocity, which is given at state $\mathbf{x}$. Note, that some references define the vorticity slightly different as $\omega = (1/2) \cdot \operatorname{rot}\mathbf{f}(\mathbf{x})$.

A scalar term related to the vorticity as defined above is the stream vorticity Ω [29, 75]. It is the cosine of the angle enclosed by the vorticity vector and the flow vector $\mathbf{f}(\mathbf{x})$. This term characterizes the type of rotation in the system. If Ω is 1, the flow rotates around the flow vector $\mathbf{f}(\mathbf{x})$, whereas a value of 0 implies, that either there is no vorticity or the flow rotates in a plane which also contains the direction of the flow.

$$\Omega = \frac{\mathbf{f} \cdot \omega}{|\mathbf{f}| \cdot |\omega|} = \frac{\mathbf{f} \cdot (\nabla \times \mathbf{f})}{|\mathbf{f}| \cdot |\nabla \times \mathbf{f}|} \tag{3.6}$$

Just slightly different from the above definition is the specification of helicity [19]. Furthermore the helicity density H_d as given in the literature is just the same as helicity [67]. A value of 0 means exactly the same as no stream vorticity, but helicity increases proportional to the length of ω and $\mathbf{f}$. It is defined by:

$$H_d = \Omega \cdot |\mathbf{f}| \cdot |\omega| = \mathbf{f} \cdot \omega = \mathbf{f} \cdot (\nabla \times \mathbf{f}) \tag{3.7}$$

3.5 Interpreting linear dynamical systems

As already stated previously, linear dynamical systems are especially simple to analyze. Since we need this procedure for the rest of this chapter, we briefly discuss some different approaches of analyzing the matrix of a linear and autonomous dynamical system's matrix $\mathbf{A}$ [83].

Eigenvalues and eigenvectors

Continuous dynamical systems ($\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x}$) as well as discrete systems ($\mathbf{x}_{n+1} = \mathbf{A} \cdot \mathbf{x}_n$) that are autonomous and linear can be entirely analyzed by investigating the matrix $\mathbf{A}$ and its characteristics. One possibility is to compute $\mathbf{A}$'s eigenvalues and its eigenvectors from Eqs. 3.8 and 3.9, respectively [72, 83]. The knowledge of $\mathbf{A}$'s eigenvalues and eigenvectors is always sufficient to describe the qualitative behavior of a linear system.

$$\det(\mathbf{A} - \lambda_i \cdot \mathbf{I}) = 0 \tag{3.8}$$

$$\mathbf{A} \cdot \mathbf{e}_i = \lambda_i \cdot \mathbf{e}_i \tag{3.9}$$

Interpreting matrix $\mathbf{A}$ as a (linear) transformation the eigenvectors of $\mathbf{A}$ specify exactly those lines (containing $\mathbf{e}_i$), which are invariant under the transformation. The way such a line itself is transformed is given by the corresponding eigenvalue λ_i.

The interpretation of the eigenvalues λ_i – they can be either real or complex – is different for continuous and discrete dynamical systems, because a continuous system is specified by the change of the current state, whereas a discrete dynamical system is specified by giving the next state of the system (see Tab. 3.1).

Convergence, divergence, and rotation are to be interpreted relatively to the origin of the coordinate system. Note, that a critical point of a continuous dynamical system is called hyperbolic, if its eigenvalues do not lie on the imaginary axis ($\mathrm{Re}\,\lambda_i \neq 0$). Critical points of discrete dynamical systems are hyperbolic, if $|\lambda_i| \neq 1$ for all eigenvalues.

	continuous case	discrete case
convergence	$\mathrm{Re}\,\lambda_i < 0$	$\lvert\lambda_i\rvert < 1$
divergence	$\mathrm{Re}\,\lambda_i > 0$	$\lvert\lambda_i\rvert > 1$
rotation	$\mathrm{Im}\,\lambda_{j,k} \neq 0$	$\mathrm{Im}\,\lambda_{j,k} \neq 0$

Table 3.1: Interpreting the eigenvalues of a linear system.

Decomposing matrix A

Another possibility of analyzing matrix $\mathbf{A}$ of a linear and autonomous system is by decomposing it into a symmetric matrix $\mathbf{A}^+$ and an asymmetric matrix $\mathbf{A}^-$ as follows [19]:

$$\mathbf{A}^+ = \frac{\left(\mathbf{A} + \mathbf{A}^{\mathrm{T}}\right)}{2}, \quad \mathbf{A}^- = \frac{\left(\mathbf{A} - \mathbf{A}^{\mathrm{T}}\right)}{2} \tag{3.10}$$

The elements of $\mathbf{A}^+$ and $\mathbf{A}^-$ can be interpreted rather straightforwardly [75]:

$$\mathbf{A}^+ = \begin{pmatrix} d_x & \cdot & \cdot \\ \cdot & d_y & \cdot \\ \cdot & \cdot & d_z \end{pmatrix}, \quad \text{and } (d_x + d_y + d_z) = \mathrm{div}\,\mathbf{f}(\mathbf{x}) \tag{3.11}$$

The elements of $\mathbf{A}^+$ marked with '$\cdot$' built up the shear strain portion of this linear system.

$$\mathbf{A}^- = \frac{1}{2}\begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix}, \quad \text{and } \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \mathrm{rot}\,\mathbf{f}(\mathbf{x}) \tag{3.12}$$

Analysis in the local coordinate system

A third possibility of linear system analysis is especially useful while investigating a flow's Jacobian $\mathbf{J}$. It can be transformed into the local Frenét-frame at some point of a trajectory ($\mathbf{J} \to \mathbf{J}^{\mathrm{local}}$). Then the elements of $\mathbf{J}^{\mathrm{local}}$ as given in Eq. 3.13 allow a detailed characterization of the underlying flow [19].

$$\mathbf{J}^{\mathrm{local}} = \begin{pmatrix} \hat{a} & \hat{s} & \hat{s} \\ \hat{c} & \hat{d} & \hat{d}, \hat{t} \\ \hat{c} & \hat{d}, \hat{t} & \hat{d} \end{pmatrix} \tag{3.13}$$

$$\hat{a}, \hat{c}, \hat{d}, \hat{s}, \hat{t} \quad \ldots \quad \text{markups of } \mathbf{J}^{\mathrm{local}}\text{'s elements.}$$

Elements of matrix $\mathbf{J}^{\mathrm{local}}$ that are marked with '$\hat{a}$', '$\hat{s}$', or '$\hat{c}$' specify changes of the flow that are parallel to $\mathbf{f}(\mathbf{x})$. The element marked with '$\hat{a}$' gives the acceleration of the flow, whereas the elements marked with '$\hat{s}$' give the shear strain at this state of the system. Elements that are marked with '$\hat{c}$' give the curvature of the flow.

Remaining elements of matrix $\mathbf{J}^{\text{local}}$, that are marked with either '$\hat{d}$' alone or '$\hat{d}$' and '$\hat{t}$', specify the changes of the flow that are perpendicular to $\mathbf{f}(\mathbf{x})$. Splitting the bottom-right 2×2-matrix into a symmetric and an asymmetric one gives the divergence (by the elements marked with '$\hat{d}$') and the torsion (by the elements marked with '$\hat{t}$') of the flow.

3.6 Analysis near critical points or cycles

Linear systems by themselves have a rather simple dynamical behavior. The reason, why linear system analysis is so important, is that non-linear systems are often analyzed by local linearization [83], i.e., a non-linear function is approximated (locally) by a linear function. Linearization is typically done by using a Taylor expansion of a function and neglecting the higher-order terms. See Eq. 3.14 for Taylor expansion of a scalar function $f : \mathbf{R} \to \mathbf{R}$, Eq. 3.15 shows the Taylor expansion for a vector function $\mathbf{f} : \mathbf{R}^n \to \mathbf{R}^n$. This kind of analysis is especially easy near critical points, since the long-term behavior trivially coincides with the local behavior at these points.

$$f(c+d) = \sum_{i\geq 0} \frac{1}{i!}\, d^i \cdot f^{(i)}(c) \approx f(c) + d \cdot f'(c) \tag{3.14}$$

$$\mathbf{f}(\mathbf{c}+\mathbf{d}) = \sum_{i\geq 0} \frac{1}{i!}\, (\mathbf{d}\cdot\nabla)^i\, \mathbf{f}\,\big|_{\mathbf{c}} \approx \mathbf{f}(\mathbf{c}) + \mathbf{d}\cdot\nabla\mathbf{f}\big|_{\mathbf{c}} \tag{3.15}$$

Dynamical system analysis near a critical point

Analyzing the system's behavior near its critical points can help to understand the evolution of any state of the system. Assuming the system is non-linear and hyperbolic, linearization can be used to determine the behavior near critical points completely. Continuous and discrete systems can be treated rather similar [72] (See Tab. 3.2).

To keep the analysis simple, we assume the system to be autonomous and time-independent (see Tab. 3.2(a) for the definitions). Assuming the existence of at least one critical point (see Tab. 3.2(b) for the definitions) any state of the dynamical system near critical point $\mathbf{c}$ can be rewritten with respect to $\mathbf{c}$ (see Tab. 3.2(c)). With this reformulation the dynamical

	continuous case	discrete case	
vector field def.	$\dot{\mathbf{x}} = \mathrm{d}\mathbf{x}/\mathrm{d}t = \mathbf{f}(\mathbf{x})$	$\Delta\mathbf{x}_n = \mathbf{f}(\mathbf{x}_n)$	(a)
critical point def.	$\dot{\mathbf{c}} = \mathbf{f}(\mathbf{c}) = 0$	$\Delta\mathbf{c} = \mathbf{f}(\mathbf{c}) = 0$	(b)
re-writing $\mathbf{x}$, $\mathbf{x}_n$	$\mathbf{x} = \mathbf{c} + \mathbf{d}$	$\mathbf{x}_n = \mathbf{c} + \mathbf{d}_n$	(c)
using Taylor exp.	$\dot{\mathbf{x}} \approx \nabla\mathbf{f}\|_{\mathbf{c}} \cdot \mathbf{d}$	$\Delta\mathbf{x}_n \approx \nabla\mathbf{f}\|_{\mathbf{c}} \cdot \mathbf{d}_n$	(d)
linearized system	$\dot{\mathbf{d}} = \nabla\mathbf{f}\|_{\mathbf{c}} \cdot \mathbf{d}$	$\Delta\mathbf{d}_n = \nabla\mathbf{f}\|_{\mathbf{c}} \cdot \mathbf{d}_n$	(e)

Table 3.2: Local linearization near critical points

system can be approximated by a Taylor expansion as shown in Tab. 3.2(d). $\nabla\mathbf{f}|_{\mathbf{c}}$ denotes the Jacobian matrix of $\mathbf{f}(\mathbf{x})$ evaluated at $\mathbf{c}$. Using Tab. 3.2(c) again, the left side of the Taylor expansion in Tab. 3.2(d) can be rewritten. This operation yields the linearized systems for small perturbations around critical point $\mathbf{c}$ (see Tab. 3.2(e)). These linear systems can now be analyzed as discussed in the last section.

Dynamical system analysis near a cycle

Cycles are another important class of characteristic subsets within continuous dynamical systems. A cycle is given, when the system returns to a previous state. The system behavior near such a cycle can be analyzed by using a Poincaré map. Such a map is a discrete dynamical system, that is produced from a continuous dynamical system and that is of a lower dimension than the original system. A Poincaré map is specified by the cross-section of a surface perpendicular to the cycle (usually a plane) and a trajectory near the cycle. The Poincaré map is a discrete dynamical system with at least one critical point $\mathbf{c}$, i.e., $\mathbf{c}$ is the intersection of the cycle and the surface. Thus the Poincaré map can be analyzed as shown in the section before and the results are then used for interpreting the system's behavior nearby the cycle [72].

3.7 System analysis near trajectories

In the following we propose another approach to analyze a dynamical system's behavior. It is somewhat similar to the method by de Leeuw and van Wijk [19], as the dynamical system is also transformed into the Frenét-Frame Φ of a point on the trajectory. Contrary to their approach we use the analysis by eigenvalues and eigenvectors to interpret this transformed

Jacobian matrix. Expressing a dynamical system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ in terms of Φ one gets

$$\dot{\mathbf{u}} = (\mathbf{g2l} \circ \mathbf{f} \circ \mathbf{l2g})(\mathbf{u}) = \tilde{\mathbf{f}}(\mathbf{u}) \tag{3.16}$$

$\mathbf{u}$ ... a state of the system in terms of Φ

$\mathbf{g2l}$ ... transformation from the global coordinate system into Φ

$\mathbf{l2g}$ ... transformation from Φ into the global coordinate system

Near the point of interest $\mathbf{p}$ (represented in the global coordinate system) a state of the system can be written as $\mathbf{u} = 0 + \mathbf{d}$ in terms of the local coordinate system. Note, that $\mathbf{p}$ represented in terms of Φ is 0. Using a Taylor expansion of $\tilde{\mathbf{f}}(\mathbf{u})$ up to first-order terms, we get

$$\dot{\mathbf{u}} = \tilde{\mathbf{f}}(\mathbf{u}) = \tilde{\mathbf{f}}(0 + \mathbf{d}) \approx \tilde{\mathbf{f}}(0) + \left.\frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{u}}\right|_{\mathbf{u}=0} \cdot \mathbf{d} = \lambda \cdot \phi_1 + \nabla\tilde{\mathbf{f}}\Big|_0 \cdot \mathbf{d} \tag{3.17}$$

ϕ_1 ... unit-vector in terms of Φ, colinear to the axis corresponding to the trajectory's tangent

λ ... length of $\mathbf{f}(\mathbf{p})$

Note, that $\tilde{\mathbf{f}}(0)$ can be written as $\lambda \cdot \phi_1$, since the first axis of the local coordinate system (Frenét-Frame) is specified to be colinear to the flow. Transforming the left-most side of Eq. 3.17 by using $\mathbf{u} = 0 + \mathbf{d}$ we get a linearized system for small perturbations of $\mathbf{p}$ (in terms of Φ), because $\mathrm{d}0/\mathrm{d}t = \tilde{\mathbf{f}}(0) = \lambda \cdot \phi_1$.

$$\dot{\mathbf{u}} = \mathrm{d}(0 + \mathbf{d})/\mathrm{d}t = \dot{0} + \dot{\mathbf{d}} \quad \Longrightarrow \quad \dot{\mathbf{d}} = \nabla\tilde{\mathbf{f}}\Big|_0 \cdot \mathbf{d} \tag{3.18}$$

Now perturbations that are especially useful to analyze are investigated. The elements of $\mathbf{d}$ can be separated into a scalar $\mathbf{d}[1]$ and a vector $\mathbf{d}[2\ldots]$ that is of one dimension less than $\mathbf{d}$. $\mathbf{d}[1]$ is assumed to be 0, since perturbations of $\mathbf{p}$ that are not perpendicular to the trajectory's tangent make no sense at all – a state of the system that is represented as a perturbation of $\mathbf{p}$ with a component $\mathbf{d}[1] \neq 0$ usually can be more accurately expressed as a (perpendicular) perturbation of another point on exactly the same trajectory. Thus $\dot{\mathbf{d}}$ does not depend on the first row of matrix $\nabla\tilde{\mathbf{f}}|_0$. The remaining elements of $\tilde{\mathbf{f}}$'s Jacobian $\nabla\tilde{\mathbf{f}}|_0$ can be decomposed into the first line $\nabla\tilde{\mathbf{f}}|_0[1, 2\ldots]$ and the lower-right sub-matrix $\nabla\tilde{\mathbf{f}}|_0[2\ldots, 2\ldots]$.

Decomposing $\dot{\mathbf{d}}$ similar to $\mathbf{d}$ yields a part parallel to the trajectory's tangent (scalar $\dot{\mathbf{d}}[1]$) and a part perpendicular to a (sub-vector $\dot{\mathbf{d}}[2\ldots]$):

$$\begin{aligned} \dot{\mathbf{d}}[1] &= \nabla\tilde{\mathbf{f}}\Big|_0 [1, 2\ldots] \cdot \mathbf{d}[2\ldots] \\ \dot{\mathbf{d}}[2\ldots] &= \nabla\tilde{\mathbf{f}}\Big|_0 [2\ldots, 2\ldots] \cdot \mathbf{d}[2\ldots] = \mathbf{A} \cdot \mathbf{d}[2\ldots] \end{aligned} \tag{3.19}$$

We now have a locally linearized system ($\dot{\mathbf{d}}[2\ldots] = \mathbf{A} \cdot \mathbf{d}[2\ldots]$) that describes the evolution of variations orthogonal to the flow. The system has one dimension less than the original system. Matrix $\mathbf{A}$ can be analyzed as already shown for continuous systems at the neighborhood of critical points. But we must be careful with the interpretation of this analysis, because all the results hold for the investigated point $\mathbf{p}$ only. For example, if the analysis of matrix $\mathbf{A}$ reveals that the system's evolution is convergent (critical point is an attractor) the only thing that can be said is that nearby trajectories are locally attracted by the trajectory at the specific location chosen. To detect convergent, divergent, or saddle regions of a trajectory it must be shown that the structural characteristics of matrix $\mathbf{A}$ are persistent for a certain region of the trajectory. This might be not simple analytically, but can be done approximately by numerical simulation.

3.8 Discussion

This chapter compiles important terms and definitions that are useful for analyzing analytically defined dynamical systems. Widely varying terms and denotations are sometimes used in literature to describe important concepts of dynamical systems. Thus a clarifying survey of these sometimes interchangeable terms and definitions is given.

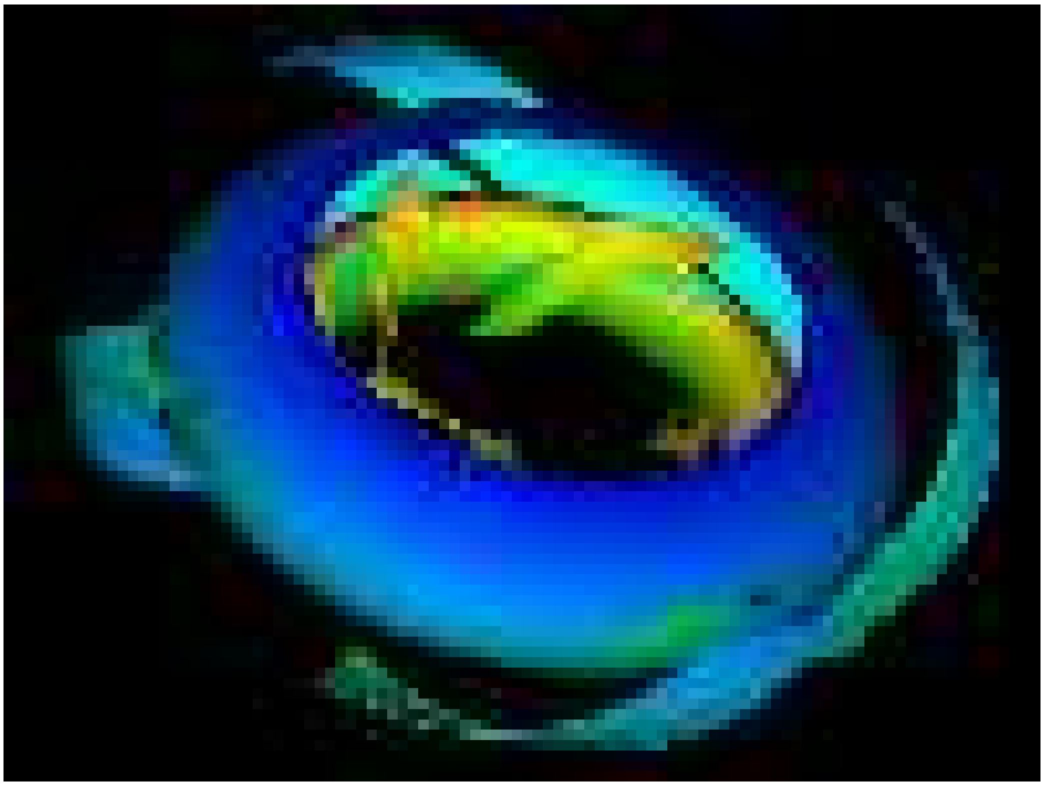
After presenting a classification of dynamical systems, tools of differential geometry are discussed with respect to the analysis of trajectories of dynamical systems. The description of terms defining flow characteristics of dynamical systems (e.g., divergence, rotation) is followed by discussing linearization techniques for dynamical systems.

Together with an investigation of flow behavior close to a critical point and cycles a concept for the local analysis of a dynamical system close to an arbitrary trajectory is presented. This approach basically investigates perturbations orthogonal to the chosen trajectory by determining eigenvalues and eigenvectors of a matrix which is closely related to the Jacobian matrix of the dynamical system but with lower dimension.

Chapter 4

Stream arrows

Τὰ πάντα ῥεῖ (all things are in flux).
Heraclitus (540-480 BC)



As first chapter of this compilation of advanced visualization techniques for dynamical systems, stream arrows [51] and their hierarchical extension [50] are described. The use of stream surfaces [33] for the purpose of directly visualizing system dynamics is extended by several means:

(1) Arrows are added to the stream surface, coding direction and orientation of the flow. (2) Anisotropic spot noise is used to communicate time lines and stream lines simultaneously within the stream surface. (3) Selectively removing parts of the stream surface allows to cope with occlusion and spatial registering. (4) Extensions into 3D are proposed to include further information, for example, the flow near the stream surface or local properties.

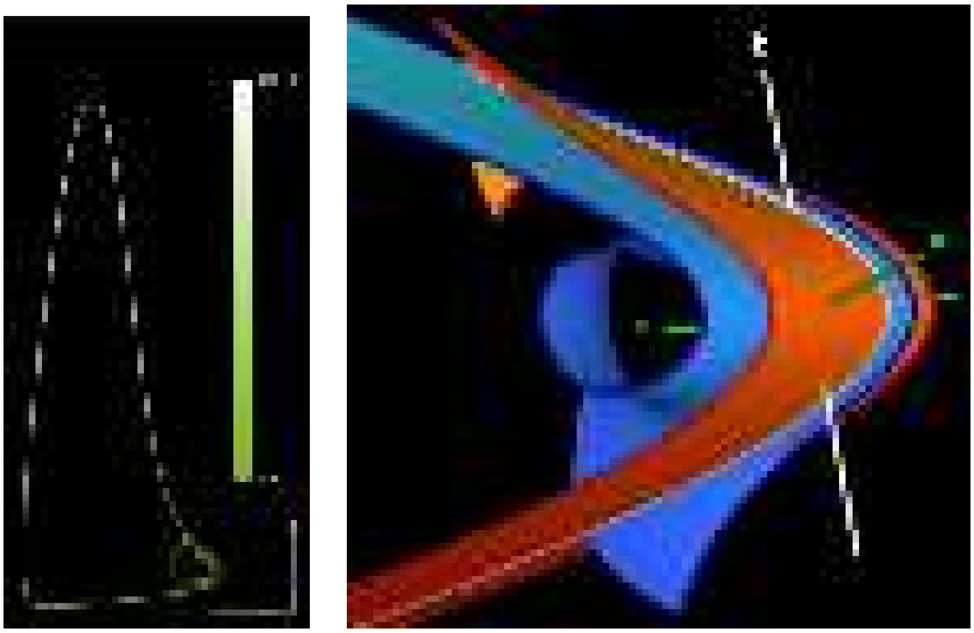
4.1 Introduction

The stream arrows technique was developed during a cooperation with mathematicians who investigated dynamical systems that exhibit mixed-mode oscillations [56, 57]. Specifically, a simplified model, called the three-dimensional autocatalator, which describes the interactions of three chemical entities, is investigated. See Eq. 4.1 for the mathematical description of this three-dimensional dynamical system. See Fig. 4.1(a) for a stream line typical for this dynamical system.

$$
\begin{array}{rcl}
\dot{a} & = & \mu \cdot (\kappa + c) - ab^2 - a \\
\varepsilon \dot{b} & = & ab^2 + a - b \\
\dot{c} & = & b - c \\
a, b, c & \ldots & \text{system variables} \\
\mu, \kappa, \varepsilon & \ldots & \text{system parameters, in our example fixed to } \mu = \\
 & & 0.298,\ \kappa = 2.5,\ \varepsilon = 0.013
\end{array}
\tag{4.1}
$$

Stream surfaces were determined to be a proper representation of the principal dynamics of this dynamical system. However, stream surfaces usually are spatially extensive, and thus *occlusion* often becomes a problem when this visualization technique is used. See Fig. 4.1(b) for two stream surfaces calculated for the same system. By introducing semi-transparent stream arrows and selectively removing parts of the stream surface two improvements are provided (see Sects. 4.2, 4.3, and 4.5).

A major inspiration concerning a solution of how to deal with the problem of occlusion caused by complex shaped (for example, curly) stream surfaces comes from a book on the "Geometry of Behavior" by Abraham and Shaw [1]. The authors discuss various types of three-dimensional dynamical systems



(a) (b)

Figure 4.1: A typical stream line (a) and stream surface (b) calculated for a dynamical system exhibiting mixed-mode oscillations.

by using hand-drawn illustrations which represent the topological structure of their systems. Stream surfaces make up an important part of most of their images. To reduce the negative effects caused by occlusion they use only arrow-shaped parts of a stream surface instead of the whole surface. Additionally they use arrow-shaped holes within stream surfaces to diminish occlusion. Using this approach directional information is also added to the stream surfaces. This enables the viewer to obtain a better feeling for the flow within a stream surface. They also use simple textures in their hand-drawn illustrations to convey a better understanding of the shape of objects in phase space. Refer to Fig. 4.2 for a typical image out of this book.

Another issue of this work is to increase the information provided by a stream surface. Plain stream surfaces, for example, do not represent the direction of flow within the stream surface. Also, neither temporal cues about the integration (time lines) nor information about the flow in the vicinity of a plain stream surface is available.

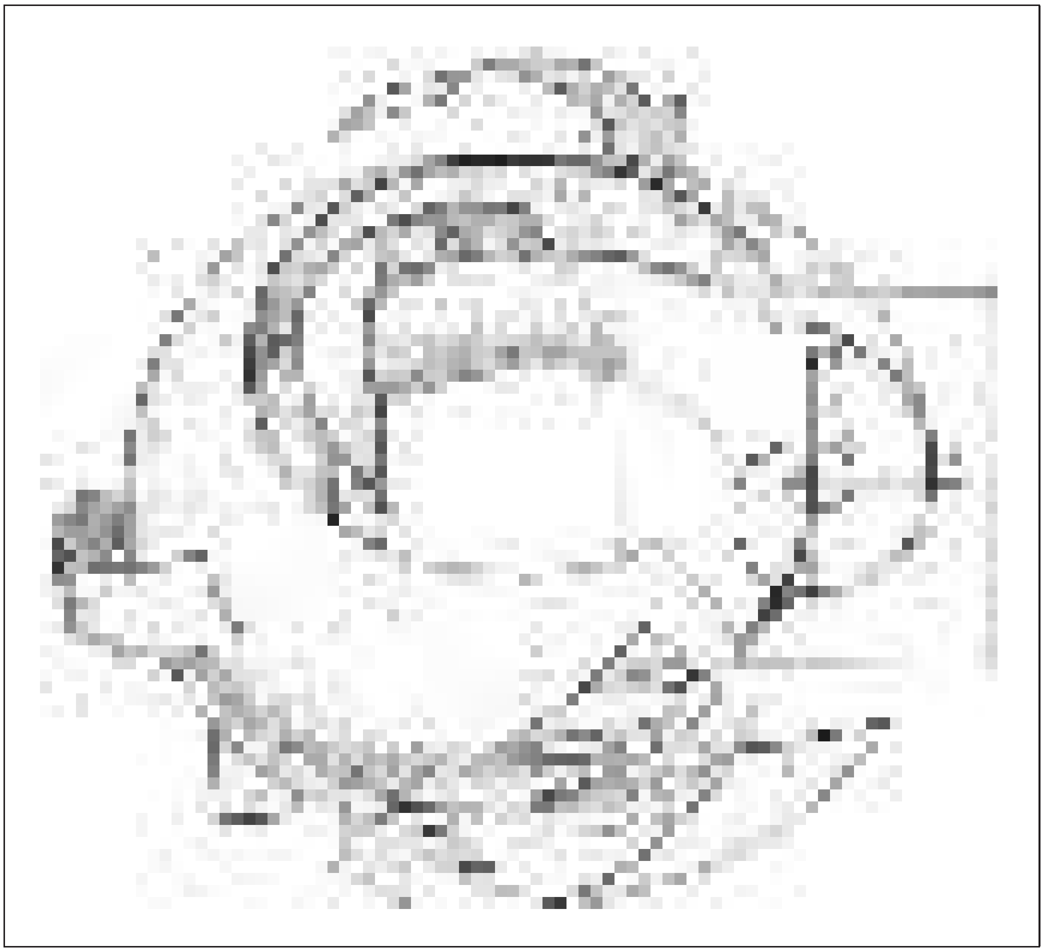


Figure 4.2: Visualization of a dynamical system by using stream arrows [1].

4.2 Stream arrows for stream surfaces

The main extension to the standard stream surface technique is the introduction of stream arrows. A stream surface is thereby segmented into a set of arrow-shaped objects, i.e., the stream arrows, and the remaining surface portion. By assigning a certain level of semi-transparency to the stream arrows the viewer sees through the “holes” in the stream surface and gets more information about the structure of the model. In addition to that, the use of stream arrows allows to visualize local information that is available on the stream surface, e.g., direction of evolution, velocity, and local divergence or convergence (see Fig. 4.3).

The segmentation of the surface into stream arrows and the remaining surface portions is performed by mapping a regularly tiled texture of arrow-

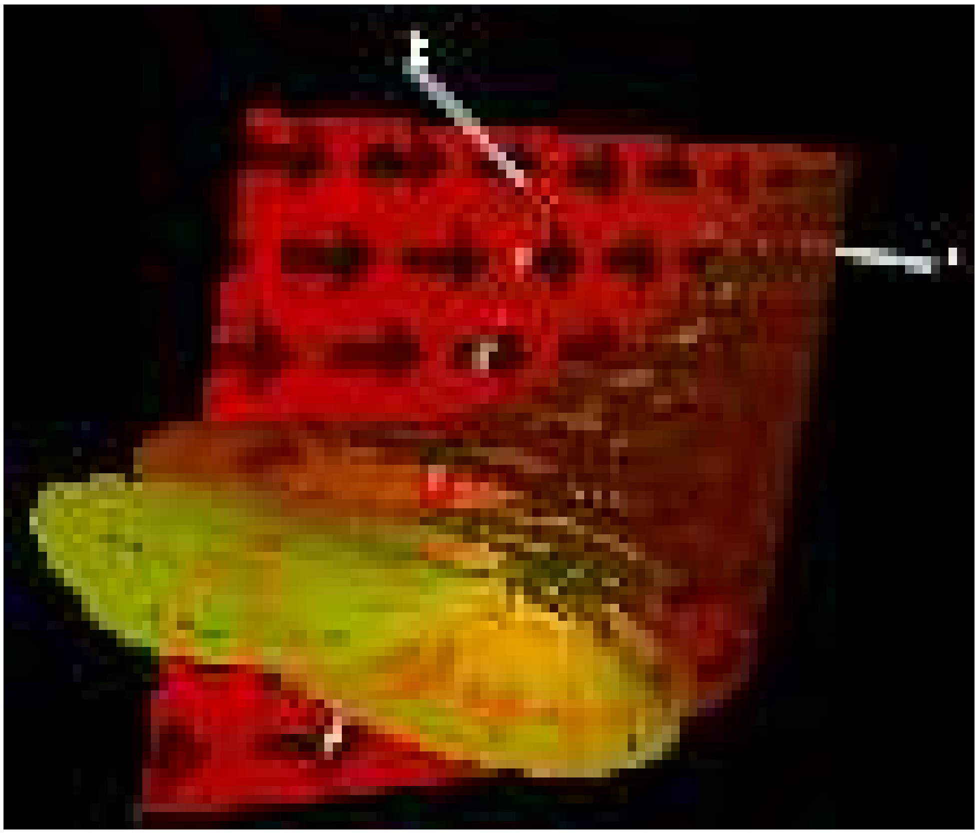


Figure 4.3: Stream arrows for mixed-mode oscillations.

shaped patterns onto the stream surface and cutting its patches along stream arrows borders. The texture is constructed by specifying a base tile, i.e., the shape of one stream arrow, and tessellating the texture using this base tile. See Fig. 4.4 for a comparison of two different shapes of the base tile. Three sets of geometric objects, namely the inside of stream arrows, the outside, and the separating border, are extracted. After this segmentation either the stream arrows or the remaining surface portions can be assigned a certain level of semi-transparency. See Fig. 4.5 for an illustration of this method and Fig. 4.6 for an example, where both possibilities were used.

A segmentation of a stream surface into an entirely opaque portion and highly transparent holes (stream arrows) gives a good impression of the interior structure of a curved stream surface while still retaining a good overview of the spatial arrangement of the stream surface itself. Using homogeneously

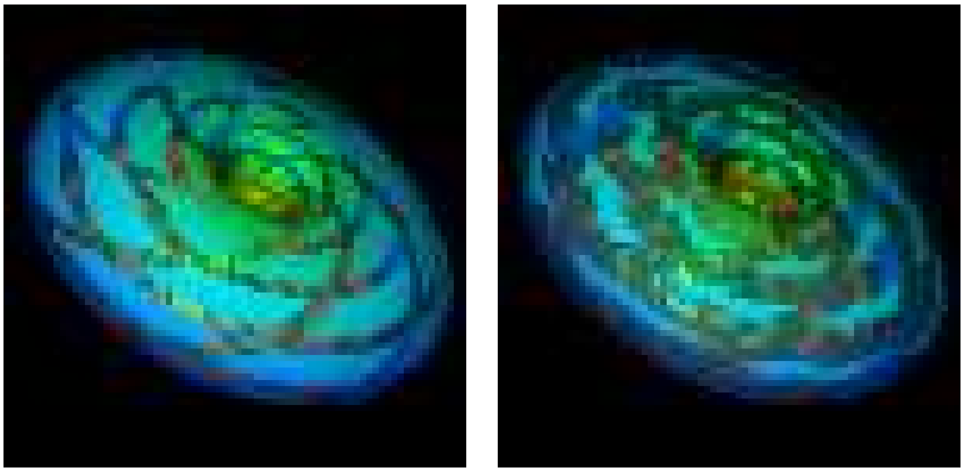


Figure 4.4: Varying the shape of the stream arrows texture.

transparent surfaces would produce several layers of overlapping stream surface segments which are quite difficult to interpret spatially [71].

4.3 Hierarchical stream arrows

The stream arrows approach [51] is based on a regular tiling of texture space. This is not well-suited for stream surfaces that spread over regions of high divergence or convergence. The arrows become either too big or too small in certain areas. To eliminate this undesirable effect we present a hierarchical extension to the stream arrows technique.

A stack of stream arrows textures

The goal is to develop an algorithm, which is capable of generating stream arrows that are almost equal-sized in the rendered image. As we do not want to loose the ability to represent local divergence or convergence using the stream arrows technique, we decided to construct a hierarchical algorithm instead of a continuous solution. When flow diverges locally, the hierarchical stream arrows method switches discretely to the next detailed level of stream arrows. These stream arrows are smaller in texture space, but since they are used for divergent areas of the stream surface, they are finally almost equal-

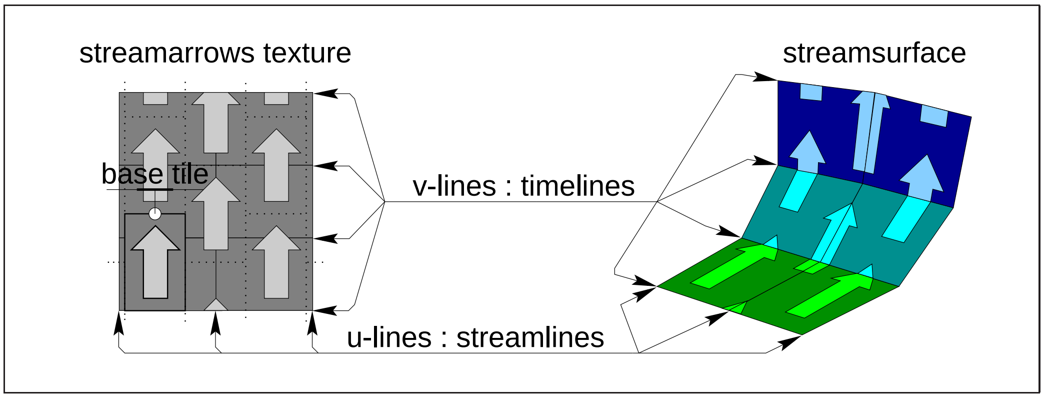


Figure 4.5: Mapping the stream arrows texture to a stream surface.

sized to all other stream arrows in the rendered image. Refer to Fig. 4.7 for an image which demonstrates this situation.

The *hierarchical stream arrows texture* is specified by the shape of one tile, i.e., the outline of an arrow, and two vectors **dc** and **dr** (see Fig. 4.8) which define the offsets between adjacent columns and rows of arrows, respectively. Additionally there is a factor a which represents the scale relation between level i and $i+1$. If $a=1/2$ the size of stream arrows is doubled, when the algorithm switches to the next coarser level. Finally there is a vector **o**, which is the offset of the entire texture with respect to the origin of texture space. Offset vector **o** becomes important, when animation is applied. Due to this specification each stream arrow can be addressed by exactly one identity ID given by three numbers. ID ($level$, col, row) identifies one stream arrow as a copy of the base tile, first translated by $\mathbf{o} + col \cdot \mathbf{dc} + row \cdot \mathbf{dr}$, and then scaled about the origin by a^{level}.

The separation algorithm

The hierarchical stream arrows technique is triangle oriented, because we produce the stream surfaces as triangular meshes: the front of the stream surface ($\approx$ time line) is advanced through phase space while the stream surface is integrated. Triangles are smaller in stream surface areas, where curvature is high, flat areas of stream surfaces are triangulated with larger triangles.

During the stream surface algorithm vertices are assigned 2D texture coordinates. One texture coordinate of each vertex (v coordinate) is set to the integration time of a stream line from the seed point to the vertex. The other texture coordinate of each vertex is set in such a way that all vertices

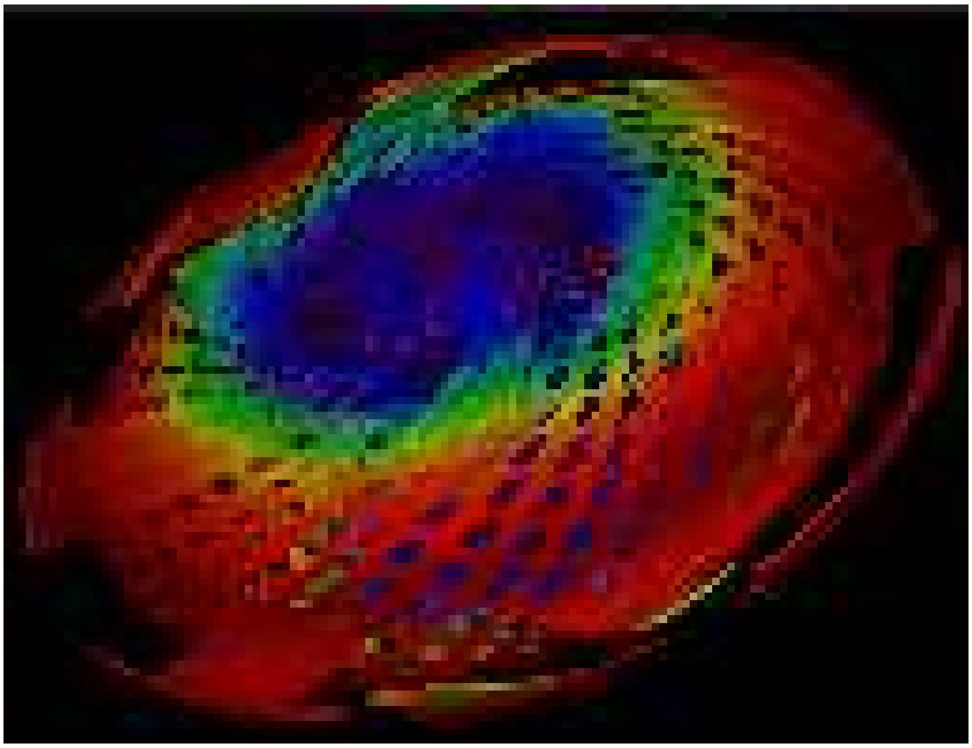


Figure 4.6: Using semi-transparency either for the arrows or the remaining stream surface portions.

connected with one stream line get the same texture value, i.e., the 1D seed parameter of the start-point of the stream line (u coordinate). See Fig. 4.5.

To apply the hierarchical stream arrows texture to the stream surface the hierarchical stream arrows algorithm processes the stream surface triangle by triangle and performs the following separation algorithm:

```
activeTiles = {}                  // . . .  IDs of active tiles
lockedTiles = {}                  // . . .  IDs of locked tiles
FOR ALL Triangles tri DO:
| level:=findLevelOfTriangle(tri) // get most appropriate level
| tiles:=getMaybeTiles(tri,level) // . .  get overlapping tiles
| FOR ALL Tiles tile IN tiles DO:
| | IF NOT (tile.active OR tile.locked) THEN:
| | | IF overlap(tile,activeTiles) THEN: tile.lock
| | | ELSE:                              tile.activate
| intersect(tri,activeTiles)      // . . . .  do the separation
```

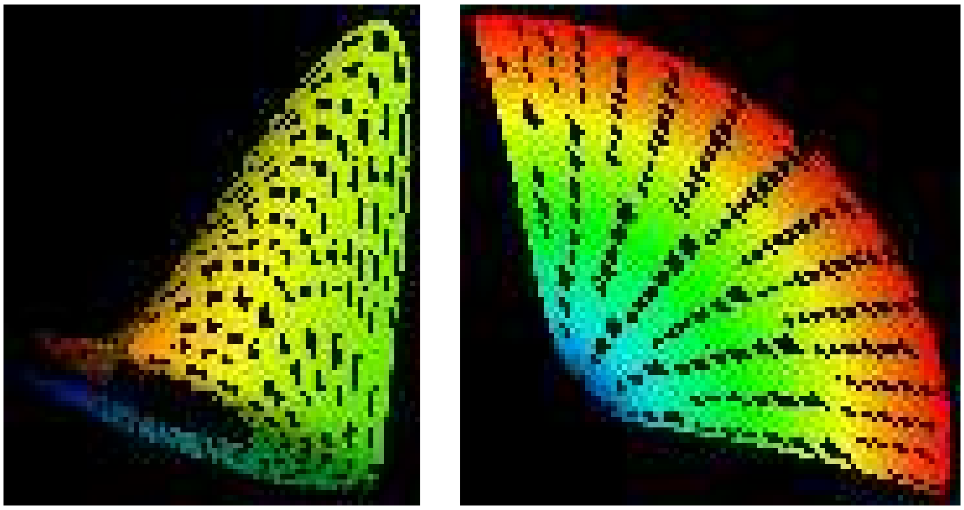


Figure 4.7: Hierarchical stream arrows, two examples.

The algorithm needs two data structures in addition to the triangular mesh of the stream surface: `activeTiles` stores the IDs of all tiles that actually are instanced within the stream surface, whereas in `lockedTiles` all IDs of tiles are stored that overlap at least one active tile and thus should not be generated. Both data structures need to be be searched as fast as possible (in `overlap()` and `intersect()`) and easy to extend by a new tile. Since tiles are related via their 2D location in texture space and searching is also performed in a 'geographical' manner using texture coordinates (IDs), we use a linked data structure that closely represents the spatial relation between tiles.

The algorithm processes the stream surface triangle by triangle. For each triangle `tri` the corresponding `level` in the hierarchical stream arrows texture is determined by comparing the size of triangle `tri` in texture space to its size in phase space coordinates. This ratio is used to find the most appropriate level in the stack of stream arrows textures. Then all tiles in that level, which might intersect triangle `tri` are determined (`getMaybeTiles()`). Tiles which are already activated or locked are omitted. All remaining tiles are checked, whether they overlap any active tile (`overlap()`). Tiles that overlap at least one active tile are locked (added to `lockedTiles`), and all the others are activated (added to `activeTiles`). See Fig. 4.9 for an example with two triangles. After all tiles are checked triangle `tri` is intersected with all active tiles and separated into three sets: parts that belong to the

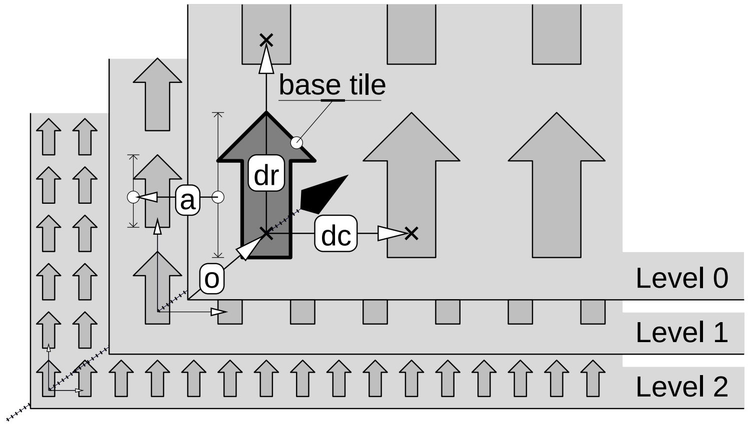


Figure 4.8: Hierarchical stream arrows texture, i.e., a stack of stream arrows textures – specification parameters.

arrows, parts that do not, and the separating outline. All three sets can be individually processed, e.g., assigned a certain level of semi-transparency, after the segmentation algorithm has finished.

The mechanism of `activeTiles` and `lockedTiles` ensures that neighboring triangles of a stream surface are consistently covered by entire tiles of the hierarchical stream arrows texture. If the tile instanced for the currently processed triangle is overlapped by a tile which was already instanced for a previously processed and nearby triangle—this tile is active and of different level than the current tile—the active tile is used for the current triangle. This ensures consistency. The current tile which overlaps the active tile is furthermore locked.

4.4 Anisotropic spot noise

Another part of the proposed approach is the use of spot noise as an additional stream surface texture. Spot noise (see Fig. 4.10) has been introduced to the computer graphics community by Jarke van Wijk in 1991 [87]. It is a powerful technique to generate various random textures that are suitable for different purposes. By constructing a spot noise texture, which emphasizes stream lines and time lines on a stream surface, the visualization of flow using stream surfaces can be improved. See Fig. 4.11 for a typical image. To find an appropriate spot is facilitated by the fact that spot noise

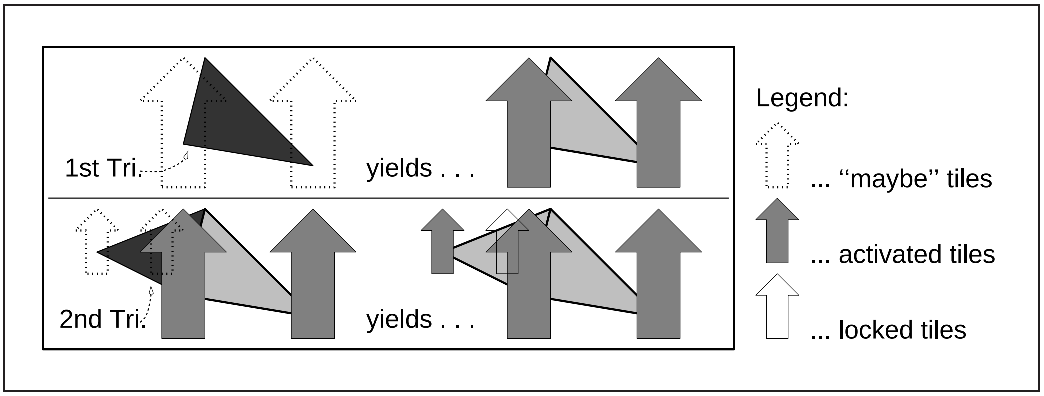


Figure 4.9: Activating and locking tiles, example with two triangles.

textures directly reflect the geometric characteristics of the spot used for its construction. As we want to use the texture to emphasize stream lines and time lines, we parameterize the stream surface such that parameter lines directly correspond to stream lines and time lines. This parameterization is easily achieved, since the construction of stream surfaces is based on stream line integration. Using this type of stream surface parameterization a spot emphasizing horizontal and vertical directions in texture space, e.g., a cross or a hash, should be used. In Fig. 4.10 the spot we use and the resulting spot noise texture is shown. Fig. 4.11 shows a textured stream surface of the mixed-mode oscillations model.

We also thought of using line integral convolution [14] as an alternative to spot noise, but mainly two reasons induced us to use spot noise instead. First, the use of spot noise allows to emphasize both stream lines *and* time lines simultaneously. Using line integral convolution on the other hand, just stream lines *or* time lines could be emphasized. Another reason for the use of spot noise is, that it is less costly than line integral convolution.

Combining spot noise texture and stream arrows produces an expressive stream surface visualization technique. While the use of the stream arrows texture might raise a problem at ill-behaved areas of the stream surface, e.g., regions of large divergence or convergence, spot noise textures suffer from less problems in this situations.

4.5 Selective cuts

The use of 2D cross-sections is a well-established method to reduce the problem of occlusion in 3D. A similar effect can be achieved, when parts of the

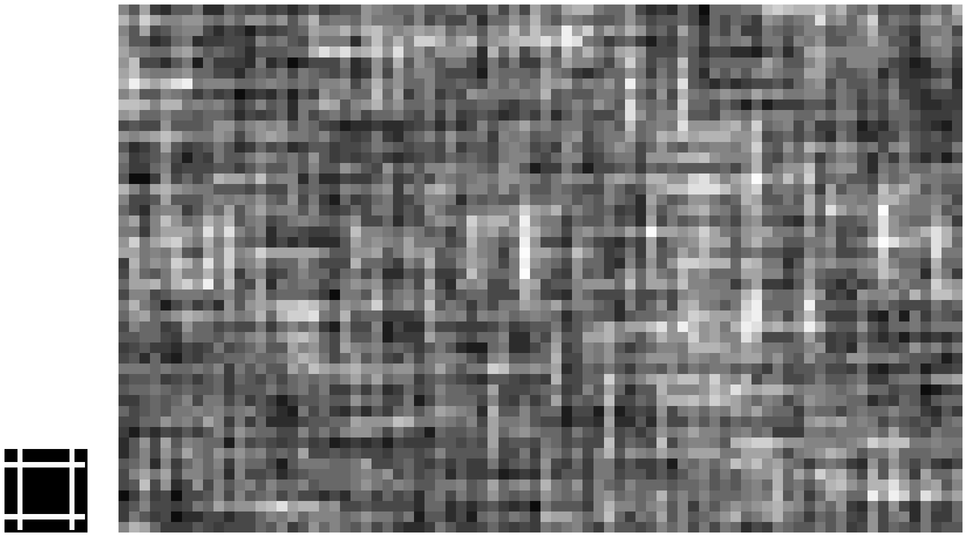


Figure 4.10: spot (enlarged) and the resulting spot noise texture.

model are cut away and removed. Simple geometric objects, e.g., planes, are used to cut through a 3D model.

We use this method for the stream surface representation of the given dynamical system, but do not remove surface parts entirely. Specifying a certain cut plane the stream surface is separated into three parts, i.e., components of the stream surface behind, on, and in front of the cut plane. The geometrical separation of the stream surface allows to specify different visibility parameters for all three parts. Components in front of the cut plane may be rendered semi-transparently. The intersection curves of the stream surface with the cut plane may be emphasized by representing them as tubes. Several arbitrary cut planes may be applied simultaneously so that, for example, a wedge-shaped part of the stream surface is extracted. Fig. 4.11 gives an example, where parts in front of the plane are rendered semi-transparently. Enhancing some of the stream lines, for example, the edges of the stream surface, facilitates the perception of the spatial arrangement of the semi-transparent portions.

The location and orientation of objects, that are used for model separation and components removal, are crucial for the benefit of this method. These parameters can be determined either automatically or interactively by the user. An automatic approach would mean to search for an optimal position and orientation of the cut such that the most important parts of the visual

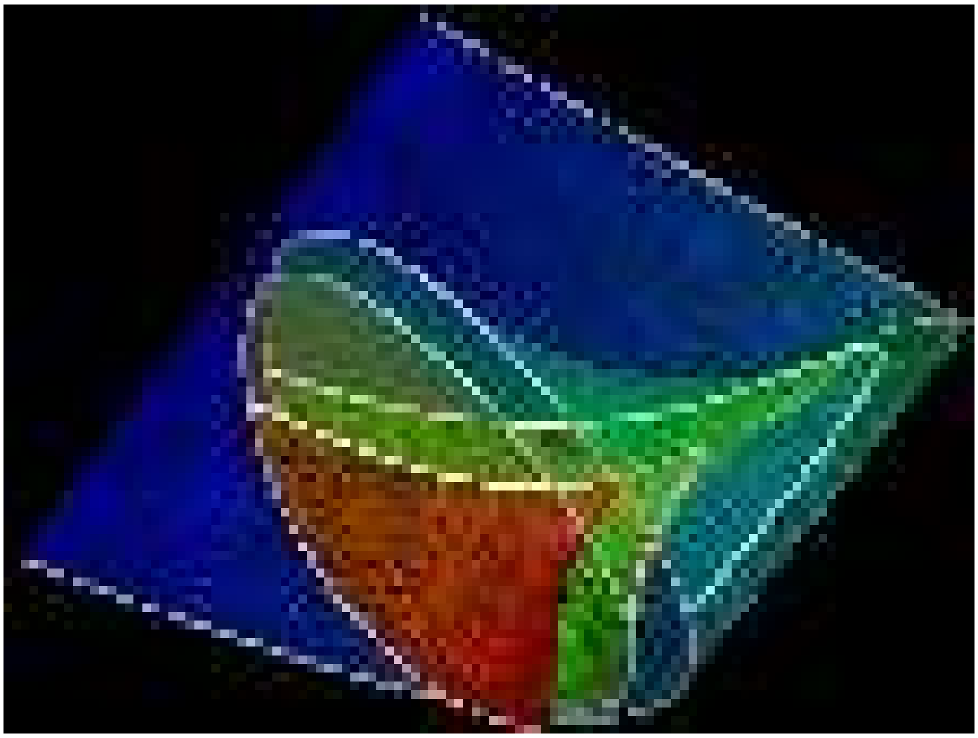


Figure 4.11: Stream surface with anisotropic spot noise texture.

representation are kept and less important parts are removed. Obviously an automated approach is easily extended to support the calculation of animation sequences. On the other hand it is quite difficult to meet the user's requirements automatically, when studying the system representation. In the case of this thesis an automated 'intelligent' approach for the placement of cut planes was not implemented. Animation sequences were generated by specifying simple movements of the cut plane. The cut plane can, for example, move along one coordinte axis while remaining orthogonal to this axis during the entire animation. See Fig. 4.12 for two snapshots out of an animation sequence.

4.6 Animation aspects

As fourth part of stream surface enhancement we apply animation to certain parts of the visual representation. As stream arrows are implemented in the scope of the DynSys3D system (see Chapter 8), which itself is based

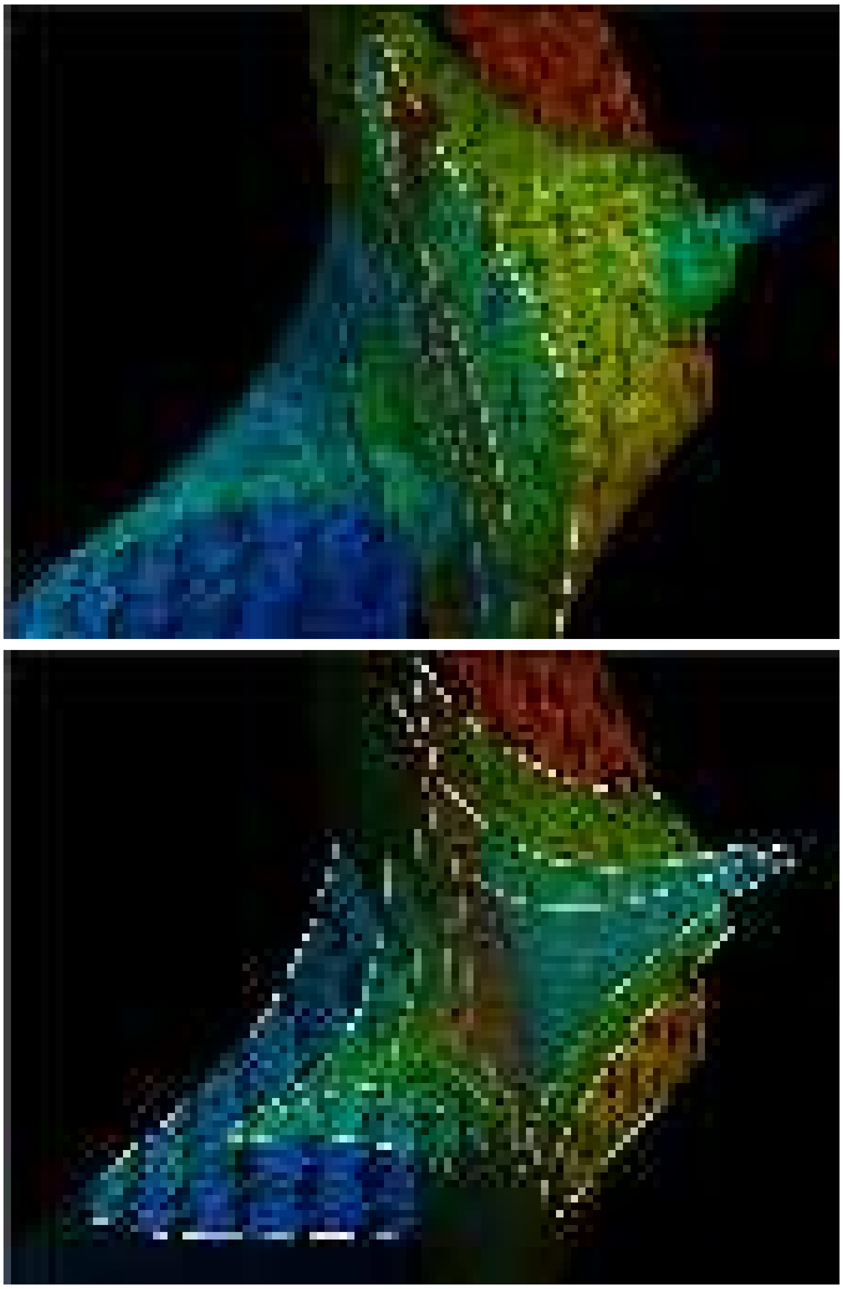


Figure 4.12: Two snapshots from an animation sequence – the first from the beginning and the second from the end of the animation sequence.

on AVS [2], animation is well-supported. AVS allows to export certain module parameters as input ports. Furthermore there are already some basic animation modules available as, e.g., *animated integer* or *animated float*. Connecting such a module to an exported module parameter is the simplest method of animating a visualization setup.

Several parameters can be animated to improve the visualization. An animated offset vector **o** (see Sect. 4.3 and Fig. 4.8) is used to move stream arrows along stream lines within the stream surface. Composing the tiles of the texture in a regular manner, just a few images of the animation sequence have to be rendered. A longer sequence can be produced by looping the cyclic sequence several times.

Other sequences are rendered by animating the construction of a stream surface. The temporal evolution of a stream surface starting from an initial set of colinear points is illustrated. Taking the stream surface of the mixed-mode oscillations model to be opaque clearly shows how some parts of the model representation severely occlude other portions. With stream arrows this unsatisfactory situation is improved. The animation of the stream surface evolution gives a good impression of the dynamics induced by the given dynamical system.

The removal of certain parts of the model can be animated as well. A simulated process of successively removing more and more parts of the stream surface is easier to interpret than still images where parts of the model have been removed. Another animation sequence we rendered shows such a process—a cut plane is used to distinguish semi-transparent parts of the stream surface above the plane from opaque areas below. During the animation the plane is moved towards the center of the model, so that successively most parts of the stream surface become visible.

Many other parameters are suitable to be animated. Moving the viewpoint around the model does certainly help to understand the system behavior. The initial curve of the constructed stream surface may be moved to demonstrate stability features of the dynamical system. This corresponds to successively displaying not only one but various adjacent stream surfaces. The stream lines used to enhance the edges of the stream surface could be animated by moving their origins along the line of initial conditions. This would improve the visualization of the flow within a stream surface.

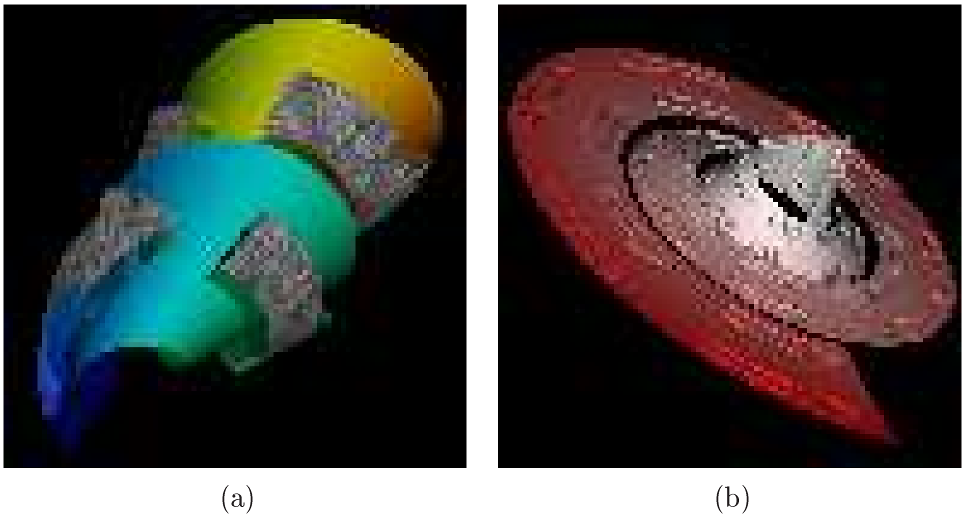


Figure 4.13: (a) Stream arrows shifted out of the stream surface plus anisotropic spot-noise. (b) Stream arrows outlines represented as 3D tubes and color coding of velocity magnitude.

4.7 Additional extensions

The stream surfaces technique is used with color coding local attributes, e.g., velocity magnitude, integration time, divergence, or helicity (see also Chapt. 3). This extension allows to focus the viewer's attention on stream surface regions that exhibit certain values of a local parameter.

After the geometric separation of stream arrows has been performed two possibilities of extending stream arrows into 3D have been realized. One possibility is to shift the separated stream arrows slightly in a direction perpendicular to the remaining stream surface portions. To avoid any confusion – one could interpret the stream arrows as local solutions of the dynamical system, which is certainly not the case in most situations – the shifted stream arrows are connected with the remaining parts of the stream surface by semi-transparent patches. Generating 3D arrows using this technique improves the perception of spatial location and orientation of the stream surface (see Fig. 4.13(a)). In this image stream arrows are shown with anisotropic spot noise [87] illustrating stream lines and time lines.

Although this method of shifting stream arrows slightly out of the stream surface often generates useful results, it has its disadvantages in other cases.

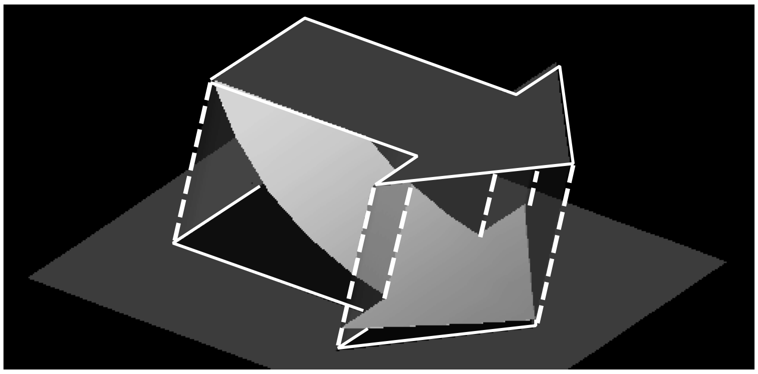


Figure 4.14: Misleading shifted stream arrows (bright: integrated stream arrow).

Especially if nearby stream surfaces are far from being coplanar to the enhanced stream surface – a stream surface as a stable system solution can be imagined as such a case (see Fig. 4.14) – this technique might give a wrong impression.

Therefore, another extension of the underlying dynamical system allows to build up the shifted stream arrow as a local solution. This is achieved by starting a new stream surface at the shifted tail of the stream arrow and cutting the arrow out of this stream surface. This technique will allow to represent the dynamical system not just at the stream surface, but also in its vicinity. Other shapes of 3D stream arrows, e.g., a tent-like shape or variations of the 2D base shape, also have been investigated.

Another 3D extension is the representation of the separating outline as a 3D tube (see Fig. 4.13(b)). Using this approach stream arrows can be realized without removing any part of the stream surface. Almost all of the advantages of the stream arrows technique are preserved as, e.g., indication of flow direction and local velocity.

Further extensions to the stream arrows technique might be 3D stream arrows which can be seen as glyphs with a set of free parameters. Scalar data such as velocity and helicity as well as vector data, for example, vorticity, may be mapped to a stream arrows parameter. Not only 3D shape attributes of stream arrows could be interpreted as glyph parameters, but also size, 2D shape, opacity, and color.

Related to the hierarchical stream arrows texture we worked on other extensions concerning the placement of stream arrows. One idea is to randomly position stream arrows on stream surfaces. Another idea is to use local stream surface attributes for the location of stream arrows. Surface curvature could be used for this purpose. This has already been shown to be useful [34]. The opacity of stream arrows could modulated according to the curvature of the stream surface such that rather opaque arrows are place in regions of high surface curvature. Finally we think of including viewing parameters to determine optimal placements of stream arrows. This idea was again inspired by the book of Abraham and Shaw [1], where these parameters are included in the hand-drawn illustrations as well. Location as well as length and width of arrows could be chosen in a way that occlusion of details behind is diminuished.

4.8 Discussion

Stream surfaces are a useful technique to visualize three-dimensional flow data. For an entire continuous set of initial conditions the temporal evolution is depicted. Using stream arrows, some of the disadvantages related to stream surfaces are omitted or diminished: by texturing the surface according to the flow of the 3D dynamical system, flow direction and velocity is visualized even for interior points of the stream surface. Less occlusion is achieved, using selective semi-transparency. Combining stream arrows with anisotropic spot noise and selective cuts yields expressive images of three-dimensional flows.

One difficulty with stream arrows is the number of parameters. To generate useful images in a certain situation it is necessary to tune the stream arrows parameters, for instance, scaling factor a for hierarchical stream arrows. To come up with a completely automatic parameter set-up, seems to be a non-trivial problem. Another problem with stream arrows is the number of triangles being handled, when high-quality images are to be generated. Geometric segmentation of stream arrows usually causes many triangles of the original stream surface mesh to be split in even more polygons. Thus, the use of stream arrows within an interactive set-up (e.g., a virtual/augmented environment set-up [26]) might cause performance problems.

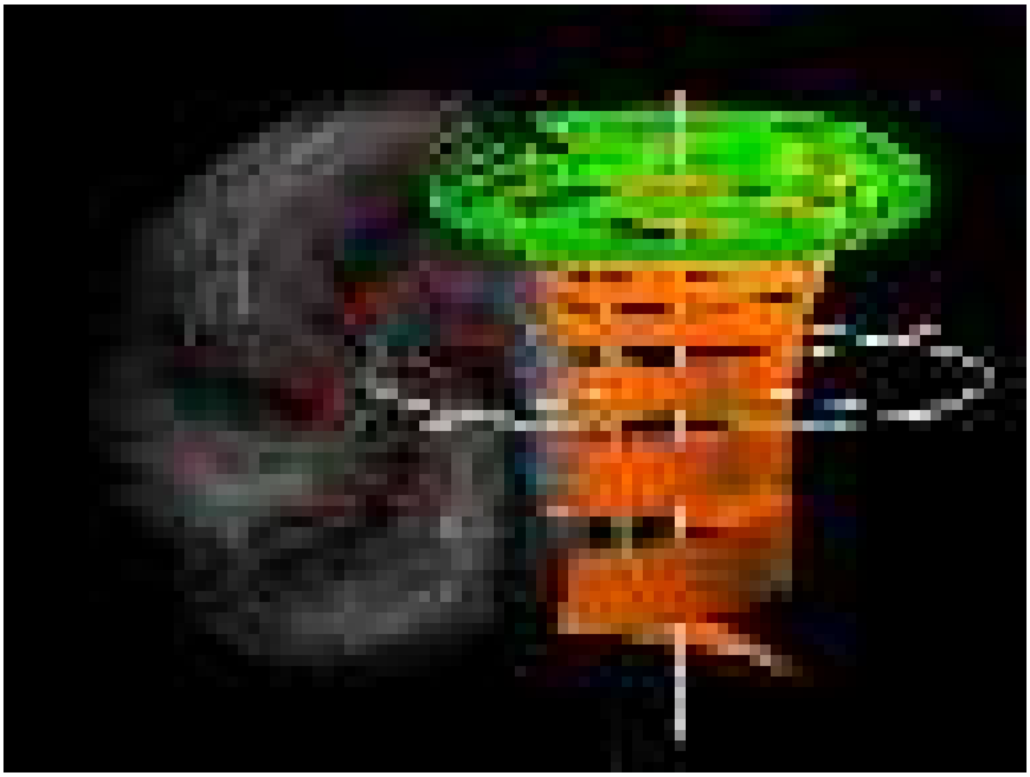
It is useful to sub-structure stream surfaces by the use of texture that is generated with respect to the underlying flow. Important local information, e.g., flow direction and velocity, is additionally integrated in the visualization. Stream arrows are quite useful to generate high-quality representations of three-dimensional dynamical systems or flow data. The cover image of this chapter (see page 36) shows a visualization, which is quite similar to one of the expressive images presented by Abraham and Shaw [1] (see Fig. 4.2).

Chapter 5

Poincaré maps and visualization

A man travels the world over in search of what he needs and returns home to find it.

George Moore (1852-1933)



This chapter presents a method which is based on Poincaré maps and Poincaré sections [49]. Visualization is enabled to more directly show the crucial aspects of system dynamics by using a mathematical tool to distill essential information from dynamical systems that are dominated by a cyclic or quasi-cyclic behavior, i.e., the derivation of a discrete Poincaré map. Cycle characteristics, for example, are especially well-suited for being visualized using this approach.

5.1 Introduction

Poincaré sections are an important tool for the investigation of dynamical systems in theory as well as in applications. They are used for models (usually in 3D) that exhibit periodic or quasi-periodic behavior. In addition to mathematical descriptions—there is a lot of theory about dynamical systems [4, 5, 32, 92]—periodic or quasi-periodic dynamical systems can be found in many fields, e.g., in physics, chemistry, biology, ecology. Especially chaotic systems are often examined by the use of Poincaré sections [80, 83, 92].

A 2D Poincaré section through a periodic 3D flow is a planar cross-section transverse to the flow such that a periodic orbit intersects it at its center. The corresponding Poincaré map is defined as a map correlating consecutive intersections of flow trajectories with the Poincaré section. The Poincaré map is a discrete dynamical system of one dimension less than the continuous flow which it is constructed from. As many of the most important flow properties are inherited by the Poincaré map and its analysis is usually more simple due to its reduced dimensionality, it is often used for analysis instead of the 3D flow. See Sect. 5.2 for a more detailed discussion of some basics on Poincaré sections and Poincaré maps.

Since their introduction to dynamical system analysis by Henri Poincaré in 1899 [66] the visual representation of Poincaré maps has been always a very important part of this research technique. Hand-drawn sketches of the Poincaré map were used for a long time to guide or illustrate the mathematical analysis [1]. Due to the ability of integrating a dynamical system numerically by the use of a computer, visualization techniques that are based on the numerical approximation of the Poincaré map have become popular [63, 83]. There are a number of programs that calculate Poincaré maps [22]. See Sect. 5.3 for a brief review of previous work in this field.

Many visualization techniques for Poincaré maps suffer from rather severe limitations. One problem is, that with the use of 2D visualization techniques

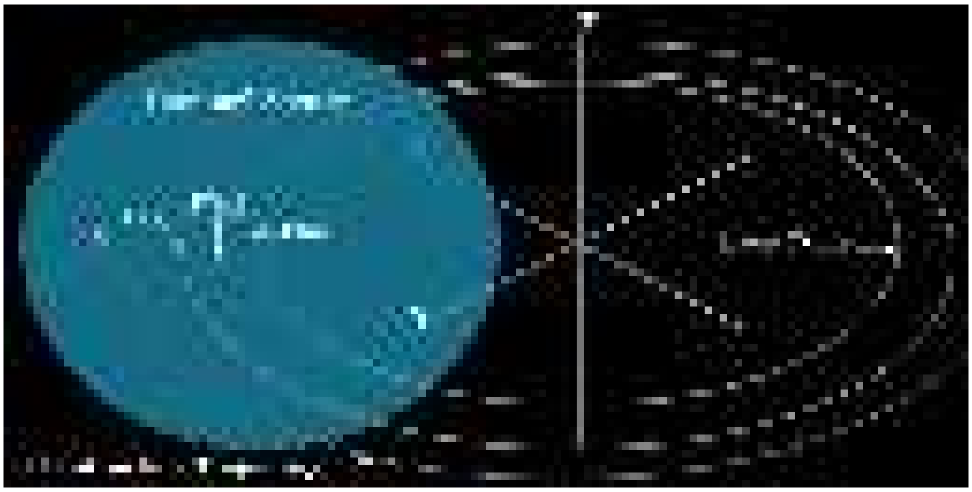


Figure 5.1: An illustration of the Poincaré map definition.

the context of the 3D flow is lost. Certain features, e.g., the number of windings of a Möbius band, can not be derived from the 2D Poincaré map alone. Another problem with these techniques is that the temporal correlation between points of the Poincaré map is not encoded within the 2D image. Most limitations of the 2D techniques in this field are usually not due to a weakness of the software or method, but rather due to an inherent difficulty with dimension reduction approaches.

We therefore propose a set of advances within this rather untouched field of visualization. First, we suggest to adapt some well-known visualization techniques as, e.g., spot noise, to Poincaré maps to improve the visual representation of the 2D map. See Sect. 5.3 for a discussion of these ideas. Furthermore we present an embedding of these techniques within a 3D visualization of the underlying flow. This approach allows to significantly reduce some limitations of previously known techniques. Refer to Sect. 5.6 for a description of this approach.

5.2 About Poincaré maps

A *Poincaré section* is used to construct a $(n-1)$-dimensional discrete dynamical system, i.e., a *Poincaré map*, of a continuous flow given in n dimensions. This reduced system of $n-1$ dimensions inherits many properties,

(a) (b)

Figure 5.2: (a) An example of a traditional Poincaré map visualization [76]. (b) An example of a 3D Poincaré map [15].

e.g., periodicity or quasi-periodicity, of the original system. We will concentrate in the following on the case of n being equal to three.

Poincaré maps are used to investigate periodic or quasi-periodic dynamical systems. Often these systems exhibit a periodic cycle or a chaotic attractor. A Poincaré section $\mathcal{S}$ is now assumed to be a part of a plane, which is placed within the 3D phase space of the continuous dynamical system such that either the periodic orbit or the chaotic attractor intersects the Poincaré section. The Poincaré map is now defined as a discrete function $\mathbf{p} : \mathcal{S} \to \mathcal{S}$, which associates consecutive intersections of a trajectory of the 3D flow with $\mathcal{S}$ (see Fig. 5.1).

There are some important relations between a 3D flow and the corresponding Poincaré map: A cycle $\mathcal{C}$ of the 3D system which intersects the Poincaré section $\mathcal{S}$ in q points ($q \geq 1$) is related to a periodic point $\mathbf{c} = \mathcal{C} \cap \mathcal{S} = \mathbf{p}^q(\mathbf{c})$ of Poincaré map $\mathbf{p}$, i.e., $\mathbf{c}$ is a critical point of the map $\mathbf{p}^q$. Furthermore stability characteristics of the cycle are inherited by the critical point: stable, unstable, or saddle cycles result in stable, unstable, or saddle nodes, respectively. Therefore many characteristics of periodic or quasi-periodic dynamical systems can be derived from the corresponding Poincaré map.

5.3 Previous and related work

Visualization techniques have been used for the illustration of Poincaré maps since they were introduced by Henry Poincaré. Most of these imaging methods are 2D or 1D plots that are calculated by numerically integrating the underlying flow. See Fig. 5.2(a) for an example of such a technique.

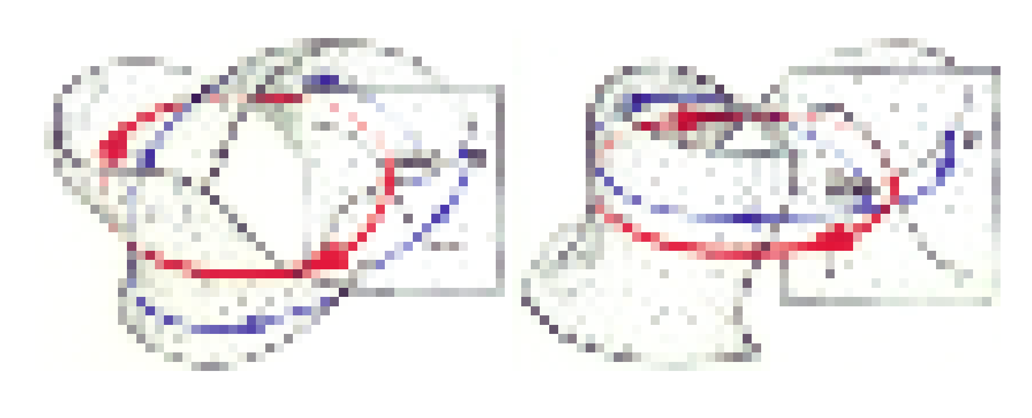


Figure 5.3: Poincaré map visualization by Abraham and Shaw [1].

Rarely Poincaré maps in 3D or even higher dimensions are investigated. See Fig. 5.2(b) for an example.

In addition to these traditional 2D plots a few more general techniques can be found in the literature. Hand-drawn images in the book by Abraham and Shaw [1] demonstrate that a combination of the Poincaré section and the underlying 3D flow within a single image convey a better understanding of the underlying flow characteristics (see Fig. 5.3). The book by Abraham and Shaw is in effect quite an inspirational source. We think that a number of artistic techniques used for the hand-drawn images in the book are well-suited for computer-supported visualization techniques.

5.4 Visualizing Poincaré map p

Since Poincaré map $\mathbf{p}$ maps $\mathbf{x}$ onto $\mathbf{p}(\mathbf{x})$, both lying in $\mathcal{S}$, a visualization based on a directed stroke connecting $\mathbf{x}$ and $\mathbf{p}(\mathbf{x})$ has been implemented in AVS (see Chapt. 8). A module named FLOW generates a set of arrows on the Poincaré section, which start at a point $\mathbf{x}_i$ and end in a correlated point $\mathbf{p}(\mathbf{x}_i)$. See Fig. 5.4 for a visualization of a non-linear saddle cycle where this technique was used. $\mathcal{S}$ is represented as a semi-transparent disk. A set of light-grey arrows is placed on $\mathcal{S}$ to visualize $\mathbf{p}^1$. Sequences of consecutive applications of $\mathbf{p}$ are visualized by the use of small red spheres, representing $\{\mathbf{p}^j(\mathbf{x}) \mid j \geq 0\}$, whereby the sphere depicting $\mathbf{x}=\mathbf{p}^0(\mathbf{x})$ is colored differently from the others. The visualization of the sequence $\{\mathbf{p}^j(\mathbf{x}_0) \mid j \geq 0\}$ with $\mathbf{x}_0$ close to the origin of phase space is combined with a visualization of the constructing flow trajectory.

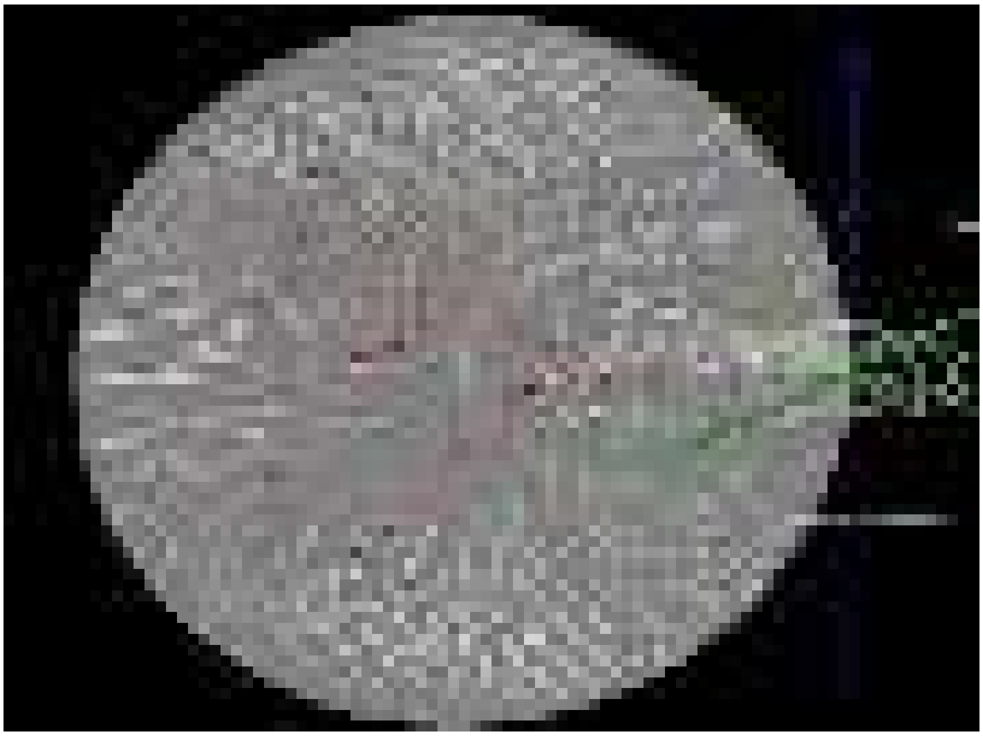


Figure 5.4: Visualizing a non-linear saddle cycle.

We also adapted spot noise [87] to Poincaré maps. We place elliptic spots onto $\mathcal{S}$ such that the focal points of the ellipses coincide with $\mathbf{x}_i$ and $\mathbf{p}(\mathbf{x}_i)$, respectively. See Fig. 5.5 for an example. This choice is due to the fact that no directional information should be encoded, when $\mathbf{p}(\mathbf{x}_i)=\mathbf{x}_i$. In this case both focal points coincide and the elliptic spot degenerates to a circular spot. Images rendered with this method are well suited to visualize the entirety of $\mathbf{p}(\mathcal{S})$ within one still image. See Fig. 5.6 for a visualization of a non-hyperbolic saddle cycle (3 stable and 3 unstable manifolds) where spot noise was used for visualization. Similar as in Fig. 5.4, six sequences $\{\mathbf{p}^j(\mathbf{x}_i) \mid j \geq 0\}_{i\in\{1,2,\ldots,6\}}$ are visualized by the use of white and red spheres.

The results of the previous techniques are now embedded into a 3D visualization of the underlying flow. We therefore represent Poincaré section $\mathcal{S}$ as a semi-transparent disk placed within the flow and realize the arrows and spot noise as a texture of this disk (see Figs. 5.4 and 5.6). Semi-transparency was used for the map to allow the viewer to see through. This improves the understanding of the context of map $\mathbf{p}$.

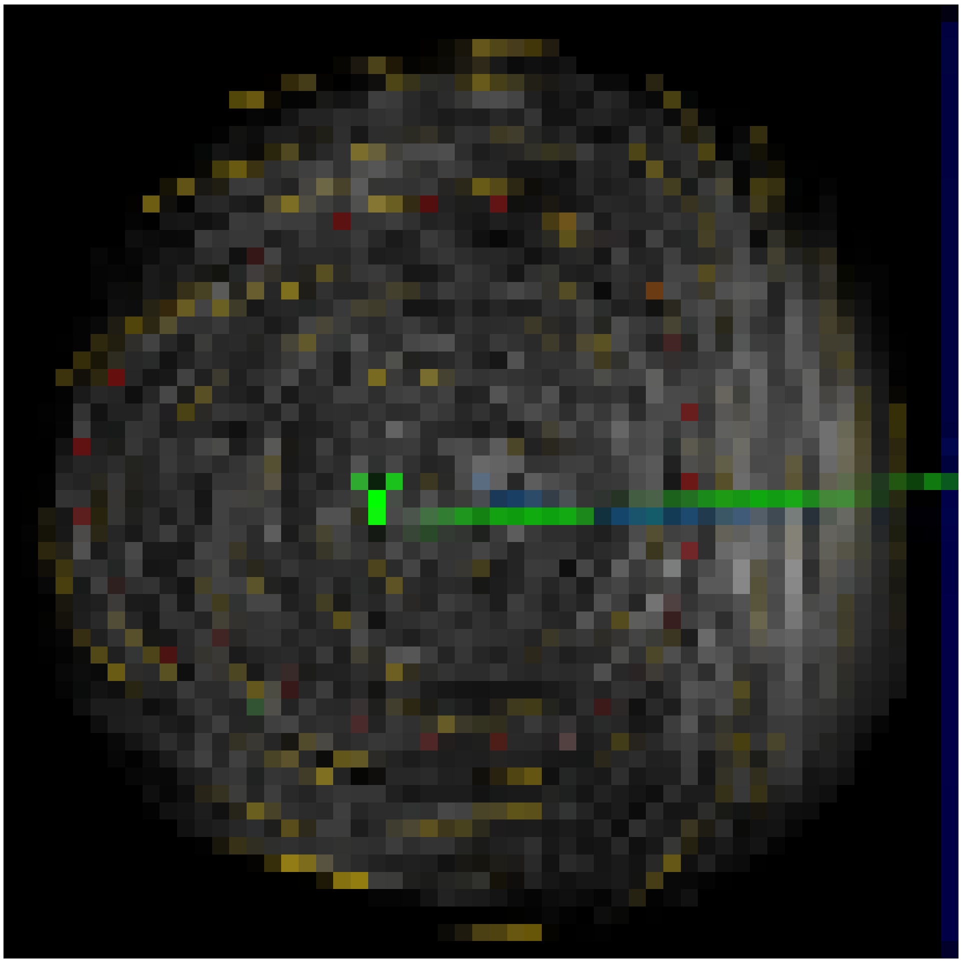


Figure 5.5: Visualizing an cycle attractor using spot noise.

5.5 Visualizing the repeated application $\mathbf{p}^n$

There are several reasons to investigate the repeated application of Poincaré map $\mathbf{p}$, i.e., $\mathbf{p}^n$. Probably the most important one is the analysis of the asymptotic behavior induced by the (iterated) Poincaré map. Periodic systems near a cycle can exhibit different asymptotic behavior, e.g., convergence or divergence with respect to the cycle. This aspect is due to different possible cycle characteristics, namely stable, saddle, or unstable behavior. A stable cycle attracts near trajectories, whereas unstable and saddle cycles repel near trajectories. A saddle cycle $\mathcal{C}$ separates its Poincaré section $\mathcal{S}$ into regions

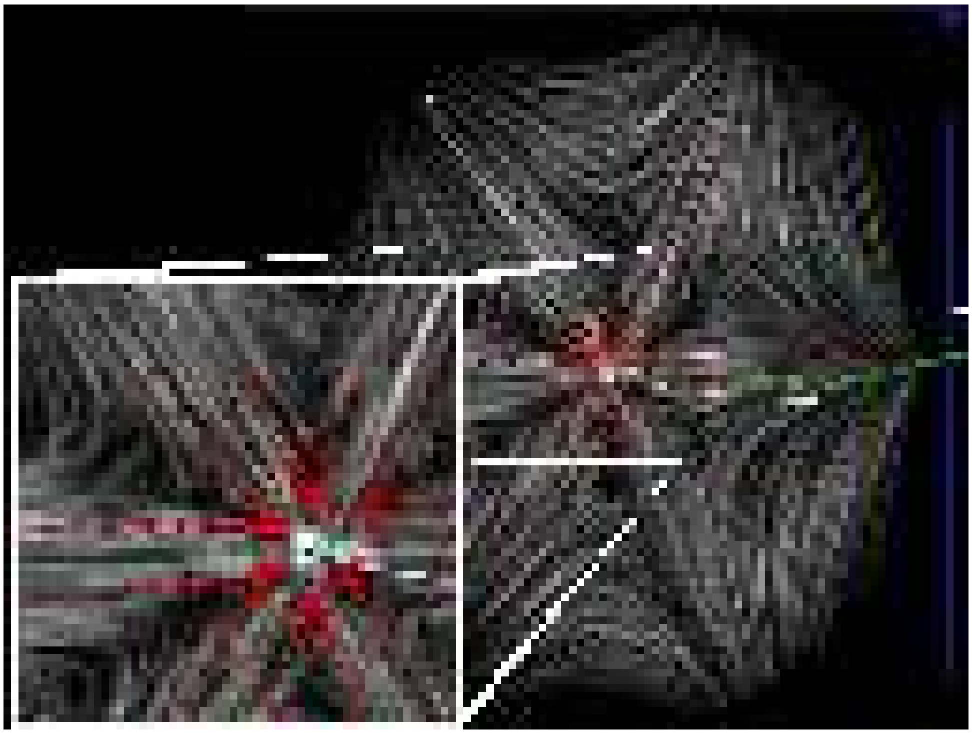


Figure 5.6: Visualizing a non-hyperbolic saddle cycle.

of attraction and regions of repulsion. Almost all trajectories near $\mathbf{c}=\mathbf{p}(\mathbf{c})$ emerge into the repelling parts of $\mathcal{S}$ and thus are finally repelled from $\mathcal{C}$. Figs. 5.4 and 5.6 show two Poincaré maps of different saddle cycles.

We implemented a module TRAJECTORY which produces a visualization of the set $\{\mathbf{p}^j(\mathbf{x}) \mid j \geq 0\}$ as described on page 58. See Fig. 5.6 for another example where this technique was used.

Sometimes $\mathbf{p}^q$, $q>1$, is more interesting to investigate than $\mathbf{p}$. This is the case, for example, when base cycle $\mathcal{C}$ itself pierces the Poincaré section q times during one complete loop (see Fig. 5.7). In the given example $\mathcal{C}$ intersects $\mathcal{S}$ two more times before it returns to the initial intersection point and thus closes the cycle. The behavior of trajectories $\mathcal{T}$ near $\mathcal{C}$ are better described by one arbitrary $\mathbf{x} \in \mathcal{T} \cap \mathcal{S}$ and $\mathbf{p}^3(\mathbf{x})$ rather than $\mathbf{x}$ and $\mathbf{p}(\mathbf{x})$. In fact any pair $(\mathbf{p}^j(\mathbf{x}), \mathbf{p}^{j+q}(\mathbf{x}))$, $0 \leq j < q$, can be chosen for this type of analysis.

The user can change the default value of q, i.e., 1, such that all the previously discussed visualization techniques, e.g., spot noise, are adapted to this new parameter setting. Moreover we allow the user to specify that

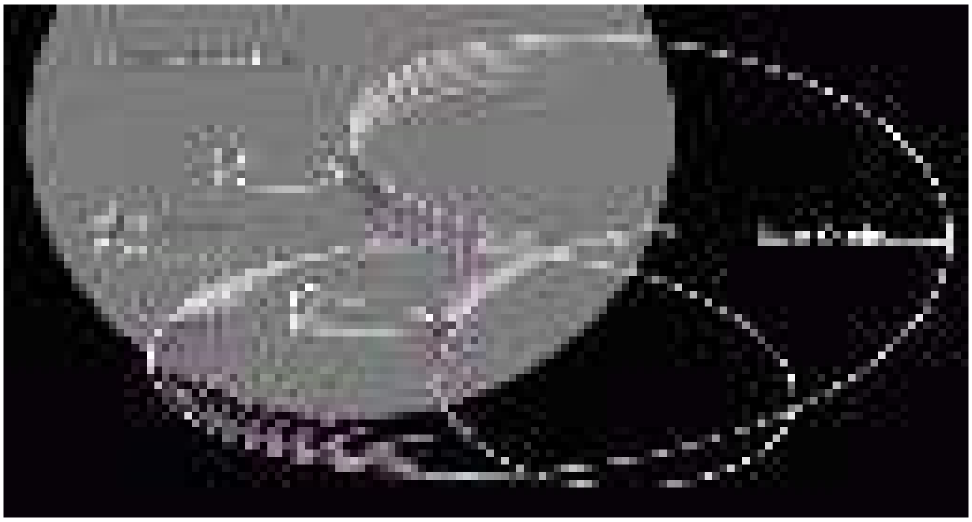
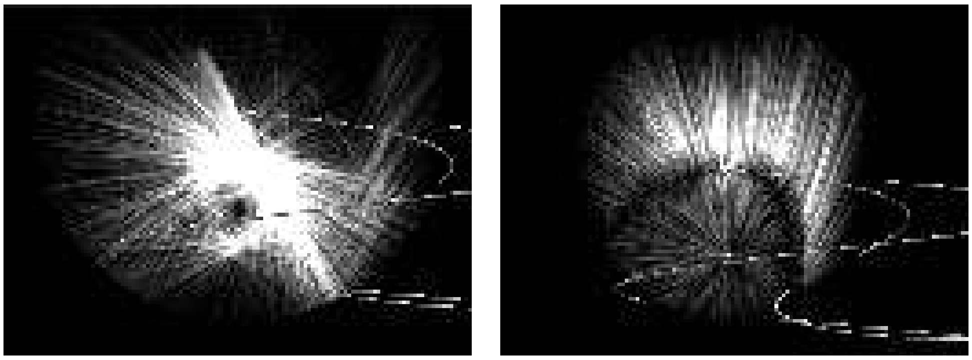


Figure 5.7: Visualizing why $\mathbf{p}^q$ is sometimes more expressive than $\mathbf{p}$.



(a) (b)

Figure 5.8: Visualizing (a) $\{(\mathbf{x}_i, \mathbf{p}(\mathbf{x}_i))\}$ vs. (b) $\{(\mathbf{x}_i, \mathbf{p}^3(\mathbf{x}_i))\}$.

only those intersections $\mathbf{y} = \mathcal{T} \cap \mathcal{S}$ are considered where $\mathbf{f}(\mathbf{y}) \cdot \mathbf{f}(\mathcal{C} \cap \mathcal{S}) > 0$ ($\mathbf{f}$ being the underlying 3D flow). Only those points on trajectory $\mathcal{T}$ are of interest where $\mathcal{T}$ crosses the Poincaré section with the same orientation. This means that $\mathcal{T}$ crosses $\mathcal{S}$ in both cases either in front-to-back or back-to-front orientation. Fig. 5.8(a) was rendered with q=1 and Fig. 5.8(b) with q=3. In this case the base cycle pierces $\mathcal{S}$ three times during one complete loop.

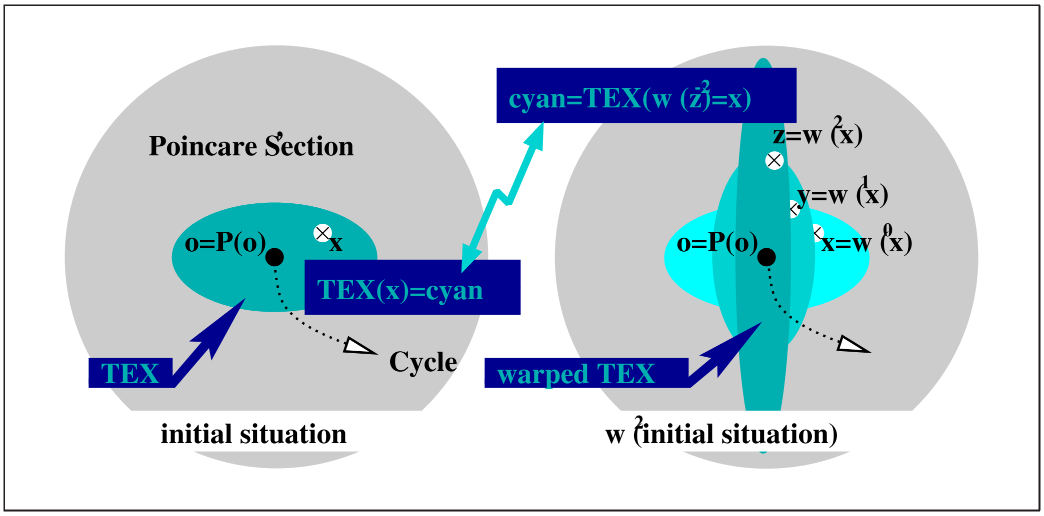


Figure 5.9: Evaluating the initial texture after two applications of $\mathbf{w}$.

Although there are still some artifacts in Fig. 5.8(b) which are due to the limited size of $\mathcal{S}$, it is more expressive than Fig. 5.8(a). The egg-shaped intersection of an invariant torus (containing the base cycle) and Poincaré section $\mathcal{S}$ can be clearly seen as dark line around the center of this image. Furthermore the unstable cycle within this torus cross-section can be distinguished as a critical point of the Poincaré map $\mathbf{p}$. The radial repulsion away from this critical point towards the torus is well represented by the star like spot noise texture.

The visualization of $\mathbf{p}^n$, $n{>}1$, is more difficult than visualizing $\mathbf{p}$ itself. A technique we investigated for the representation of $\mathbf{p}^n$, n increasing, is image warping [11]. A module Warp was implemented that approximates $\mathbf{p}$ by a warp function $\mathbf{w}$ on the basis of $\mathcal{L}$ and $\mathbf{p}(\mathcal{L})$ where $\mathcal{L}$ can be chosen to be either a jittered or regular set of line segments spread over $\mathcal{S}$. This approximation is necessary since evaluating the Poincaré map for all the points on the Poincaré section would cause extremely high computational efford – for each single evaluation potentially thousands up to millions of numerical integration steps are necessary – thus only a few evaluations are done, i.e., $\mathbf{p}(\mathcal{L})$, and warping is used to approximate $\mathbf{p}^n$.

Warp loads an initial texture TEX onto $\mathcal{S}$ and then applies the warp transformation n times, where n is specified via a parameter of Warp. The resulting texture $TEX \circ \mathbf{w}^{-n}$ (see Fig. 5.9) placed on $\mathcal{S}$ gives a good impression of the main characteristics of $\mathbf{p}^n$. See Fig. 5.10 for an images rendered using this technique.

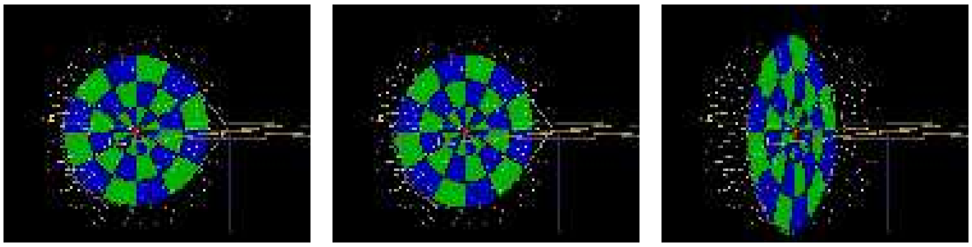


Figure 5.10: Images resulting from one, two, and eleven applications of warp function $\mathbf{w}$, i.e., $\mathbf{w}(\mathcal{S})$, $\mathbf{w}^2(\mathcal{S})$, and $\mathbf{w}^{11}(\mathcal{S})$.

The reason why we approximate Poincaré map $\mathbf{p}$ by the use of a warping function instead of using $\mathbf{p}$ directly for the transformation of the texture is that $\mathbf{p}$ is usually rather costly to compute. Warp function Warp, on the other hand, is capable of approximating $\mathbf{p}$ quite good if warping parameters are chosen appropriately. At least an idea of $\mathbf{p}^n$ is gained using this technique.

5.6 Visualizing Poincaré maps together with the 3D flow

The simultaneous visualization of a Poincaré map and the underlying flow allows to overcome some limitations of 2D visualization techniques of Poincaré maps. Flow characteristics which cannot be derived from 2D Poincaré maps alone as, e.g., the relation between consecutive intersections, can be made visible and thus enrich the capabilities of this visualization technique. See Fig. 5.11 for an example where the secondary rotation of the flow around the cycle could not be derived from the Poincaré map.

A Poincaré section $\mathcal{S}$ is represented as a circular patch that is rendered semi-transparently. Therefore visualization icons before as well as behind $\mathcal{S}$ are visible. The implementation of this technique is based on the existence of a base cycle $\mathcal{C}$ within the 3D flow. Cycle $\mathcal{C}$ defines the center of Poincaré section $\mathcal{S}$. The cycle $\mathcal{C}$ is rendered as an opaque tube through 3D phase space together with a sphere at $\mathcal{C}\cap\mathcal{S}$ where the base cycle intersects the Poincaré section. See Fig. 5.4 or 5.6 for an image where $\mathcal{S}$, $\mathcal{C}$, and the intersection $\mathcal{C}\cap\mathcal{S}$ can be easily detected.

The module which generates this Poincaré map visualization takes care that initially a useful view point is chosen. $\mathcal{S}$ is viewed under an angle which

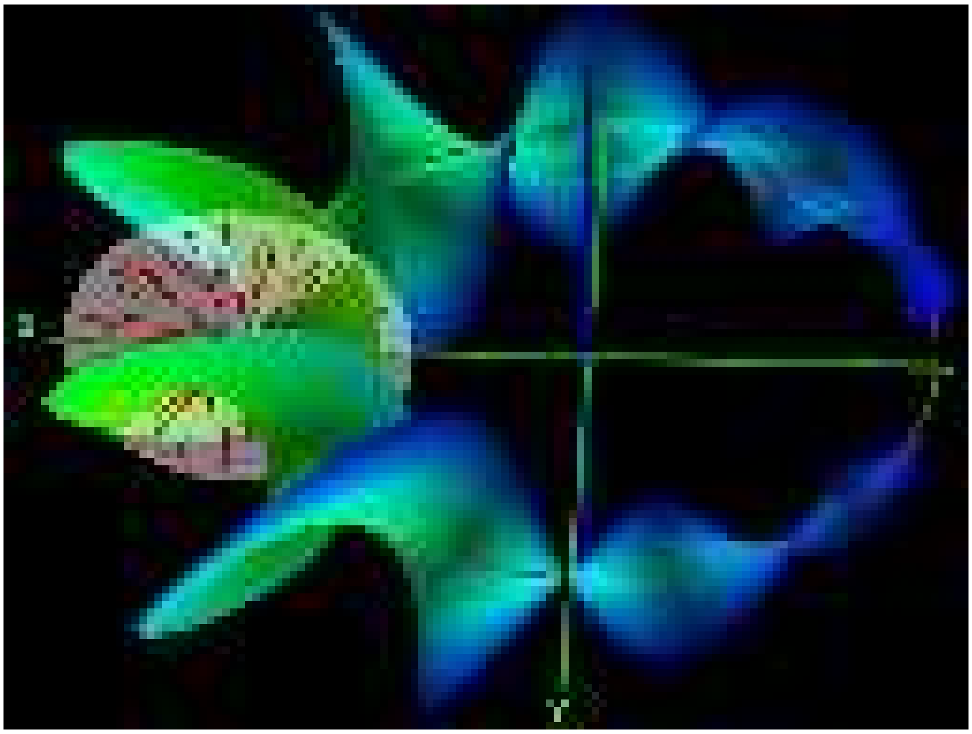


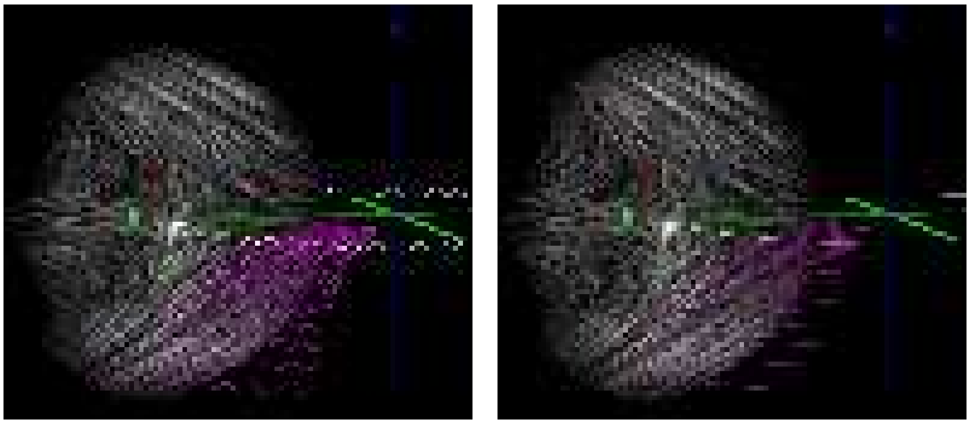
Figure 5.11: Visualizing flow properties not encoded within the Poincaré map.

is little bit less than $\frac{\pi}{2}$ so that both the Poincaré map and the intersecting base cycle are easily recognizable (see Fig. 5.4). We also found it very useful to provide a relative placement capability such that the user can move the Poincaré section easily around the base cycle $\mathcal{C}$. The actual position of the map on the cycle is specified by a value between 0 and 1.

Additionally we suggest some more elaborated 3D extensions to Poincaré map visualization. Module Trajectory, for example, generates either the entire trajectory which constructs the orbit $\mathcal{O} = \{\mathbf{p}^j(\mathbf{x}) \mid j \geq 0\}$, only short parts of this trajectory in the vicinity of $\mathcal{S}$, or just the spheres representing the consecutive intersections.

Drawing the entire trajectory that generates orbit $\mathcal{O}$, for example, helps to relate consecutive points on $\mathcal{S}$ mentally (see Figs. 5.1 and 5.7). Long trajectories, however, may clutter the image.

Using only short parts of the trajectory near $\mathcal{S}$ avoids the problem of visual clutter (see Fig. 5.12). Assume a periodic 3D system which exhibits



(a) (b)

Figure 5.12: Visualizing supplementary information in 3D.

some high frequency oscillation parallel to the rotational axis of the flow (see Fig. 5.12(a)). If the frequency (measured in oscillations per one entire revolution of the carrying periodic system around the rotational axis of the flow) is an integer number, the resulting Poincaré map is not affected by this oscillation at all. Results are the same as for a system without the modulated frequency. Compare Fig. 5.12(a) and 5.12(b). The difference between both systems which is not apparent in the Poincaré maps is visible through the embedding of the Poincaré sections in the 3D flow.

Both images in Fig. 5.12 are generated with techniques already discussed previously in this chapter. Spot noise is used to represent the entirety of $\mathbf{p}$, whereas white and red spheres are used to visualize three sequences $\{\mathbf{p}^j(\mathbf{x}_i) \mid j \geq 0\}_{i\in\{1,2,3\}}$. The sequence starting near the origin of phase space is enhanced by short parts of the constructing flow trajectory. This enhancement is necessary to visualize the differences between both systems.

Another technique for the investigation of Poincaré maps is realized as module SeedLine. By parameters r, ϕ, and *dist* a line segment $\mathcal{Y}$ of length $2 \cdot r$ is specified within $\mathcal{S}$ that is perpendicular to the vector connecting $\mathcal{C}\cap\mathcal{S}$ and the mid-point of $\mathcal{Y}$. The length of this vector is specified by parameter *dist*. Parameters ϕ and r are the polar coordinates of one endpoint of this line segment with respect to its mid-point. SeedLine can be used to generate a stream surface [33] or a rag of stream lines alternatively (see figure on page 54). In addition to the stream surface, the flow trajectory, which constructs a certain sequence of consecutive applications of $\mathbf{p}$, was visualized by the use of a green tube. Other techniques discussed previously in this chapter have been used for visualizing Poincaré section $\mathcal{S}$.

5.7 Animation aspects

Animation is a powerful approach to increase the dimensionality of visualization results. We found the following parameters of the modules suitable to be animated:

TRAJECTORY parameters *no* and *len* – Module TRAJECTORY generates a sequence $\mathcal{O}$ of consecutive intersections of trajectory $\mathcal{T}$ and Poincaré section $\mathcal{S}$, i.e., $\mathcal{O} = \mathcal{T} \cap \mathcal{S}$. Parameter *no* specifies how many intersections should be calculated. *len* can be used to control $|\mathcal{O}|$ via the spatial length of $\mathcal{T}$.

Animating one of these parameters, the construction of Poincaré map $\mathbf{p}$ can be visualized. Furthermore the asymptotic behavior of $\mathbf{p}^n(\mathbf{x})$, $n \to \infty$, can be investigated. This specific application of animation is capable of representing the *inherent* nature of Poincaré map $\mathbf{p}$.

TRAJECTORY parameters r and ϕ – Another pair of TRAJECTORY parameters, which is very well suited for animation, is (ϕ, r). It encodes the initial condition $\mathbf{x}$ of solution $\mathcal{T}(\mathbf{x})$ in polar coordinates with respect to some arbitrary local coordinate system on $\mathcal{S}$. In other words (ϕ, r) specifies the starting point $\mathbf{x}$ of sequence $\{\mathbf{p}^j(\mathbf{x}) \mid j \geq 0\}$.

Animating these two parameters allows to investigate the development of arbitrary curves $\mathcal{Q}$ within $\mathcal{S}$. Such a curve $\mathcal{Q} = (\mathcal{Q}_\phi(s), \mathcal{Q}_r(s))$ should be given as a parameterized subset of $\mathcal{S}$ with s as the parameter. Given such a curve $\mathcal{Q}$, parameter s can be animated: The module TRAJECTORY takes $\mathbf{x}(s) = (\mathcal{Q}_\phi(s), \mathcal{Q}_r(s))$ as an initial condition for the generation of $\{\mathbf{p}^j(\mathbf{x}(s)) \mid j \geq 0\}$. Initial condition $\mathbf{x}(s)$ moves along curve $\mathcal{Q}$, and simultaneously its long-term behavior $\{\mathbf{p}^j(\mathbf{x}(s)) \mid j \geq 0\}$ is visualized.

WARP parameter n – The animation of WARP parameter n (number of applications) improves the expressiveness of the image warping approach. A sequence of images with consecutive applications of the warping function $\mathbf{w}$ shows the overall behavior of $\mathbf{p}^n$, $n \to \infty$. Refer to Fig. 5.10 for three images resulting from consecutive applications of the warping function $\mathbf{w}$.

Figure 5.13: Extreme phase relations as difficult cases for the visualization of Poincaré maps.

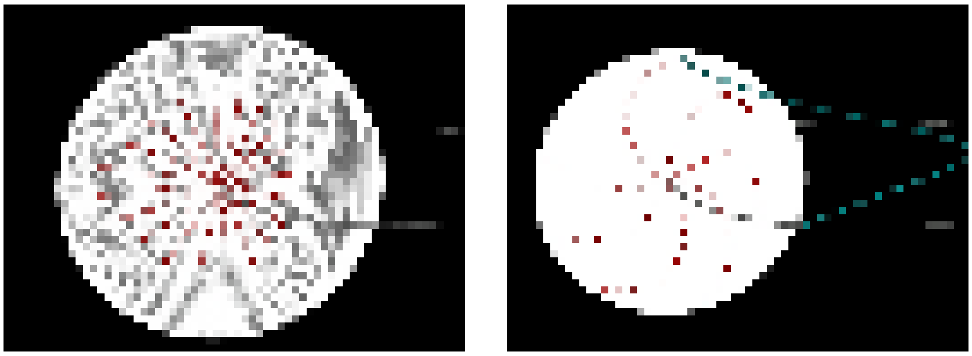


Figure 5.14: Combined visualization techniques to disambiguate results.

5.8 Discussion

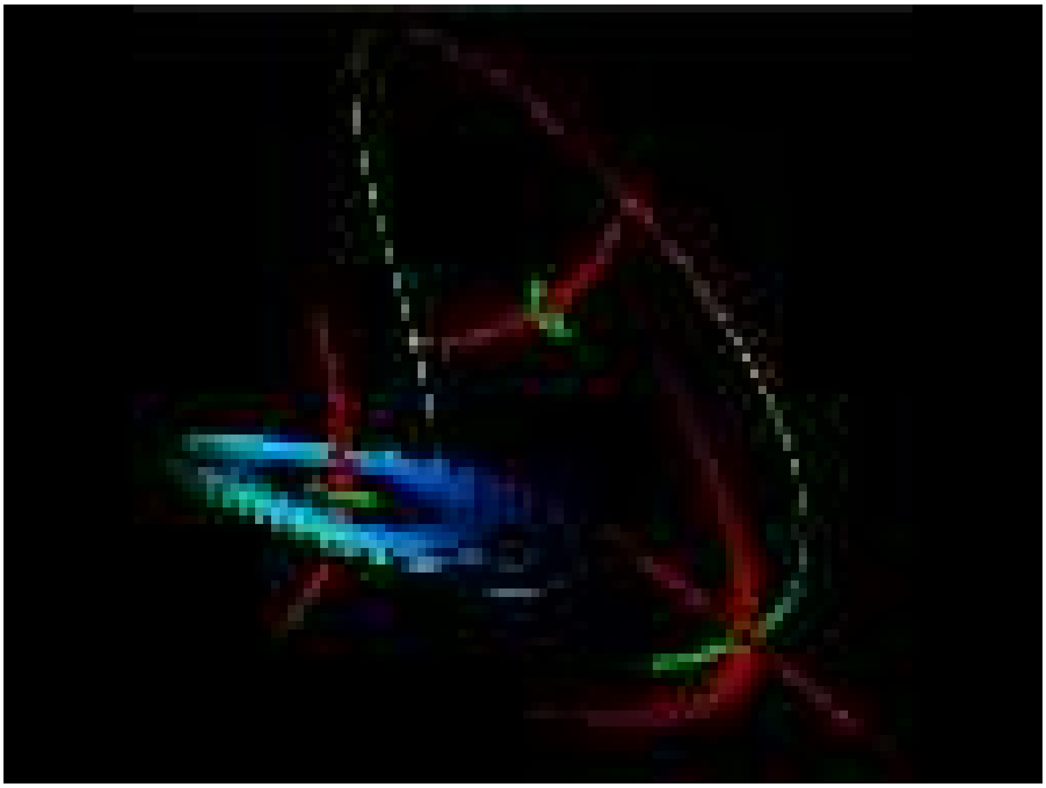
Poincaré maps are a useful mathematical concept for the analysis of periodic or quasi-periodic flows. Thus, it is a good idea to use this concept for the visualization of dynamical systems that exhibit such a periodic behavior. Similarly to the investigation of continuous dynamical system, also the examination of Poincaré maps depends on the investigation goal of the user. The technique proposed in this chapter allows to investigate the long-term behavior of specific points of interest, or one iteration of the map for many points of the Poincaré section (or even the entire set). Furthermore it is useful to integrate visualization on the basis of Poincaré maps and direct flow visualization.

There are, however, some difficult situations, where the application of the Poincaré technique itself is not useful. For instance, if extreme phase relations occur comparing main rotation and secondary rotation, then resulting images usually are difficult to interpret – see Fig. 5.13 for two examples. In cases of frequency coupling, for example, visualization composed of at least two of the presented techniques can help to disambiguate results. See Fig. 5.14 for two examples.

Chapter 6

Visualization of critical points

> To move freely you must be deeply rooted.
>
> *Bella Lewitsky (1916-)*



	density	completeness	quantities
the topological structure	high	usually yes	almost non
the geometry of behavior	medium	potentially yes	just a few
selective direct visualization	regular	no	yes, details

Table 6.1: density and completeness of information, and amount of quantities, provided through different approaches of dynamical system analysis.

After stream arrows for stream surfaces and a visualization technique based on Poincaré maps, a local visualization technique focusing on critical point is presented [44]. Critical points, i.e., system states where there is no evolution at all, usually are properties investigated in the very first place. Examining the neighborhood of the critical points often tells quite important principal characteristics about the entire system behavior. Thus, it's logical to come up with visualization techniques focusing on that kind of information.

6.1 Introduction

As already addressed in Chapter 1, there are many ways to investigate a dynamical system. An abstract approach to analysis is the investigation of the topological skeleton of a dynamical system [5, 6]. The topology of behavior is built from characteristic elements, such as, for example, critical points and separatrices (see also Section 6.2). Once the topological structure of a dynamical system is derived, the long term evolution is known for all the points in phase space qualitatively. However, almost no quantitative information, such as, e.g., spatial extent of trajectories, is provided through this approach.

Spatially quantitative insights are gained from building the geometry of behavior. Location and shape of the topological entities are composed to a geometry within phase space [1, 31]. The topological structure is extended by the spatial attributes of its elements. Thereby at least some quantitative information is provided. The geometry of behavior is a rather dense description of the flow without flow details apart from characteristic structures.

Detailed (local) information is gained by directly visualizing the dynamics using stream lines, stream surfaces, particle systems, or other integral objects [67]. Selected trajectories are visualized through these techniques. This kind of visualization technique always lacks completeness.

As the previously described approaches have their advantages and disadvantages each (see Tab. 6.1 for a summary), a combination of them is appropriate. Most approaches found in literature, on the other hand, concentrate on either the one or the other.

In this chapter we first present two new techniques for the visualization of a continuous dynamical system in 3D space, which belong to two different approaches concerning the analysis of the flow data. Furthermore, we demonstrate that combining both methods yields better results.

6.2 Vector field topology and local analysis

A common procedure to extract the topology of a dynamical system [6] is to proceed in the following way [5]:

1. Identify all the critical points $\mathbf{c}_i$ (also called fixed points, roots, etc.) by solving equation $\mathbf{f_p}(\mathbf{c}_i) = 0$.
2. Investigate the Jacobian matrix of $\mathbf{f_p}$ at the critical points, i.e., $\nabla\mathbf{f_p}|_{\mathbf{c}_i}$. In the hyperbolic case this matrix of derivatives represents the major components of the flow near the critical points. Eigenvalues and eigenvectors intuitively describe the characteristics of the dynamics of $\mathbf{f_p}(\mathbf{c}_i{+}\mathbf{d})$, i.e., of the flow near $\mathbf{c}_i$.

   Depending on the local flow characteristics, critical points are classified as attractors, repellors, or saddles. Focal critical points exhibit a pair of conjugated complex eigenvalues, whereas nodes exhibit real eigenvalues only.
3. Search for higher order characteristic elements, such as, e.g., cycles $\mathcal{C}_i(t)$ ($\mathcal{C}_i(t{+}T) = \mathcal{C}_i(t)$). Again perform an analysis of derivatives to learn about local flow characteristics.
4. Extend the eigen-manifolds near the characteristic elements into phase space to determine their relation to other characteristic elements. Thereby structures such as separatrices, i.e., elements which divide the phase space into regions of qualitatively different dynamics, are identified.

Extracting location and shape of the topological elements, the geometry of behavior [1] is constructed. For a review of state of the art visualization techniques concerning flow topology see Chapter 2.

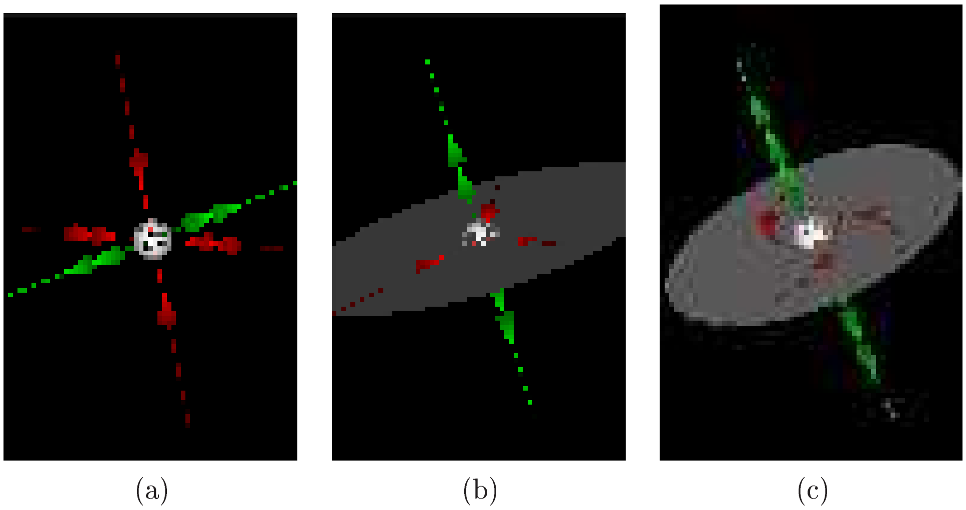


Figure 6.1: Visualizing the geometry of behavior near the critical point of a linear dynamical system – three different saddle configurations.

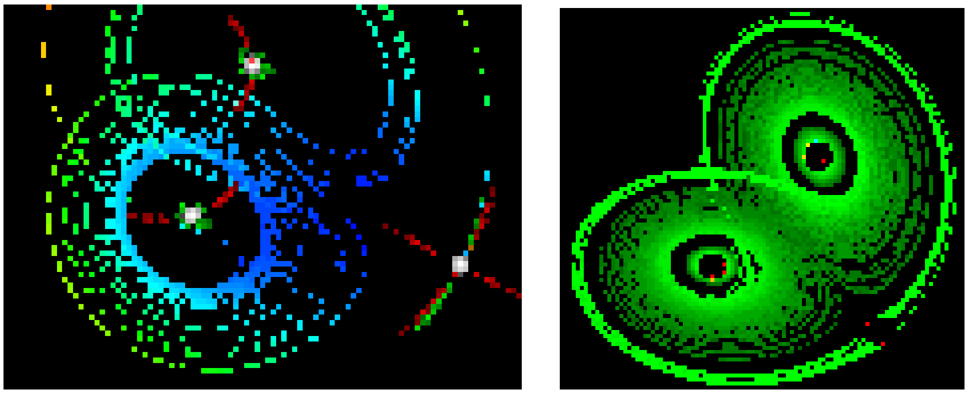
6.3 CharDirs – visualizing eigen-manifolds

The first visualization technique presented here, i.e., CharDirs, is similar to the iconic representation of flow near critical points [31, 74]. Instead of using a glyph to encode the flow topology, we directly represent the geometry of behavior by the use of stream lines and stream surfaces. We first inspect the eigenvalues of the Jacobian matrix and distinguish between topological different cases:

1. If all the eigenvalues are real, different from each other, and different from zero, three pairs of stream lines are integrated into the direction of the corresponding eigenvectors. Thereby the (locally) most significant trajectories are depicted (see Fig. 6.1(a)).

2. If all the eigenvalues are real, different from zero, but two of them are equal, a 1-manifold and a 2-manifold corresponding to the double eigenvalue build up the geometry of behavior near the critical point. In addition to a pair of stream lines we use three stream lines within the 2-manifold plus an optional stream surface to encode this special flow topology (see Fig. 6.1(b)).

3. If two eigenvalues are complex and the real parts of all eigenvalues are different from zero, the same geometry of behavior is present as



(a) (b)

Figure 6.2: Visualizing the flow characteristics near the critical points of the Lorenz system: (a) CharDirs and (b) SphereTufts.

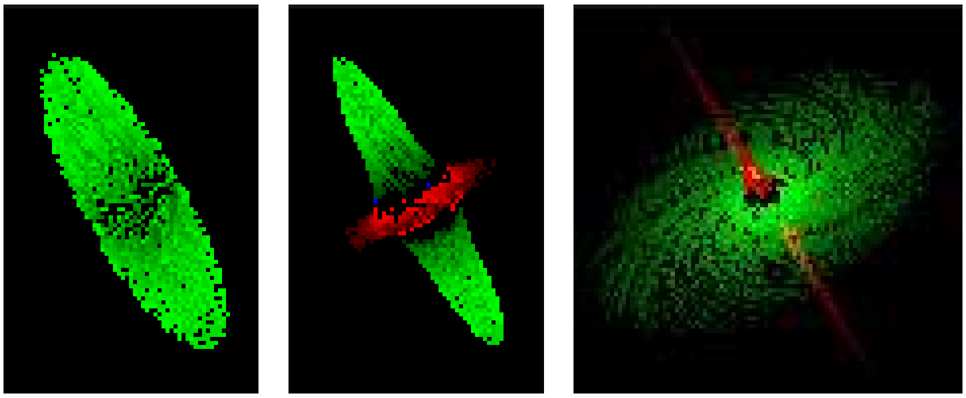
> in the second case. Thus the same visualization technique is used (see Fig. 6.1(c)). However, the flow characteristics are different – spiraling vs. radial attraction/repulsion occurs.

To add more quantitative information we encode the order of magnitude of the eigenvalues by a certain number of arrows along the characteristic trajectories. Thereby the geometry of behavior near critical points is visualized for the most important cases. Degeneracies of flow geometry, e.g., non-hyperbolic critical points, are not considered through this approach.

In Fig. 6.2(a) the Lorenz system [63] was visualized by the use of this technique. Two saddle foci with (each) a pair of conjugated complex eigenvalues and a large negative real eigenvalue drive the rotating characteristic of this chaotic dynamical system. A third saddle coordinates the alternating dominance of these two foci.

6.4 SphereTufts – using many streamlets

The second technique presented here, i.e., SphereTufts, is based on direct visualization of flow rather than on topological or geometrical analysis. A bunch of streamlets (many short stream lines) is used near the critical points of a dynamical system to directly represent the flow characteristics. The seed points of all the streamlets are stochastically chosen on a small



(a) (b) (c)

Figure 6.3: Visualizing a linear system near the critical point with a pair of conjugated complex eigenvalues: (a) a repellor and (b,c) two saddles.

sphere around the critical point. Forward integration as well as backward integration in time is performed with a fixed length in time. Thereby the spatial length of the streamlets directly encodes velocity. Forward streamlets are colored differently from backward integrated ones to distinguish between saddles from attractors/repellors.

In Fig. 6.3 three examples are given. Fig. 6.3(a) shows a linear repellor focus, whereas Fig. 6.3(b) depicts the saddle focus of a linear dynamical system. Fig. 6.3(c) shows also a linear saddle focus as in Fig. 6.3(b) with different eigenvalues. Using this direct visualization technique subtle differences in the flow characteristics become visible. By the use of SphereTufts the differences between Fig. 6.3(b) and Fig. 6.3(c) become visible, although the flow geometry is identical in both cases: the flow component related to the real eigenvalue is much stronger in Fig. 6.3(c) than in Fig. 6.3(b).

In Fig. 6.2(b) the Lorenz system was visualized by placing bunches of streamlets near the critical points. The most important flow characteristics are intuitively depicted. See Fig. 6.4 for a visualization of the Lorenz system by the use of SphereTufts and CharDirs.

Although the results gained with this method are quite expressive, the uniform distribution of streamlet seed points over a sphere enclosing the critical point might not be the optimal choice. Using a seed value distribution, which reflects the distance to the characteristic directions, i.e., the

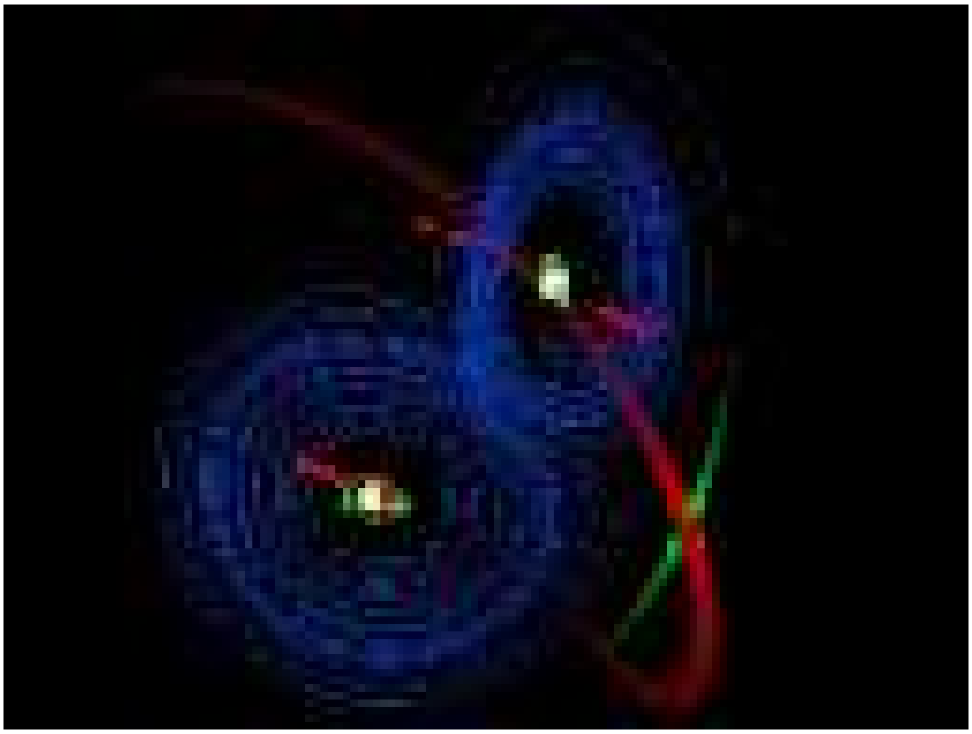


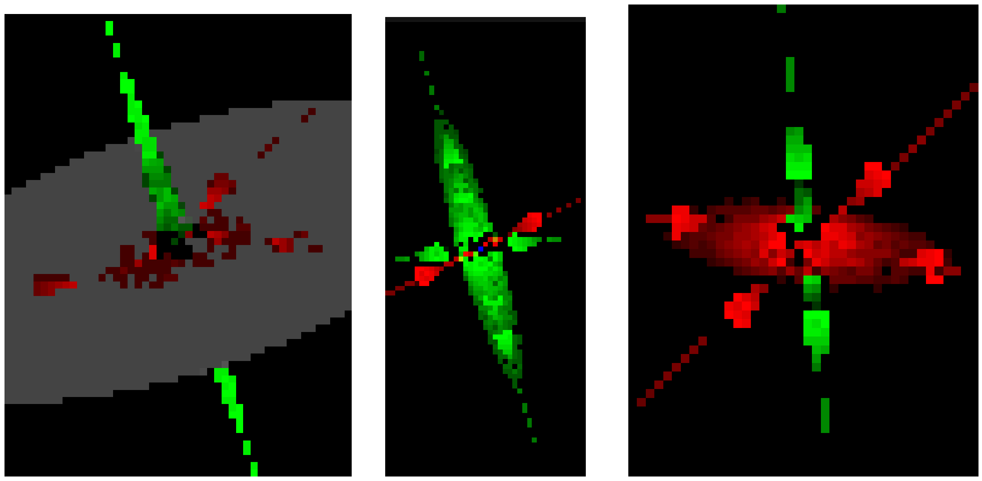
Figure 6.4: Visualizing the Lorenz system using CharDirs and SphereTufts.

trajectories which coincide with the eigenvectors of the critical point, the visualization can be improved.

6.5 Combining CharDirs and SphereTufts

If both methods presented in the previous two sections are combined, the following advantages are achieved: first, a complete description of the flow topology is given through the CharDirs visualization. Additionally direct visualization cues are used to intuitively describe the local flow dynamics.

Fig. 6.5 demonstrates this combination of advantages in three different linear cases. A saddle focus (Fig. 6.5(a)) and two saddle nodes (Fig. 6.5(b,c)) are shown. Topology, geometry, and flow details are integrated within one pictorial representation. The figure on page 70 shows the Lorenz attractor visualized with this combined technique. The strong differences concerning



(a) (b) (c)

Figure 6.5: Three different saddle configurations visualized using both the eigen-manifold visualization and bunches of streamlets.

velocity (about three orders of magnitude) become visible through direct visualization, while still topological information is provided.

6.6 Discussion

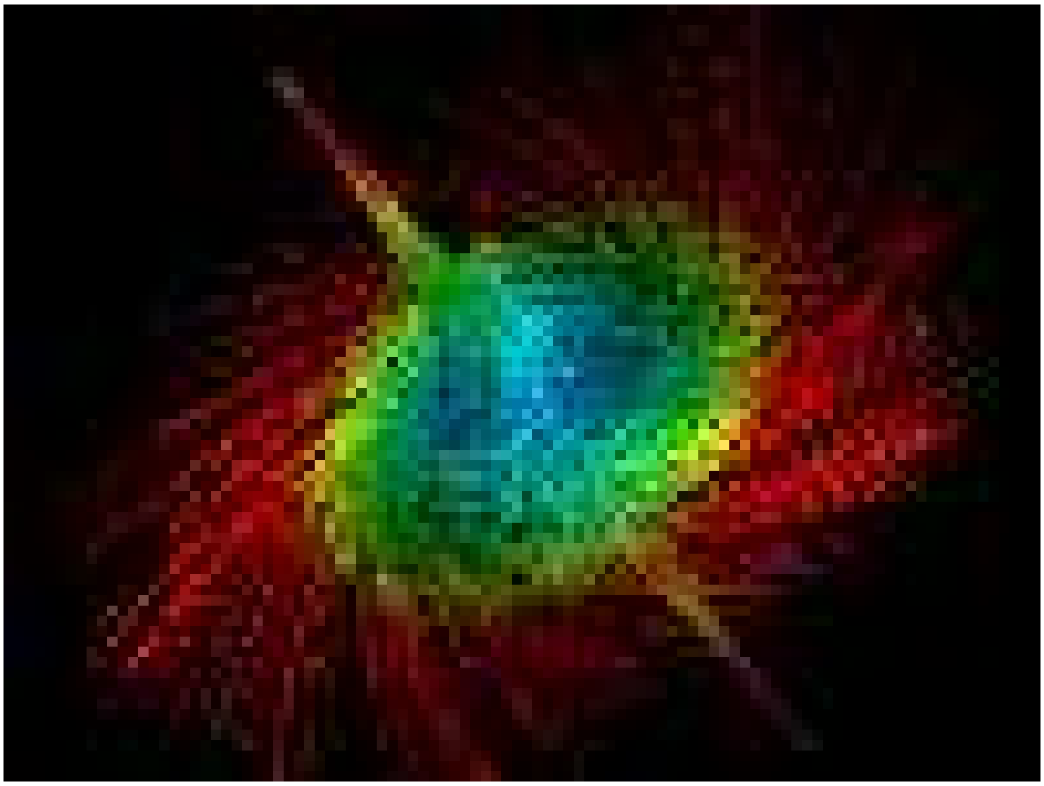
Most approaches of either visualizing the geometry of behavior or, on the other hand, directly representing flow dynamics, have advantages and disadvantages. Visualizing the topology of behavior facilitates a dense and abstract description of the flow dynamics. Directly visualizing the vector field yields more intuitive representations of the dynamical system by simultaneously lacking completeness.

In this chapter two techniques were presented that belong to both classes and demonstrate how a combination of both can be used to achieve a more expressive visualization method. Enhancing the abstract information by cues of direct visualization helps the viewer to more intuitively depict flow motion within critical regions.

Chapter 7

Visualizing characteristic trajectories

Slight not what's near through aiming at what's far.
Euripides (485-406 BC)



The fourth approach to visualizing dynamical systems in this row of different techniques is a result of the aim to locally enhance the visualization of characteristic trajectories [46]. System analysis often results in a set of characteristic points and stream lines. Visualizing such stream lines together with information about the behavior in their vicinity has the power of representing crucial information in a compressed way instead of over-populating phase space with direct visualization cues.

7.1 Introduction

Several approaches to the visualization of dynamical system can be distinguished [47]. One class of techniques deals with the visualization of *characteristic elements* as, e.g., critical points, cycles, or separatrices. A structure of lower-dimensional objects is composed in phase space to describe the key features of the system's behavior [1]. For example, a separatrix is visualized to indicate two subsets of phase space with qualitatively different dynamics. A brief overview on the relation between local linearization and characteristic structures can be found in Chapt. 3.

Another class of approaches deals with the *direct visualization* of the system behavior. Integral curves visualize the evolution of specific seed points which change according to the dynamics of the underlying flow. Many techniques are already available for the 2D case. Spot noise [87] and line integral convolution (LIC) [14], for example, provide an overview of 2D dynamics within a 2D domain. In 3D, however, direct visualization is difficult. Rendered images tend to be overloaded when entire portions of flow in three-space are simultaneously visualized. Some attempts into this direction are illuminated stream lines [94] and volume-rendered 3D flow [25]. Work by Interrante et al. [35] also addresses this problem.

In addition of the visualization of characteristic elements and direct visualization, a third class of techniques deals with the representation of local properties [52]. Glyphs [19, 86] can represent quantities derived from the Jacobian matrix (local linearization of the flow) as, e.g., acceleration, rotation, or divergence. Another approach [75] transforms a polygon positioned perpendicular to a trajectory to represent local information.

In this chapter we present a technique which to a certain extent belongs to all of the three classes mentioned above. It was inspired by the concept of modeling knit wear as yarn with a complex micro-structure [28]. A yarn thereby consists of many fibres with similar spatial location and orientation.

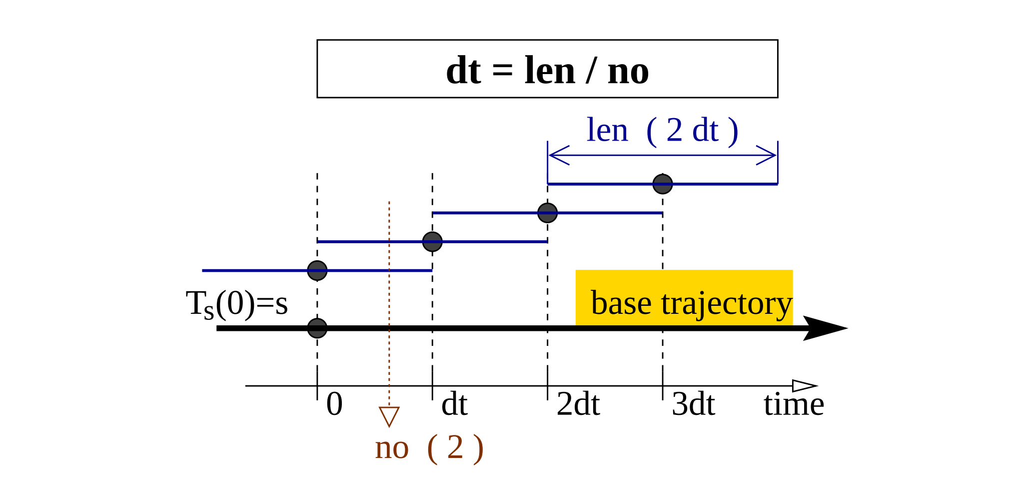


Figure 7.1: Relation between streamlet density (no), streamlet integration length (len), and streamlet instantiation interval (dt).

We visualize the vicinity of interesting trajectories, e.g., the stream lines emanating from critical points. A large number of short integral curves (streamlets) is used to directly code the system's behavior near the base trajectory. By this approach of selectively placing streamlets we omit distracting image cluttering while still providing direct cues to the (local) system behavior. Visualizing the vicinity of characteristic stream lines enhances the abstract representation of the system's behavior by local cues of direct visualization.

7.2 A thread of streamlets

To come up with a useful technique of locally enhanced stream lines, we propose a model for the generation of a *thread of streamlets*. Near a stream line of interest $\mathcal{T}_\mathbf{s}$ (the *base trajectory*) many short streamlets are placed. Thereby a continuous representation of the system's behavior in the vicinity of the base trajectory is approximated.

Using constant flow as a reference model – stream lines are straight lines in this case – the thread of streamlets $\{\mathcal{T}_{\mathbf{s}_i}\}$ is defined as follows: Any cross-section perpendicular to base trajectory $\mathcal{T}_\mathbf{s}$ is pierced by a constant number (no) of streamlets. Using integration time t as parameterization of the base trajectory ($\mathcal{T}_\mathbf{s}(0) = \mathbf{s} =$ seed point of $\mathcal{T}_\mathbf{s}$), streamlets $\mathcal{T}_{\mathbf{s}_i}$ are instantiated at time $t_i = i \cdot dt$ and integrated over the time interval $[i \cdot dt \pm len/2]$. See Fig. 7.1 for an illustration of the relationship between no, dt, and len,

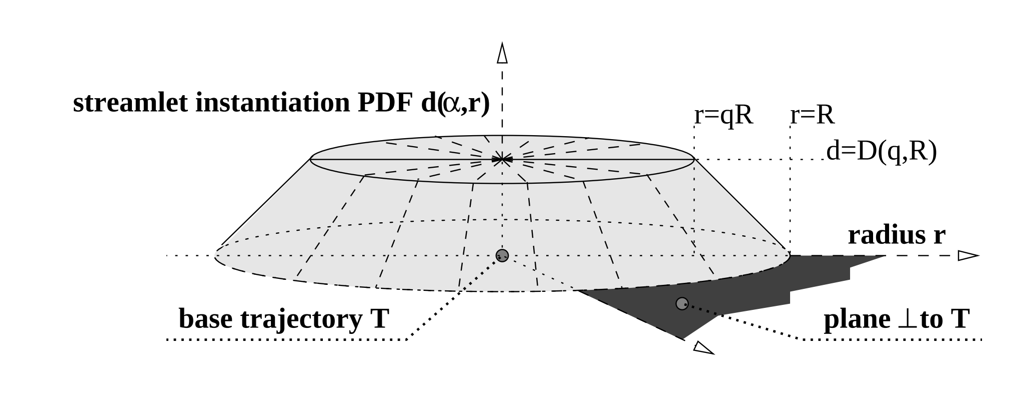


Figure 7.2: Probability density function $d(\alpha, r)$ for the instantiation of streamlets based on a perpendicular cross-section through the base trajectory.

i.e., $dt = len/no$. Seed points $\mathcal{T}_{\mathbf{s}_i}(t_i) = \mathcal{T}_{\mathbf{s}_i}(i \cdot dt)$ of newly instantiated streamlets are randomly chosen within a perpendicular cross-section through $\mathcal{T}_{\mathbf{s}}(i \cdot dt)$ (reference location on the base trajectory) corresponding to a probability distribution function (PDF) $d(\alpha, r)$ (see Eq. 7.1 and Fig. 7.2).

$$d(\alpha, r) = \begin{cases} D & \text{if } 0 < r \leq qR \\ \frac{R-r}{R-qR} D & \text{if } qR < r \leq R \\ 0 & \text{if } R < r \end{cases} \tag{7.1}$$

PDF $d(\alpha, r)$ is defined by parameters R (the maximal distance between $\mathcal{T}_{\mathbf{s}}(t_i)$ and $\mathcal{T}_{\mathbf{s}_i}(t_i)$) and $q \in [0, 1)$. The latter parameter is used to define PDF d as a truncated cone. This shape provides the fade-out characteristic of the streamlet placement procedure with respect to the distance from $\mathcal{T}_{\mathbf{s}}$. To guarantee that d is a PDF, $\int d(\alpha, r)\, \mathrm{d}\alpha\, \mathrm{d}r$ must equal to 1, i.e., the volume of the truncated cone must be 1. This constraint can be expressed as specification for parameter D:

$$D = \frac{3}{(1 + q + q^2) R^2 \pi}$$

In other words,

- many streamlets are positioned around a certain base trajectory $\mathcal{T}_{\mathbf{s}}$ in a circular fashion. Thus, polar coordinates (r and α) where used to describe the seed points of the streamlets.

- Through PDF d the generated streamlet distribution is uniform within a certain radius (qR) and fades out linearly outside radius qR. This way of instantiating streamlets emphasizes the flow near base trajectory $\mathcal{T}_{\mathbf{s}}$.

Computing a thread of streamlets for the reference model ($\dot{\mathbf{x}}$=**const**.), a bunch of line segments (streamlets $\{\mathcal{T}_{\mathbf{s}_i}\}$) of equal spatial length ($len \cdot |\dot{\mathbf{x}}|$) is generated. It this case of constant flow the streamlets are parallel to the base trajectory which is a straight line itself. The seed points of the streamlets, i.e., $\{\mathbf{s}_i\}=\{\mathcal{T}_{\mathbf{s}_i}(t_i)\}$, are determined according to the PDF $d(\alpha, r)$. For any time t the cross-section perpendicular through $\mathcal{T}_{\mathbf{s}}(t)$ is pierced by exactly $no = len/dt$ streamlets.

Applying this model to real (usually non-constant) flow data, local flow characteristics are visualized through the following variations from the constant flow reference setup:

- the **shape of the streamlets** directly visualizes the flow locally to the base trajectory. Local convergence/divergence or rotational behavior with respect to the base trajectory is intuitively depicted. Since local variations are significant in the area of (partial) degeneracies of the flow, characteristic trajectories are especially well suited to be chosen as base trajectories.
- the **streamlet length** is a direct visualization of flow velocities near the base trajectory. Flow velocity, for example, can be depicted very good. Compared to color coding which is often used for velocity visualization the use of streamlets is more effective.

Taking a linear node repellor with eigenvalues 1, 10, and 100 as example, the flow characteristics in the vicinity of this critical point can be visualized in different ways (see Fig. 7.3). Using threads of streamlets for a visualization of the characteristic trajectories – those which are aligned with the eigenvectors of the critical point's Jacobian matrix – a dense and intuitive representation of the 3D flow near the critical point is generated. Through the threads of streamlets (Fig. 7.3(b)) the flow next to the characteristic trajectories is visualized. A purely abstract notation (Fig. 7.3(a)) encodes the eigenvectors of the Jacobian matrix and the magnitudes of the associated eigenvalues. No direct information about the vicinity of the characteristic trajectories is provided.

7.3 Rendering

Drawing 1D objects in 3D space poses several problems in the rendering stage. Shading, for example, improves the visual cues concerning the spatial arrangement of objects, but shading is usually defined on the basis of a

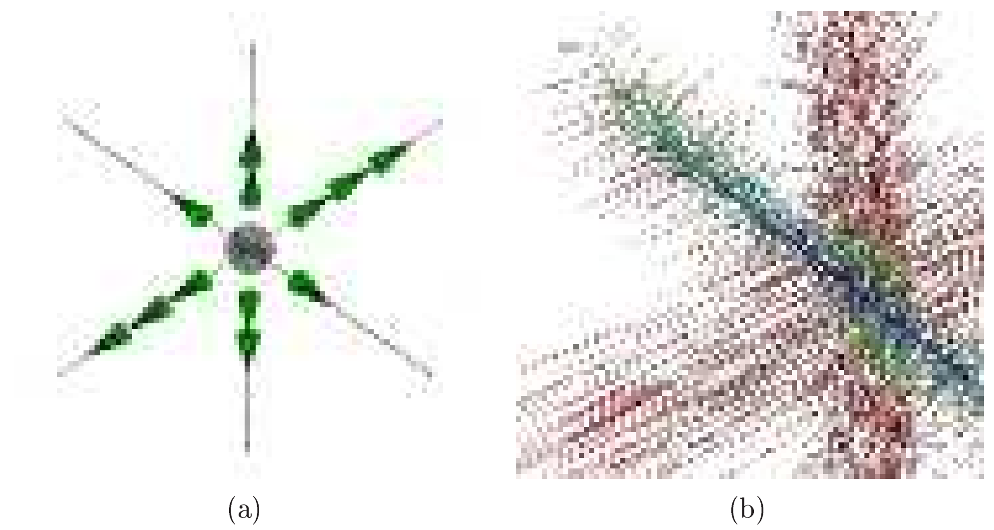
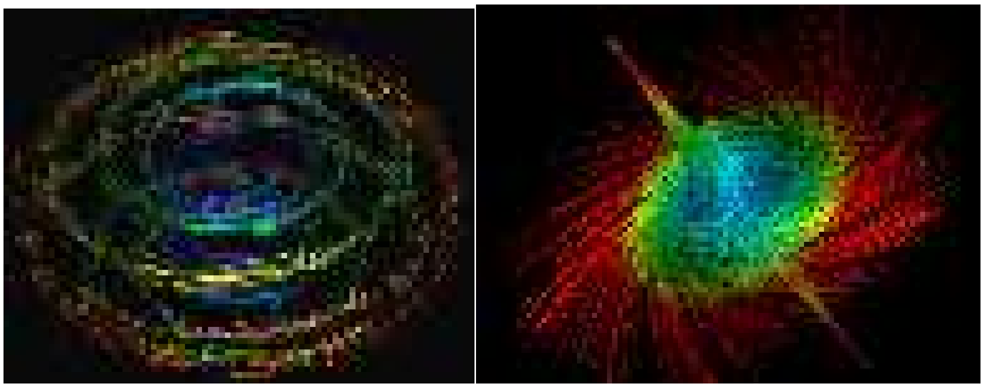


Figure 7.3: Visualizing the flow near a linear node repellor in 3D: (a) eigenvectors and eigenvalues, (b) characteristic trajectories plus threads of streamlets.

surface (normal). In 3D lines and curves have an infinite number of normals at each of their points. Therefore typical shading models as Phong shading [64] can not be applied directly to 1D objects in 3D.

In 1989 Kajiya presented an "ad hoc" approach to deal with the problem of line shading in 3D which is based on an integration of all reflected intensities [38]. In 1996 Zöckler et al. described an efficient computation scheme for line shading in 3D which generates comparable results to the technique proposed by Kajiya [94]. A general framework for the task of shading k-manifolds in n-space was worked out by Banks in 1994 [8]. In addition to a consistent framework for shading with arbitrary co-dimensions Banks also dealt with the problem of excess brightness-compensation which becomes an important topic when manifolds with co-dimension higher than 1 are shaded.

Another problem associated with line shading in 3D is (self-)shadowing. Normally, when shading 2-manifolds in 3-space, we (implicitly) deal with this aspect by assuming all surface points in (self-)shadow, where the outward normal $\mathbf{n}$ points away from the light vector $\mathbf{l}$, i.e., $\mathbf{n}\cdot\mathbf{l} < 0$. Furthermore we consider shadow rays before we compute surface shading. Both aspects are difficult with line shading in 3D. One approach to deal with these aspects comes from volume rendering: lines populating certain regions of three-space can be considered as volume opacity of a certain density. This assumption



(a) (b)

Figure 7.4: (a) A thread of streamlets visualizing the flow near a torus in 3D. (b) Flow near a 3D focus visualized using two threads of streamlets.

yields an exponential brightness attenuation for light passing through such a region. A paper by Max in 1995 compiles a comprehensive list of diverse models dealing with this effect [53].

For the implementation of this technique the shading model by Zöckler was used for shading the streamlets. Additionally we used depth cueing as a rough approximation of shadowing to enhance the spatial perceptibility of the streamlets in three-space. See Fig. 7.4(a) for an example. The heads of the streamlets are represented by small arrow-heads to indicate the orientation of the flow. Color is used to encode the flow velocity (blue $\leftrightarrow$ slow, red $\leftrightarrow$ fast). Line shading and depth cueing has been applied as described above.

7.4 Results and Implementation

To test the technique we firstly applied it to a simple cases, i.e., the critical points of a linear dynamical system. Depending on the Jacobian matrix evaluated at this point, different results are obtained. Fig. 7.3(b) shows six threads of streamlets applied to the characteristic trajectories emanating from the critical point. In this case the eigenvalues of the Jacobian matrix at the critical point are 1, 10, and 100. The new visualization technique allows to easily depict the slow, medium, and fast directions of flow. Moreover, an impression is conveyed, how system states are repelled from the plane defined by the slow and medium direction (eigenvalues 1 and 10). Within that plane states are repelled from the slow direction which itself is therefore extremely instable in this setup. These flow characteristics typical for a dynamical system near a critical point cannot be communicated by either showing an abstraction only (Fig. 7.3(a)) or a complete set of stream lines.

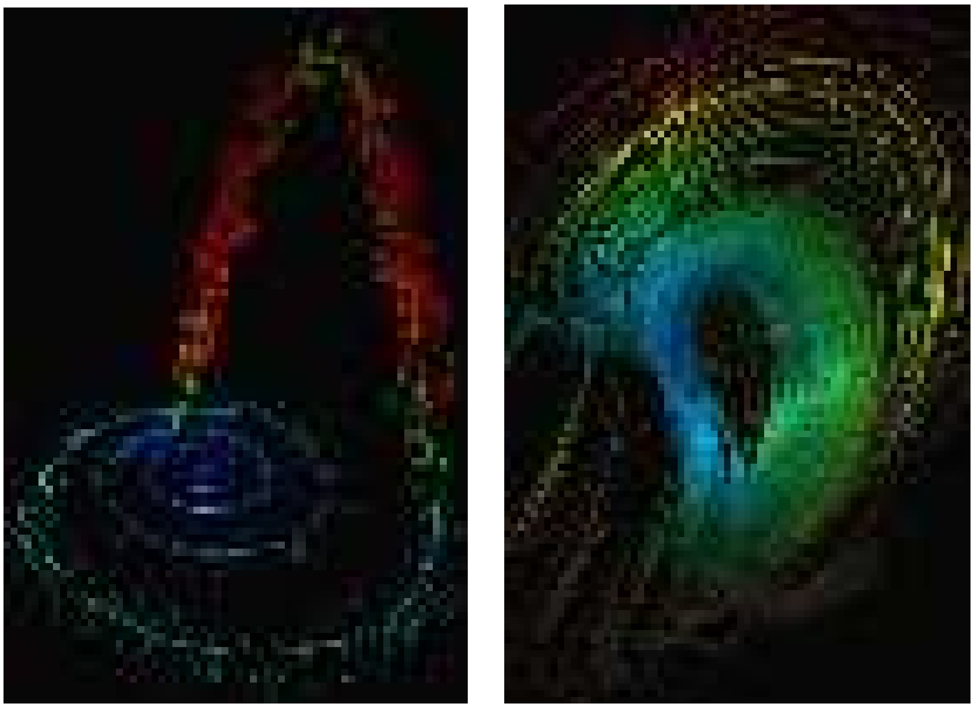
Fig. 7.4(b) is generated by using two threads of streamlets for the visualization of a 3D focus, of a linear dynamical system. The Jacobian matrix of this system exhibits one negative eigenvalue and two conjugated complex eigenvalues with positive real part. System states are attracted along an instable 1-manifold – a line in the case of a linear system – and repelled into a stable 2-manifold (plane) perpendicular to the instable set. Applying the threads to both instable trajectories the dynamics near this critical point are visualized. As in Fig. 7.4(a) color was used to encode flow velocity.

There is no restriction to apply the new technique to characteristic trajectories only. Fig. 7.5 shows two examples where different results where produced with this technique. The left image shows a thread of streamlets through the Roessler system. Instead of the streamlets themselves just arrow-heads at the end of each streamlets are displayed. Using size and color according to the velocity of the flow slow and fast areas within this system are visualized. The right image depicts the dynamics of a periodic flow near a twisted torus. Color coding indicates the velocity along the streamlets. In Fig. 7.5(a) and 7.5(b) no characteristic trajectories were used, the evolution of the streamlets is aligned with the base trajectory. Regions of local convergence/divergence are shown as areas with more/less streamlets.

The technique presented in this chapter was implemented within DynSys3D (see Chapter 8). The module generates one thread of streamlets for a specific dynamical system by using a specific numerical integrator. Parameters for the module are the starting location $\mathbf{s}$ of the base trajectory ($\mathbf{s} = \mathcal{T}_{\mathbf{s}}(0)$) and its length (either temporal or spatial), the number of streamlets per cross-section (no), the maximum distance of their seed-points (R) together with the fade-out parameter (q).

7.5 Discussion

We present a new technique for the visualization of dynamical systems, namely the use of a thread of streamlets for characteristic trajectories. This is useful, since a trade-off is made between only displaying structural information as, e.g., critical points and separatrices, and directly visualizing the system dynamics by the use of stream lines or stream surfaces. As an abstract denotation of the dynamics caused by a dynamical system is very hard



(a) (b)

Figure 7.5: (a) Visualizing the flow velocity near a stream line of the Roessler system. (b) Visualizing the dynamics of a periodic dynamical system exhibiting a twisted torus.

to understand for most users, enhancing this information by locally adding cues of direct visualization helps to communicate the crucial aspects of the system behavior.

Contrary to surface-based stream line visualization techniques like the stream tube of sweep-based trajectory representations, threads of streamlets visualize the flow continuously in the vicinity of a stream line. Using a thread of streamlets instead of entirely populating 3D phase space with stream lines, has the advantage of reducing occlusion. Although quite a number of papers deal with densely visualizing flow in three-space, it seems to be necessary to place visual cues selectively to reduce occlusion problems.

Chapter 8

Implementation: DynSys3D

> As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.
>
> *Albert Einstein (1879-1955)*

> The distance is great from the firm belief to the realization from concrete experience.
>
> *Isabella I of Spain (1451-1504)*

After presenting four different approaches to the visualization of dynamical systems, implementation details about all these approaches (see Chapters 4, 5, 6, and 7) are discussed. A system, named DynSys3D, that itself is based on AVS [2] has been developed which allows to easily combine different visualization techniques corresponding to the needs of the user.

8.1 Introduction

DynSys3D [45] is a flexible environment for implementing and evaluating new ideas in the field of advanced visualization techniques with special emphasis on three-dimensional dynamical systems. For design and implementation of DynSys3D knowledge gained from previous projects (VEGA [82], BaBel [81]) was used. Starting with some specific requirements and goals (see Section 8.2) the system design of DynSys3D (see Sections 8.3 and 8.4) was chosen to comply with most of these goals. DynSys3D provides an experimental workbench to easily investigate visualization techniques and dynamical systems in a collaborative environment.

When starting in a new field of research the question of where to embed the test implementation is very important. Especially in the field of visualization this question is crucial, since implementing the whole visualization pipeline [60] is clearly not practicable or desirable in most situations. In the case of this thesis it was decided to implement and test new visualization techniques in the field of three-dimensional dynamical systems on the basis of AVS [2].

AVS is a general-purpose environment for developing and compositing visualization techniques. Applications are thereby realized as a set of modules (elementary tasks in the visualization process), which are connected to form a data-flow network by associating output and input ports of consecutive modules.

AVS modules contain easy to specify user-interface components. Apart from a large set of predefined modules [3], user defined modules may be implemented. As long as the data of a user-defined module, which is handed over to AVS, conforms to standard AVS data structures, these modules can be easily combined with already existing ones. The availability of many visualization techniques within AVS led us to choose AVS as the basis for DynSys3D.

8.2 System requirements and goals

When starting to build DynSys3D, we wanted to develop a workbench for several researches working collaboratively in the field of three-dimensional dynamical systems. Therefore several requirements and goals influenced the design of DynSys3D:

- Extendibility: It should be very easy to extend the system by new components, e.g., new visualization techniques. Similarly it should be as simple as possible to apply already implemented visualization techniques to new dynamical systems.

- It should be possible to easily compare different techniques, e.g., the behavior of different numerical integration methods. Components of the system, which obviously allow alternative solutions, should be exchangeable as well.

- The system should especially support rapid prototyping in a collaborative environment, i.e., the structure of DynSys3D should allow multiple users to develop additional system components in parallel, even if they depend on each others work.

- Interactivity: Real time visualization is quite difficult to achieve in most situations. Within DynSys3D the geometric representation is controlled by a parameter which influences the geometric level of detail to allow an interactive examination of the results.
- The system design should induce some symmetry guidelines to help developers to implement modules, which are intuitive to combine and use. Questions as where to use the AVS data-flow mechanism or which parameters should be handed over to the user, should be answered prior to individual decisions during the development.
- The system should provide control mechanisms to allow developers to selectively offer only those parameters to the user, which he is primarily interested at in his current investigation. It is very important to adjust number and representation of module parameters to provide a sensible user interface. Too many parameters as well as missing ones hamper the usability of a module.
- It should be possible for developers to concentrate specifically on the visualization part of the investigation. This requirement especially targets related problems as, e.g., how to find critical points or cycles of a dynamical system. General solutions to these tasks were not in the scope of this project.

8.3 DynSys3D: system design

The design of DynSys3D mainly consists of a set of structural specifications and principal decisions. There is no specific functional core of the system, which has to be available when implementing further software based on this workbench. The main advantage of working within DynSys3D is the ability to reuse already existing software components and, simultaneously, to extend the system by new features.

Principal components

A major element of the system is the *principal component*. Initially DynSys3D consisted of three of them: Dynamical System, Numerical Integrator, and Visualization Technique. A module is built by choosing appropriate representatives of the principal components involved and linking them together to an AVS module. For example, a combination of Lorenz, Euler and Streamline, would result in an AVS module, which is capable of generating a stream line which represents the Lorenz attractor by using Euler integration [63].

All representatives of a principal component must conform to an interface given by DynSys3D. The interface specification is represented as a header file within the system. This file lists the maximum capabilities of a principal component. Representatives of a component, which do not implement the whole list of functions are also possible, but may not, due to their limitations, be able to work with certain other component instances. The check whether some instances work together or not is done automatically during the link step.

Separating visualization techniques and module interfaces

The most important principal component of DynSys3D is the Visualization Technique. Instances of this class represent the core of the modules and thus provide the mechanisms that are necessary for building an AVS module. The visualization technique itself should be clearly decoupled from the AVS interface functionality. This allows easy reuse of the visualization-technique code within other compound visualization techniques. For example, it should be possible to reuse the stream line technique within another module which generates an entire rack of stream lines.

AVS modules mainly consist of one main C procedure each. It is invoked by AVS, whenever new data has been propagated to the module through the data-flow network or a module parameter was changed by the user. During each call the current input data and parameter values are handed over to this main function via parameters. Generally this main function performs the visualization technique represented by the module and finally returns the resulting output data to AVS.

Within DynSys3D the main function of a module represents a separation layer between the AVS interface and the implementation of the visualization technique. Parameters, input, and output are managed within this layer and another function is called to perform the required visualization technique. It is this second function which can be reused by other modules without major changes or adaptations.

Multiple developers

DynSys3D was designed to support multiple developers extending the system in parallel. In general several difficulties arise in such a situation: Developers often get a local copy of the system and extend it by their own contributions. Afterwards all these copies have to be merged, which is potentially rather difficult. Even if the instantiation of multiple copies of the system is omitted, problems may arise: Code which was already usable may become corrupt again, when debugging is performed on the code.

The design of the system allows to reduce these problems. Only the directory hierarchy of DynSys3D is replicated for all the developers. They build their implementations locally until it is finished. Whenever a developer needs an already existing system component, a link to the DynSys3D directory tree is installed instead of the replicated local directory. Developers are encouraged to make already working parts of their work early available to the others by placing them as compiled code within the DynSys3D file space. These submitted object files stay available to the others even if the local copy of the developer may not work momentarily. When a new component is finished, it is moved from the developers local hierarchy to the DynSys3D area and installed as standard component.

Focus on visualization

The design of DynSys3D reflects the intention of providing a system, which allows to concentrate on visualization aspects. Some tasks during the analysis of dynamical systems, however, belong to other areas like mathematics and numerics and had to be tackled, at least partially, as well.

One class of such problems is the identification of characteristic subsets in a dynamical system. Writing a piece of software, which is capable of finding all these elements as, e.g., critical points, cycles, and separatrices, is beyond the scope of DynSys3D. In this specific case we transfered the search for these characteristic elements from the visualization component to the dynamical system itself. This is done as it is much easier for a specific dynamical system to declare its critical points than for a general piece of software to identify characteristic subsets for arbitrary dynamical systems. Another reason why this solution seems to be appropriate is that often the results of analytical analyses are available prior to the visualization step.

Interactivity

Visualization usually is demanding in terms of time and computer resources [73]. Complex operations during the several steps of the visualization pipeline and the huge amount of data involved in the calculations slow down the visualization in most cases. Moreover in many cases the data generation, e.g., numerical simulations, take even longer than the visualization itself.

In the case of this thesis the actual visualization techniques are also often far from being interactive. Anyhow we thought that at least an interactive inspection of the results of the visualization should be possible for the techniques investigated. Therefore the visualization modules include a parameter, which controls the complexity of the geometric representation of the result, e.g., the number of triangles.

This geometric representation parameter does not influence the numerical accuracy, which is used for the calculation of the results. It just forces the module to use a courser representation of the results than reported from the calculation.

Symmetry

Another concept of DynSys3D is the aim to increase the amount of software symmetry across different implementations of multiple developers. Symmetry in this context means that implementation should reflect certain macro structures, e.g., semantical relations (stream line $\leftrightarrow$ stream surface) or the modular concept.

The separation of principal components and the specification of the interfaces is a key design element, which leads to more symmetry. Additionally some principal decisions, e.g., the requirement of a parameter controlling the complexity of the geometric representation, were thought to lead to more symmetry throughout the system. The achievement of a high degree of symmetry is formulated as a design guideline which must be adhered to by the different developers.

8.4 Evaluation

After presenting the design of DynSys3D we now want to evaluate the system against the goals and requirements of Section 8.2.

The concept of linking a set of already compiled principal components allows to rapidly extend the system by new components. Existing numerical integrators, for example, can be later combined with new visualization techniques without any re-compilation. Comparing different solutions to a specific problem is also very easy as long as the implementation is extracted as a separate component. Currently we can easily compare different dynamical systems, numerical integrators, and visualization techniques.

The design decision to postpone the combination of components to be done after the compilation step has another advantage as well: Multiple developers can work together and extend the system in parallel. Working components of the implementation are made available to the other developers by placing compile code in the DynSys3D file area.

Interactivity is a tough goal to meet. With most of the investigated visualization techniques it is not possible to achieve real time response. The modules allow a variation of the geometric complexity of the results. Therefore at least an interactive inspection is feasible.

DynSys3D provides some mechanisms, which enforce a certain amount of symmetry. Additionally we are trying to increase the amount of symmetry by specifying guidelines, which developers have to follow. For example, we decided to hand over information about the starting location of stream lines by using the AVS data-flow mechanism, although such a seed point also could be specified by the use of UI elements. This clearly does not hold for the specification of the seed locus of a stream surface. Providing the seed input for both techniques (stream line, stream surface) by the same implementation mechanism, more symmetry is achieved.

The separation of interfaces and the implementation of visualization techniques, allows to reuse certain methods within compound visualization techniques. Input data and parameters of such a reused component can be controlled by the module using it and thus it is possible to reduce the parameter set of any AVS module to a meaningful and intuitive extent.

Up to now we were able to omit situations, where we would have had to include some non-trivial techniques from a field we are not experts in, e.g., numerics. For example, we did not implement a general search engine, which reports the characteristic subsets of a dynamical system. Instead we require the dynamical system itself to tell it's characteristic features, e.g., critical points and cycles. Obviously the user runs into troubles if this feature is not implemented for a specific dynamical system he would like to visualize.

See Fig. 8.1 for a sample network showing the concept of visualization based on DynSys3D. In the upper part three groups of modules can be dis-

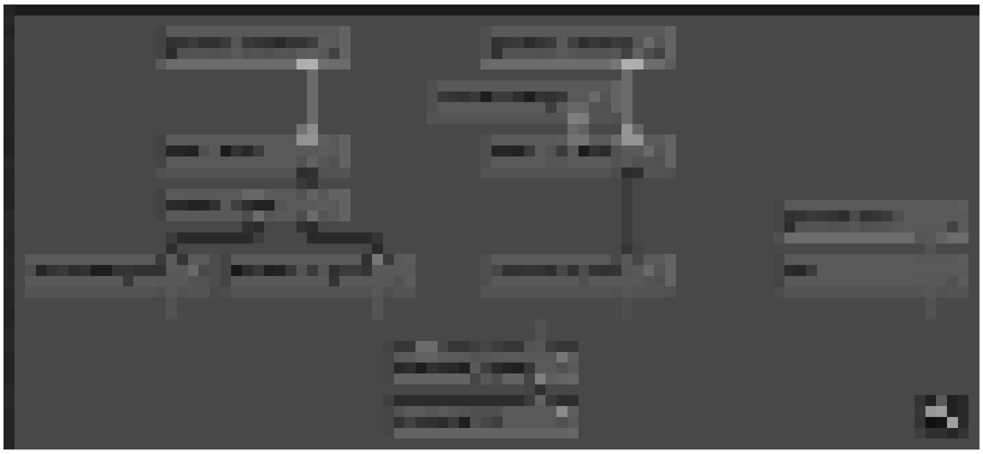


Figure 8.1: An example for a data-flow network in DynSys3D.

tinguished: on the left side a stream surface is computed and stream arrows are computed. Next to it an "back-stream" surface, i.e., a stream surface computed using backward integration, is computed. On the right side a representation of the axes of phase space is generated. All three groups feed their output into the viewer module where the scene is rendered.

8.5 System capabilities

In the scope of the projects presented in previous chapters of this thesis a number of system components were implemented into DynSys3D. Basic visualization techniques as, e.g., Streamline and Streamsurface have been realized within the system [67]. Additionally three different numerical integrators, namely Euler, RK4, i.e., a Runge-Kutta integrator of fourth order, and AdapRK4, a fourth order Runge-Kutta integrator with fifth order error correction, is available to DynSys3D developers [70]. Several dynamical systems have been implemented as well. Prominent examples are, e.g., Lorenz and Roessler, two well-known chaotic attractors [63].

Chapter 9

Summary

> If you have an important point to make, don't try to be subtle or clever. Use a pile driver. Hit the point once. Then come back and hit it again. Then hit it a third time – a tremendous whack.
>
> *Winston Churchill (1874-1965)*

Flow visualization and the visualization of dynamical systems make up an important research area in the field of visualization. Complex dynamics apparent over multi-dimensional domains require sophisticated visualization techniques to be investigated efficiently. Either simulated data – usually huge data-sets resulting from finite element calculations – or analytic models – differential equations representing a dynamical system – significantly extend the size of easily and quickly understandable information. This, generally, is the point where visualization is required.

Quite a reasonable number of useful visualization techniques already are available for the purpose of getting insight in data-sets of that size. Direct visualization by means of direct visual encoding of the given dynamics probably is the most intuitive and, thus, the oldest approach to the visualization task. Hedgehog plots, streamlets and stream lines already have a long tradition in flow visualization. Especially in 2D they directly and intuitively represent the motion to be visualized.

More elaborated techniques like spot noise and line integral convolution, stream lines in 3D and stream surfaces, stream balls, stream ribbons, stream polygons and stream tube, flow volumes, as well as particle systems enrich the possibilities of directly visualizing a dynamical system or flow data.

Instead of directly representing the flow dynamics, sometimes local or abstract data derived from the dynamical system should be visualized. Critical points, for example, usually are especially interesting. Thus, local information extracted from higher-order derivatives, are visualized using, for instance, glyphs or icons. Abstract data, like lower-dimensional skeletons that describe the flow dynamics just qualitatively, i.e., the topological structure of a dynamical system, often are of special interest also. Combining those two approaches appears to be especially useful, since a dense representation is used, the most important information is conveyed.

9.1 Stream arrows

The direct extension to stream lines in 2D are stream surfaces in 3D. A one-dimensional set of initial conditions is developed through phase space by the use of local integration of the flow data – a stream surface is constructed. Since three-dimensional space is much more difficult to investigate than 2D, some disadvantages are given for the use of stream surfaces. The projection of 3D data into image space and the use of large-scale surface structures (stream surfaces), often cause occlusion to become a severe problem – important parts of the visualization cues may be hidden from the viewer.

Stream arrows, i.e., the use of a special texture featuring semi-transparent arrows within the stream surface, enhances the visualization of three-dimensional flow by several means:

- The problem of occlusion is diminished, as parts of the stream surface are modeled semi-transparently and the viewer is able to perceive visualization cues placed behind. See Fig. 4.3 for a typical image featuring stream arrows.
- With stream arrows orientation, direction and velocity of the flow is shown additional to the spatial extent of the integral surface. Even convergence/divergence becomes visible by the use of stream arrows.

Extensions to stream arrows are:

- Anisotropic spot noise is combined with stream arrows to depict stream lines as well as time lines simultaneously on the stream surface (see Fig. 4.11).

- Stream arrows can be extended into 3D. For instance, the vicinity of a stream surface is visualized together with the stream surface. In the case of stable (stream) surface-sets the behavior near such a surface often is very important to be understood. The integrated stream arrow in Fig. 4.14 shows the attracting character of the stream surface.

The procedure for generating stream arrows is rather straight forward. First, a stream surface is calculated for a specific set of initial conditions. Texture coordinates are assigned to each vertex using the following principle: A 1D parameterization of the set of initial conditions, for instance, the arc-length parameterization, is assumed. All surface points lying on one stream line inherit the 1D parameter of the associated initial condition as u-coordinate. The v-coordinate of a surface point is the integration time, assuming the initial condition to have time zero (see Fig. 4.5).

Next, the stream arrows texture is defined. Vertical lines in texture space are correlated to stream lines, horizontal lines correspond to time lines within the stream surface. In the implementation the texture is defined on the basis of a base tile representing one arrow and a tiling mechanism generating implicitly as many stream arrows as necessary. Instead of one single stream arrows texture, an entire stack of textures can be used (hierarchical stream arrows). This eliminates some problems with the standard stream arrows technique in cases of great local divergence/convergence. See Fig. 4.8 for an illustration of this extension.

There are (at least) two possibilities to realize stream arrows within stream surfaces. One is to specify an alpha-texture for the surface element, and let the renderer care about semi-transparency. Another technique is to geometrically segment the stream surface into three separate triangle sets, one for the opaque parts, the border elements (1D), and the semi-transparent parts. An efficient segmentation algorithm is described in Sect. 4.3.

Anisotropic spot noise is generated using the same texture coordinates specification as used for the stream arrows and again exploiting the correlation between u-/v-lines and stream/time lines. A cyclic texture is generated in texture space. Constant flow along u-lines is assumed and an anisotropic spot is used to emphasis stream and time lines, simultaneously. See Fig. 4.10 for an (enlarged) image of the spot and the resulting texture. Mapping such an anisotropic spot noise texture to the stream surface illustrates stream and time lines.

Another approach to diminish the problem of occlusion is to use selective cuts through the visualization model. Parts in front of others, also important parts of the visualization are rendered almost transparently to allow insight within the model. Animation is used to move cut planes and help the viewer to understand the cutting operation performed. See Fig. 4.12 for two images out of an animation where the cut plane was moved through the model. The intersection of cut plane and stream surface was enhanced by white tubes.

In addition to selective cuts there are several other points in the stream arrows technique, where animation easily and useful can be hooked in. Arrows can be move over the stream surface into the direction of flow, the initial conditions may be altered within an animation. Viewpoint animation also improves the perceptibility of visualization models generated using the stream arrows technique.

9.2 Poincaré maps and visualization

Dynamical systems often exhibit cyclic or quasi-cyclic behavior, e.g., food chains, oscillating chemical reactions, weather models based on the period of one year, etc. The cyclic property of such systems usually dominates the character of the behavior. Since the periodic or quasi-periodic behavior usually is known in advance, the local changes turn after turn are much more interesting.

In cases like these Poincaré maps, a technique used by mathematicians, becomes useful. A planar cross-section, called the Poincaré section, is placed orthogonal to the periodic flow. Consecutive intersections of flow trajectories are related via the Poincaré map, i.e., a discrete dynamical system of one dimension less than the cyclic flow. This Poincaré map inherits many important properties from the periodic or quasi-periodic flow. See Fig. 5.1 for an illustration of this relation between the 3D flow and its 2D Poincaré map.

The visualization of periodic or quasi-periodic dynamical systems can be done on the basis of Poincaré maps. Several possibilities are given:

Visualizing $\{\mathbf{p}(\mathbf{x}_i) \mid 0 \leq i \leq m\}$ – A direct visualization of Poincaré map $\mathbf{p}$ is to visually correlate $\mathbf{x}$ and $\mathbf{p}(\mathbf{x})$. This can be done by placing small arrows on the Poincaré section with the tail aligned with $\mathbf{x}$ and the head coinciding with $\mathbf{p}(\mathbf{x})$. See Fig. 5.4 for a visualization of a non-linear saddle cycle where this technique was used.

Visualizing $\mathbf{p}(\mathcal{S})$ – Instead of plotting a rather small number of arrows for a discrete set of pairs $(\mathbf{x}_i, \mathbf{p}(\mathbf{x}_i))$, a continuous representation of $\mathbf{p}(\mathcal{S})$ by the use of adapted spot noise can be used. Elliptic spots are placed on the spot noise texture such that the focal points of the ellipses coincide with $\mathbf{x}$ and $\mathbf{p}(\mathbf{x})$. Using a high number of spots a direct representation of continuous $\mathbf{p}(\mathcal{S})$ is achieved (see Fig. 5.6).

Visualizing $\{\mathbf{p}^j(\mathbf{x}_0) \mid j \geq 0\}$ – Often the long-term behavior of a dynamical system is of special interest. Thus the visualization of the repeated application of Poincaré map $\mathbf{p}$ is also very important. Instead of showing many arrows representing *one* application of $\mathbf{p}$ to many points of the Poincaré section, $\{\mathbf{p}^j(\mathbf{x}_0) \mid j \geq 0\}$, i.e., the repeated application of $\mathbf{p}$ to one specific initial condition is represented. The discrete orbit can be shown, for instance, as set of small spheres. See again Fig. 5.4, where a few orbits are included to visualize the long-term evolution induced by the dynamical system.

Visualizing $\mathbf{p}^q$ instead of $\mathbf{p}^1$ – Sometimes, the investigation of $\mathbf{p}^q$ is more useful than visualizing $\mathbf{p}$ itself. Usually this is the case, if a q-loop cycle, i.e., a cycle that closes after q revolutions, dominates the behavior of the dynamical system. Visualization on the basis of Poincaré maps should take this into account (see Fig. 5.8).

Visualizing $\{\mathbf{p}^j(\mathcal{S}) \mid j \geq 0\}$ – Using animation even the repeated application of Poincaré map $\mathbf{p}$ to the continuous sub-sets of Poincaré section $\mathcal{S}$, i.e., $\{\mathbf{p}^j(\mathcal{S}) \mid j \geq 0\}$, can be visualized. For efficiency reasons, a texture on sub-set $\mathcal{S}$ is transformed using a warp approximation of $\mathbf{p}$ in discrete time steps for each application of $\mathbf{p}$. The vectors of the warp approximation are derived from the Poincaré map $\mathbf{p}$.

Adding Flow Visualization – It is also helpful to combine Poincaré section visualization with flow visualization. The relation between Poincaré map and continuous flow is depicted. This helps to keep the periodic or quasi-periodic structure in mind when investigating the Poincaré map (see the figure on page 54).

9.3 Visualization of critical points

Besides the direct visualization of dynamical systems the visualization after analysis is an important field of research. Often the topology of a dynamical system is extracted in advance, and visualization is used afterwards to communicate the results. Usually the following procedure is used: First, the critical points are identified. Next, the Jacobian matrix is investigated for all critical points to classify their type (attractor, repellor, etc.), and to extract the characteristic trajectories emanating from these points. Further investigation is used to clarify the interrelation between the critical points. Then, critical elements of higher order, e.g., cycles or tori, are searched. Again local derivatives are investigated to identify the type of these higher order critical elements.

The identification of critical points as the most important step of flow topology analysis raises a demand on sophisticated visualization techniques to convey the results of the analysis. Location, type, as well as the local properties derived from the Jacobian matrix can be visualized. Depending on the type of the critical point different methods are useful:

Three characteristic trajectories – if all eigenvalues of the Jacobian matrix are real, different from each other, and different from zero, three characteristic trajectories are connected to the critical point. If all of the eigenvalues are negative (positive), the critical point is an attracting (repelling) node. Mixed signs indicate real saddles.

An intuitive visualization is to (numerically) integrate the characteristic trajectories and indicate the order of attraction/repulsion by the number of arrow glyphs positioned onto them. As an example Fig. 6.1(a) shows the critical point of a linear dynamical system.

Two characteristic sets (1D & 2D) – if two of three real eigenvalues are equal or a pair of complex numbers, one local characteristic structure is a surface. Again, the type of the critical point can be an attractor, repellor, or saddle.

Visualizing such a critical point, a stream surface is integrated to represent the 2D characteristic element, and two stream lines are integrated for the 1D characteristic direction associated with the critical point.

Other types – Of course there are other, usually non-hyperbolic types of critical points. Arbitrary complex local structures can appear, especially if the Jacobian matrix degenerates and higher-order derivatives are to be investigated for analysis.

Specifying a general visualization technique for all these types is cumbersome and usually not necessary, since usually critical points are of one of the simple types described above – non-hyperbolic systems often are considered to be just a transitional element between two hyperbolic systems with differing behavior.

For the visualization of system abstractions it is useful to also include a certain amount of direct visualization to give a few intuitive visual hints for understanding the abstract structure. Streamlets, for example, usually are easily understood and, thus, well suited for being combined with the visualization of flow topology. See Fig. 6.5 for three examples of such a combination. Characteristic elements as well as few direct visual hints, i.e., streamlets originating near the critical points, are displayed.

9.4 Visualizing characteristic structures

Besides critical points, characteristic trajectories belong to the most important elements of flow topology. Similar to the visualization of critical points, it is useful to combine the visualization of abstract topology with direct visualization also in the case of characteristic curves.

In Chapter 7 the 'thread of streamlets' technique is proposed to selectively place streamlets in the vicinity of characteristic trajectories. A certain probability function is used to randomly choose seed points for a numerous set of streamlets, i.e., a thread of streamlets. Using constant flow as a reference model a relation between number of streamlets, integration length and seed point distribution is derived. The actual spatial arrangement, shape, and length of the streamlets communicates the flow velocity near the base trajectory and local information about convergence/divergence, and rotation.

See Figs. 7.3(b) and 7.4(b) for two examples where characteristic trajectories belonging to the critical points of a linear dynamical systems are visualized using this technique. Compared to the visualization of topological information as shown in Fig. 7.3(a) advantages and disadvantages can be found:

+ the dynamics related to the topological structure is more intuitively displayed using threads of streamlets. Having just topological information, it is usually impossible (or quite difficult, at least) to imagine, how a trajectory would evolve starting from a specific state near the critical point.

– the set of visual cues necessary for visualization is less dense using threads of streamlets than displaying topological information only. This reduces the remaining bandwidth of the visual channel used for visualization. The combination of this technique with other visualization results might cause problems due to visual overload.

An important topic when using 1D elements for visualization in 3D. As 1D elements in 3D have an infinite number of normals (opposed to surfaces with one pair of normals), more elaborate techniques must be used. A physically correct solution would require to evaluate an integral over all directions within the normal plane. Simple but useful approximations are available that allow efficient rendering of 1D elements in 3D.

Besides line shading the problem of line shadowing should be addressed. Again, a correct solution would require costly computations similar to volume rendering. Instead, depth cueing can be used as a rough approximation of line shadowing. Combining line shading and line shadowing, or, at least, some approximations of these aspects, allows to render useful results with the 'threads of streamlets' technique.

Threads of streamlets are extended in several directions. Color coding is used for communicating local properties. Arrow heads can be attached to the streamlets, also providing parameters for visualization (see Figs. 7.4(a) and 7.5(a)).

The entire project on advanced visualization techniques concerning dynamical systems, including the sub-projects stream arrows, visualization based on Poincaré maps, visualizing critical points, and the visualization of characteristic structures, shares a common implementation platform, called DynSys3D. The system itself is based on AVS, which is a general purpose visualization system, featuring a data-flow model together with a scheme to build modules. A few design features characterize DynSys3D:

Principal components – Each visualization module developed within the scope of DynSys3D consists of, at least, three principal components: Dynamical System, Numerical Integrator, and Visualization Technique. Rather strict interface specifications between the principal components allow to freely combine different components with each other. For example, it is very easy to reuse a new, integration technique with other existing visualization modules.

Separating Visualization and User Interface – Each instance of the component Visualization Technique is itself composed of a core providing the visualization, and a shell adding the user interface which is visible through the AVS environment. This separation allows to reuse already implemented visualization techniques like stream line integration within other, maybe more visualization methods like processing a rake of streamlets.

Focus on Visualization – Instead of coming up with general solutions, in related fields that do not directly contribute to visualization research, certain problems like the identification of critical points or the evaluation of the Jacobian matrix where decided to be solved individually instead of coming up with a general solver for all cases. A dynamical system, for example, is required to know about its critical points instead of using some general piece of code which is able to find critical points for any dynamical system.

Interactivity – One major aim of DynSys3D was to be a platform for rapid development of new ideas in the field of visualizing dynamical systems. To examine visualization results in a reasonable time it is necessary to prevent the rendering pipeline of being overloaded with too many geometry elements. The representation of the geometry being generated by DynSys3D modules is parameterized so that coarse approximations of the calculated geometry can be used for investigation. Decoupling numerical simulation accuracy and output geometry complexity allows to compute accurate solutions, but use only a small number of triangles for investigation.

Conclusions

> A clever person solves a problem. A wise person avoids it.
> *Albert Einstein (1879-1955)*

> Those are my principles. If you don't like them I have others.
> *Groucho Marx (1890-1977)*

After this project on the visualization of dynamical systems, i.e., after years of research in this field, a few conclusions can be drawn. Maybe the most important insight was the following: it is useful to intuitively communicate dynamics by the use of direct visualization. It is also useful to first derive topology data and then, afterwards, do the visualization. It seems, however, to be most useful to combine both approaches: first, see whether characteristic elements can be identified by the use of dynamical system analysis. Then, visualize these topological structures *and* add direct visualization to include some intuitive hints for reading the abstract representation.

Another experience learned from this work is an important implication of the rather obvious fact 'the bandwidth of the visual channel available for communication via visualization is limited to a certain extent': visualization research is not only working on the question 'How to visualize certain information'; equally important is the question '*What* information should be visualized, or what part of, i.e., what shall be *omitted*, etc.'.

Usually it is not sufficient to provide high-quality software to users. Often the visualization expert necessarily is included within the process of visualization to generate useful results. The knowledge of how to map data to visual information is in many cases not intuitive, and has to be learned also. Either users have to be trained to use visualization, or visualization experts are to be included within the visualization process.

An interesting point for doing visualization of three-dimensional data, is that 3D is more than just an extension of 2D. New challenges wait in 3D, for instance, characteristic structures of dynamical systems that simply do not exist in 2D like saddle cycles or tori. The rendering step involves a projection from 3D into 2D. This confronts the visualization expert with problems that are significantly more complex than 2D to 3D extensions would produce. Occlusion, for example, is a problem of a kind, which simply does not exist in 2D.

A conclusion can be drawn which is not limited to scientific visualization, computer graphics, or even technical science: the optimal visualization technique obviously depends on what the user wants to see! Especially in the case of visualization, the questions of users often vary in a significant way. New ideas in this field often do not replace other already existing methods, but, rather enrich the assortment of possible views onto the user's data.

Bibliography

[1] R. Abraham and C. Shaw. *Dynamics – The Geometry of Behavior*. Addison Wesley, 2nd edition, 1992.

[2] Advanced Visual System, http://www.avs.com/. *AVS Developers Guide, Release 4*, May 1992.

[3] International AVS center, module repositories. Indexed at http://-www.iavsc.org/. These cataloges provide a large number of public domain AVS modules.

[4] V. Arnold. *Ordinary Differential Equations*. MIT Press, 1973.

[5] D. Arrowsmith and C. Place. *An Introduction to Dynamical Systems*. Cambridge University Press, 1990.

[6] D. Asimov. Notes on the topology of vector fields and flows. Technical Report RNR-93-003, NASA Ames Research Center, 1993.

[7] D. Asimov, L. Hesselink, and F. Post. Visualization and topology of vector and tensor fields. IEEE Visualization '95, tutorial notes #6, 1995.

[8] D. Banks. Illumination in diverse codimensions. *Computer Graphics (SIGGRAPH '94 Proceedings)*, 28:327–334, 1994.

[9] D. Banks and B. Singer. Vortex tubes in turbulent flows: Identification, representation, reconstruction. In *Proc. of IEEE Visualization '94*, pages 132–139, October 1994.

[10] R. Beach. *An Introduction to the Curves and Surfaces of Computer-Aided Design*. Van Nostrand Reinhold, 1991.

[11] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26:35–42, July 1992.

[12] M. Brill, W. Djatschin, M. Hagen, S. Klimenko, and H.-C. Rodrian. Streamball techniques for flow visualization. In *Proc. of IEEE Visualization '94*, pages 225–231, October 1994.

[13] I. Bronstein and K. Semendjajew. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 1980.

[14] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 263–270, 1993.

[15] T. Conneen. Spontaneous oscillations in electrical power transmission system models. See URL `http://www.ee.cornell.edu/~conneen/-chaos/chaos.html`.

[16] R. Crawfis, C. Hansen, N. Max, G. Nielson, and W. Schröder. Advanced techniques for scientific visualization. *SIGGRAPH '94, course notes*, 1994.

[17] R. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In *Proc. of IEEE Visualization '93*, pages 261–267, October 1993.

[18] R. Crawfis and N. Max. Flow volumes. See URL `http://www.llnl.gov/-graphics/flow.html`, 1998.

[19] W. de Leeuw and J. van Wijk. A probe for local flow field visualization. In *Proc. of IEEE Visualization '93*, pages 39–45, 1993.

[20] W. de Leeuw and J. van Wijk. Enhanced spot noise for vector field visualization. In *Proc. of IEEE Visualizaiton '95*, pages 233–239, 1995.

[21] T. Delmarcelle and L. Hesselink. Visualization of second order tensor fields and matrix data. In *Proc. of IEEE Visualization '92*, pages 316–322, October 1992.

[22] E. Doedel. *Software for Continuation and Bifurcation Problems in Ordinary Differential Equations (AUTO 86 user manual)*, 2nd edition, February 1986. See also URL `http://www.chpc.utah.edu/software/math/-automan.html`.

[23] L. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In *Proc. of IEEE Visualization '94*, pages 240–247, October 1994.

[24] L. Forssell and S. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, June 1995.

[25] T. Frühauf. Raycasting vector fields. In *Proc. of IEEE Visualization '96*, pages 115–120, 1996.

[26] A. Fuhrmann, H. Löffelmann, and D. Schmalstieg. Collaborative augmented reality: Exploring dynamical systems. In *Proc. of IEEE Visualization '97*, pages 459–462, October 1997.

[27] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Proc. of IEEE Visualization '91*, pages 33–40, October 1991.

[28] E. Gröller, R. Rau, and W. Straßer. Modeling and visualization of knitwear. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):302–310, December 1995.

[29] C. Hansen. Visualization of vector fields (2D and 3D). *SIGGRAPH '93, course notes "Introduction to Scientific Visualization Tools and Techniques"*, August 1993.

[30] A. Hanson and H. Ma. Visualizing flow with quaternion frames. In *Proc. of IEEE Visualization '94*, pages 108–115, 1994.

[31] J. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics & Applications*, 11(3):36–46, 1991.

[32] J. Hubbard and B. West. *Differential Equations: A Dynamical Systems Approach: Ordinary Differential Equations*, volume 5 of *Texts in Applied Mathematics*. Springer, 1991.

[33] J. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proc. of IEEE Visualization '92*, pages 171–177, October 1992.

[34] V. Interrante, H. Fuchs, and S. Pizer. Illustrating transparent surfaces with curvature-directed strokes. In *Proc. of IEEE Visualisation '96*, pages 211–218, 1996.

[35] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proc. of IEEE Visualization '97*, pages 421–424, 1997.

[36] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing*, Focus on Computer Graphics, pages 43–55. Springer, 1997. Proc. of 8$^{\text{th}}$ Eurographics Workshop on Visualization in Scientific Computing, Boulogne sur Mer, France, Apr 28-30, 1997.

[37] B. Jobard and W. Lefer. The motion map: Efficient computation of steady flow animations. In *Proc. of IEEE Visualization '97*, pages 323–328, October 1997.

[38] J. Kajiya and T. Kay. Rendering fur with three dimensional textures. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23:271–280, 1989.

[39] D. Kenwright and R. Haimes. Vortex identification - applications in aerodynamics. In *Proc. of IEEE Visualization '97*, pages 413–416. IEEE Computer Society, `http://www.computer.org/`, November 1997.

[40] D. Kenwright and G. Mallinson. A 3-d streamline tracking algorithm using dual stream functions. In *Proc. of IEEE Visualization '92*, pages 62–68, 1992.

[41] D. Lane. Visualization of time-dependent flow fields. In *Proc. of IEEE Visualization '93*, pages 32–38, October 1993.

[42] D. Lane. UFAT - a particle tracer for time-dependent flow fields. In *Proc. of IEEE Visualization '94*, pages 257–264, October 1994.

[43] C. Levit. Visualizing the topology of vector fields – an annotated bibliography. Technical Report RNR-92-032, NASA Ames Research Center, 1992.

[44] H. Löffelmann, H. Doleisch, and E. Gröller. Visualizing dynamical systems near critical points. In *Proc. of Spring Conference on Computer Graphics and its Applications 1998*, pages 175–184, Budmerice, Slovakia, April 1998.

[45] H. Löffelmann and E. Gröller. DynSys3D: A workbench for developing advanced visualization techniques in the field of three-dimensional dynamical systems. In *Proc. of The Fifth International Conference in Central Europe on Computer Graphics and Visualization '97*, pages 301–310, Plzen, Czech Republic, February 1997.

[46] H. Löffelmann and E. Gröller. Enhancing the visualization of characteristic structures in dynamical systems. In *Proc. of 9th EUROGRAPHICS*

Workshop on Visualization in Scientific Computing, pages 35–46, April 1998. Republished in D. Bartz (ed.): Visualization in Scientific Computing '98, Springer, 1998.

[47] H. Löffelmann, E. Gröller, R. Wegenkittl, and W. Purgathofer. Classifying the visualization of analytically specified dynamical systems. *Machine* GRAPHICS *&* VISION, 5(4):533–549, 1996.

[48] H. Löffelmann, A. König, and E. Gröller. Fast visualization of 2D dynamical systems by the use of virtual ink droplets. In *Proc. of Spring Conference on Computer Graphics and its Applications 1997*, pages 111–118, Budmerice, Slovakia, June 1997.

[49] H. Löffelmann, T. Kučera, and E. Gröller. Visualizing Poincaré maps together with the underlying flow. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization, Algorithms, Applicatoins, and Numerics*, pages 315–328. Springer, 1998.

[50] H. Löffelmann, L. Mroz, and E. Gröller. Hierarchical streamarrows for the visualization of dynamical systems. In *Proc. of 8th EUROGRAPHICS Workshop on Visualization in Scientific Computing*, pages 203–211, April 1997. Republished in W. Lefer and M. Grave (eds.): Visualization in Scientific Computing '97, Springer, pp. 155–163.

[51] H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems. *The Visual Computer*, 13(8):359–369, 1997.

[52] H. Löffelmann, Z. Szalavári, and E. Gröller. Local analysis of dynamical systems – concepts and interpretation. In *Proc. of The Fourth International Conference in Central Europe on Computer Graphics and Visualization '96*, pages 170–180, Plzen, Czech Republic, February 1996.

[53] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995.

[54] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proc. of IEEE Visualization '93*, pages 19–24, 1993.

[55] W. Merzkirch. *Flow Visualization.* Academic Press, 2nd edition, 1987.

[56] A. Milik. *Dynamics of Mixed-mode Oscillations.* PhD thesis, Vienna University of Technology, 1996.

[57] A. Milik, P. Szmolyan, H. Löffelmann, and E. Gröller. Geometry of mixed-mode oscillations in the 3-d autocatalator. *International Journal of Bifurcation and Chaos*, 8(3):505–520, 1998.

[58] Data analysis group (NAS, NASA) images and animations. See URL `http://science.nas.nasa.gov/Groups/VisTech/images.html`, 1998.

[59] G. Nielson, H. Hagen, and H. Müller. *Scientific Visualization: Overviews, Methodologies, and Techniques*. IEEE Computer Society Press, 1997.

[60] G. Nielson, B. Shriver, and L. Rosenblum. *Visualization in Scientific Computing*. IEEE Computer Society Press tutorial. IEEE Computer Society Press, 1990.

[61] H.-G. Pagendarm and B. Walter. Feature detection from vector quantities in a numerically simulated hypersonic flow field in combination with experimental flow visualization. In *Proc. of IEEE Visualization '94*, pages 117–123, October 1994.

[62] H.-G. Pagendarm and B. Walter. Feature detection in vector fields. See URL `http://www.sm.go.dlr.de/sm-sk_info/library/documents/-hgpVis94/`, 1995.

[63] H.-O. Peitgen, H. Jürgens, and D. Saupe. *Chaos and Fractals – New Frontiers of Science*. Springer, 1992.

[64] B.-T. Phong. Illumination for computer generated pictures. *CACM June 1975*, 18(6):311–317, 1975.

[65] C. Pickover and S. Tewksbury, editors. *Frontiers of Scientific Visualization*. Wiley Interscience, 1993.

[66] H. Poincaré. *Les Méthodes Nouvelles de la Mécanique Céleste (3 vols.)*. Gauthier-Villars, 1899.

[67] F. Post and T. van Walsum. Fluid flow visualization. In H. Hagen, H. Müller, and G. Nielson, editors, *Focus on Scientific Visualization*, pages 1–40. Springer, 1993.

[68] F. Post, T. van Walsum, and F. Post. Iconic techniques for feature visualization. In *Proc. of IEEE Visualization '95*, pages 288–295, 1995.

[69] F. Post and J. van Wijk. Visual representation of vector fields: Recent developments and research directions. In L. Rosenblum et al., editors, *Scientific Visualization – Advances and Challenges*, pages 367–390. Academic Press, 1994.

[70] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[71] P. Rheingans. Opacity-modulating triangular textures for irregular surfaces. In *Proc. of IEEE Visualisation '96*, pages 219–225, 1996.

[72] S. Rinaldi. Nonlinear dynamical systems: Bifurcation and chaos with applications in ecology. Notes to a lecture held at the Institute of Econometrics, Vienna University of Technology, May 1995.

[73] L. Rosenblum et al., editors. *Scientific Visualization - Advances and Challenges*. Academic Press, 1994.

[74] M. Rumpf and R.-T. Happe. Characterizing gobal features of simulation data by selected local icons. In *Virtual Environments and Scientific Visualization '96*. Springer, 1996.

[75] W. Schröder, C. Volpe, and W. Lorensen. The stream polygon: A technique for 3D vector field visualization. In *Proc. of IEEE Visualization '91*, pages 126–132, October 1991.

[76] J. Sethna. Poincaré section for the L5 point. See URL `http://-www.physics.cornell.edu/sethna/teaching/sss/jupiter/Web/-L5Poinc.htm`. Author's email is `sethna@lassp.cornell.edu`.

[77] R. Seydel, F. Schneider, T. Küpper, and H. Troger, editors. *Bifurcation and Chaos: Analysis, Algorithms, Applications*, volume 97 of *International Series of Numerical Mathematics*. Birkhäuser, 1991.

[78] H.-W. Shen and D. Kao. UFLIC: A line integral convolution algorithm for visualizing unsteady flows. In *Proc. of IEEE Visualization '97*, pages 317–323. IEEE Computer Society, `http://www.computer.org/`, 1997.

[79] D. Stalling and H.-C. Hege. Fast and resolution independent line integral convolution. *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 249–256, August 1995.

[80] S. Strogatz. *Nonlinear Dynamics and Chaos: with applications to physics, biology, chemistry, and engineering*. Addison Wesley, 1994.

[81] R. Tobler, H. Löffelmann, T. Galla, and W. Purgathofer. BaBeL: A generic data structure for geometric modeling. In *Proc. of The Fourth International Conference in Central Europe on Computer Graphics and Visualization '96*, pages 359–366, Plzen, Czech Republic, February 1996.

[82] R. Tobler, H. Löffelmann, and W. Purgathofer. VEGA: Vienna environment for graphics applications. In *Proc. of The Third International Conference in Central Europe on Computer Graphics and Visualization '95*, pages 323–328, Plzen, Czech Republic, February 1995.

[83] A. Tsonis. *Chaos – From Theory to Applications*. Plenum Press, 1992.

[84] G. Turk and D. Banks. Image-guided streamline placement. *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 453–460, August 1996.

[85] M. van Dyke. *An Album of Fluid Motion*. The Parabolic Press, 1982.

[86] T. van Walsum, F. Post, D. Silver, and F. Post. Feature Extraction and Iconic Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):111–119, June 1996.

[87] J. van Wijk. Spot noise – texture synthesis for data visualization. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25:309–318, July 1991.

[88] J. van Wijk. Flow visualization with surface particles. *IEEE Computer Graphics & Applications*, 13(4):18–24, July 1993.

[89] R. Wegenkittl and E. Gröller. Fast oriented line integral convolution for vector field visualization via the Internet. In *Proc. of IEEE Visualization '97*, pages 309–316, October 1997.

[90] R. Wegenkittl, E. Gröller, and W. Purgathofer. Animating flowfields: Rendering of oriented line integral convolution. In *Proc. of Computer Animation '97*, 1997.

[91] R. Wegenkittl, H. Löffelmann, and E. Gröller. Visualizing the behavior of higher dimensional dynamical systems. In *Proc. of IEEE Visualization '97*, pages 119–125, October 1997.

[92] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, volume 2 of *Texts in Applied Mathematics*. Springer, 3rd edition, 1996.

[93] W. Yang, editor. *Handbook of Flow Visualization*. Hemisphere Publishing, 1989.

[94] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualization of 3D vector fields using illuminated stream lines. In *Proc. of IEEE Visualization '96*, pages 107–113, 1996.

Appendix A

Related URLs

> Outside show is a poor substitute for inner worth.
> *Aesop (620-560 BC)*

Stream arrows –

- http://www.cg.tuwien.ac.at/research/vis/dynsys/-Streamarrows96/
- http://www.cg.tuwien.ac.at/research/vis/dynsys/-HierStreamarrows97/
- http://www.cg.tuwien.ac.at/˜mroz/da/

Poincaré Maps and Visualization –

- http://www.cg.tuwien.ac.at/research/vis/dynsys/-Poincare97/
- http://www.cg.tuwien.ac.at/˜tkucera/

Visualization of Critical Points –

- http://www.cg.tuwien.ac.at/research/vis/dynsys/-AdvFPViz/

Visualizing Characteristic Trajectories –

- http://www.cg.tuwien.ac.at/research/vis/dynsys/-KnitDS97/

Implementation: DynSys3D –

- http://www.cg.tuwien.ac.at/research/vis/dynsys/-DynSys3D96/

Other DynSys3D-Related Material –

- http://www.cg.tuwien.ac.at/studentwork/VisSem97/
- http://www.cg.tuwien.ac.at/research/vr/studierstube/-AVS/html/

References –

- http://www.cg.tuwien.ac.at/~helwig/diss/
- http://www.cg.tuwien.ac.at/~helwig/publs.html
- http://www.cg.tuwien.ac.at/research/vis/dynsys/
- http://www.cg.tuwien.ac.at/research/TR/

Appendix B

Sample dynamical systems

> Example is not the main thing in influencing others. It is the only thing.
>
> *Albert Schweitzer (1875-1965)*

In this appendix a set of simple dynamical systems is discussed, which were specified as test cases for the visualization techniques presented in this thesis. Each exhibits a special feature of 3D dynamical systems as, e.g., a critical point or cycle.

B.1 RealFp

As a very simple test case we define a dynamical system, which has exactly one critical point at the origin of phase space and real eigenvalues/-vectors of the Jacobian matrix there:

$$
\begin{aligned}
\dot{x} &= A \cdot x, \quad A \neq 0 \\
\dot{y} &= B \cdot y, \quad B \neq 0 \\
\dot{z} &= C \cdot z, \quad C \neq 0,\ A \geq B \geq C
\end{aligned}
$$

Depending on the values of A, B, and C the critical point is either attracting, repelling, or a saddle critical point. Note, that relation $A \geq B \geq C$ is not a restriction to this system, since axes x, y, and z are arbitrary choices and can be reordered to fulfill any other relation between A, B, and C.

$$
\begin{aligned}
0 > A \geq B \geq C &\;\rightarrow\; \text{attracting node} \\
A > 0 > B \geq C &\;\rightarrow\; \text{saddle node (attracting } x \text{ axis)}
\end{aligned}
$$

$$
\begin{aligned}
A \geq B > 0 > C &\rightarrow \text{saddle node (repelling } z \text{ axis)} \\
A \geq B \geq C > 0 &\rightarrow \text{repelling node}
\end{aligned}
$$

B.2 ComplFp

In addition to the case that all eigenvalues/-vectors of the critical point's Jacobian matrix are real, a pair of conjugated complex eigenvalues/-vectors can occur. This case is not possible in dynamical system RealFp, thus another system is necessary to test this case – we start with r and ϕ denoted in polar coordinates:

$$
\begin{aligned}
\dot{r} &= A \cdot r, \quad A \neq 0 \\
\dot{\phi} &= 1 \\
\dot{z} &= B \cdot z
\end{aligned}
$$

Depending on the values of A and B this system exhibits either an attracting, repelling, or saddle critical point with a pair of conjugated complex eigenvalues/-vectors:

$$
\begin{aligned}
A < 0,\ B < 0 &\rightarrow \text{attracting focus} \\
A < 0,\ B > 0 &\rightarrow \text{saddle focus (attracting } z \text{ axis)} \\
A > 0,\ B < 0 &\rightarrow \text{saddle focus (repelling } z \text{ axis)} \\
A > 0,\ B > 0 &\rightarrow \text{repelling focus}
\end{aligned}
$$

In terms of Cartesian coordinates ($x = r \cdot \cos\phi,\ y = r \cdot \sin\phi$) this dynamical system can be written as

$$
\begin{aligned}
\dot{x} &= A \cdot x - y \\
\dot{y} &= A \cdot y + x \\
\dot{z} &= B \cdot z
\end{aligned}
$$

B.3 RealCyc

For testing Poincaré maps we define a simple cycle in 3D. The critical point of the Poincaré map of this dynamical system should be either an attracting node, a saddle, or a repelling node. For this purpose we think about points in phase space as specified by coordinates r, ϕ, and z: r and ϕ denote polar coordinates in the x-y plane. RealCyc can now be specified as follows:

$$\begin{aligned}\dot{r} &= A \cdot (r^2 - 1), \quad A \neq 0 \\ \dot{\phi} &= 1 \\ \dot{z} &= B \cdot z, \quad B \neq 0\end{aligned}$$

This dynamical system contains a cycle. This is easy to show: $\dot{r}|_{r=1} = 0$ and $\dot{z}|_{z=0} = 0$ induce that the unit cycle $\mathcal{C} = \{\mathbf{x} : r = 1\}$ is a cycle of this dynamical system. Furthermore it can be stated that there are no other cycles than $\mathcal{C}$ in this specific dynamical system: since $\dot{z}|_{z \neq 0} \neq 0$ no cycle can exist outside the x-y plane; $\dot{r}|_{|r| \neq 1} \neq 0$ assures that no other cycle than the unit cycle resides within the x-y plane.

Using this definition, we can directly influence the Poincaré map of cycle $\mathcal{C}$ by adjusting parameters A and B:

$$\begin{aligned}A < 0,\ B < 0 &\rightarrow \text{stable limit cycle } \mathcal{C} \\ A < 0,\ B > 0 &\rightarrow \text{saddle cycle } \mathcal{C},\ \text{attracting cylinder } r = 1 \\ A > 0,\ B < 0 &\rightarrow \text{saddle cycle } \mathcal{C},\ \text{attracting } x\text{-}y \text{ plane} \\ A > 0,\ B > 0 &\rightarrow \text{instable cycle } \mathcal{C}\end{aligned}$$

To embed this system in the scope of DynSys3D, it has to be defined on the basis of Cartesian coordinates:

$$\begin{aligned}x &= r \cdot \cos\phi \\ y &= r \cdot \sin\phi\end{aligned}$$

The dynamical system in terms of Cartesian coordinates is then given as:

$$\begin{aligned}\dot{x} &= a \cdot x - y, \quad a = A \cdot (r - 1/r),\ r = \sqrt{x^2 + y^2} \\ \dot{y} &= a \cdot y + x \\ \dot{z} &= b \cdot z, \quad b = B\end{aligned}$$

B.4 NLCyc1

Another test case for the Poincaré section techniques is NLCyc1, which is a dynamical system that exhibits one cycle with a non-linear Poincaré map.

First we specify a linear 2D system with a critical point in the origin and characteristic directions coinciding with the axes of the system:

$$\begin{aligned}\dot{\alpha} &= A \cdot \alpha \\ \dot{\beta} &= B \cdot \beta\end{aligned}$$

Next we apply a non-linear transformation to variable α: $u = \alpha + \beta^2$, and $v = \beta$. This transformation yields another 2D dynamical system as follows:

$$\begin{aligned} \dot{u} &= A \cdot u + (2B - A) \cdot v^2 \\ \dot{v} &= B \cdot v \end{aligned}$$

We now place the entire system into the half-plane $\{(u', v') : u' > 0\}$ by transformation $u' = e^u,\ v' = v\ (u = \ln u',\ v = v')$ and end up with the following system:

$$\begin{aligned} \dot{u'}/u' &= A \cdot \ln u' + (2B - A) \cdot v'^2 \\ \dot{v'} &= B \cdot v' \end{aligned}$$

Finally we transform the system into 3D by rotating it around the z-axis ($r = u'$, $\dot{\phi} = C$, and $z = v'$). This yields the final system as follows:

$$\begin{aligned} \dot{x} &= x \cdot (A \cdot \ln r + (2B - A) \cdot z^2) - C \cdot y, \quad r = \sqrt{x^2 + y^2} \\ \dot{y} &= y \cdot (A \cdot \ln r + (2B - A) \cdot z^2) + C \cdot x \\ \dot{z} &= B \cdot z \end{aligned}$$

B.5 RealTorus

RealTorus should be a dynamical system, which exhibits an invariant torus, either attracting or repelling. To design this system, we start in 2D:

$$\begin{aligned} \dot{r} &= A \cdot (r^2 - R), \quad A \neq 0,\ R > 0 \\ \dot{\rho} &= 1 \end{aligned}$$

Depending on the value of A this system has either an attracting or repelling cycle of radius R. In Cartesian coordinates ($u = r \cdot \cos\rho,\ v = r \cdot \sin\rho$) this system is given as follows:

$$\begin{aligned} \dot{u} &= a \cdot u - v, \quad a = A \cdot (r - R/r),\ r = \sqrt{u^2 + v^2} \\ \dot{v} &= a \cdot v + u \end{aligned}$$

We now squeeze this system into half-plane $\{(u', v') : u' > 0\}$ by transformation $u' = e^u,\ v' = v\ (u = \ln u',\ v = v')$ and end up with the following system:

$$\begin{aligned} \dot{u'}/u' &= a \cdot u - v, \quad a = A \cdot (r - R/r),\ r = \sqrt{u^2 + v^2} \\ \dot{v'} &= a \cdot v + u \end{aligned}$$

Using this dynamical system in 2D, we can construct a three-dimensional system, which actually contains an invariant torus by the following definition:

$$\begin{aligned} x &= u' \cdot \cos\phi \\ y &= u' \cdot \sin\phi \\ z &= v' \end{aligned}$$

Assuming $\dot{\phi} = C$ $(u = \ln u' = \ln\sqrt{x^2+y^2},\ v = v' = z)$ this dynamical system can be expressed as follows:

$$\begin{aligned} \dot{x} &= (a \cdot u - v) \cdot x - C \cdot y, \quad a = A \cdot (r - R/r),\ r = \sqrt{u^2+v^2} \\ \dot{y} &= (a \cdot u - v) \cdot y + C \cdot x \\ \dot{z} &= (a \cdot v + u) \end{aligned}$$

Appendix C

Notes on the notation

> Welcome to The Machine!
> *Pink Floyd, LP "Wish You Were Here" (1975)*

scalar numbers – italic letters, e.g., n (dimension of phase space), m (dimension of parameter space), indices i and j, time t, period T, radius R.

vector data – bold letters, sometimes indexed, e.g., $\mathbf{x}$ (current state of the dynamical system), $\mathbf{v}$ (some velocity vector), normal $\mathbf{n}$, initial condition $\mathbf{s}$, critical points $\mathbf{c}_i$.

vector components – vector (bold letter) indexed by the use of squared brackets, e.g., $\mathbf{v}[1]$, $\mathbf{n}[i]$.

sets, spaces, manifolds – standard labels for predefined sets, e.g., $\mathbf{R}$ (real numbers) and $\mathbf{C}$ (complex numbers); capital Greek letters for special subsets, etc., e.g., phase space $\Omega \subseteq \mathbf{R}^3$ or parameter space $\Pi \subseteq \mathbf{R}^m$; calligraphic letters for curves, surfaces, manifolds, e.g., trajectory $\mathcal{T}_\mathbf{s}$, cycles $\mathcal{C}_i$.

functions, mappings, parameterizations – embraced functional arguments, e.g., trajectory $\mathcal{T}_\mathbf{s}(t)$, dynamical system specification $\mathbf{f}_\mathbf{p}(\mathbf{x}, t)$, Poincaré map $\mathbf{p}(\mathbf{x})$.

dependence on parameters – parameter(s) as indices, for instance, $\mathbf{f}_\mathbf{p}$.

matrices – bold upper case letters, e.g., Jacobian $\mathbf{J}$, $\mathbf{A}$.

Glossary

class of dynamical systems — Instead of a single → **dynamical system (DS)**, an entire *class of dynamical systems* is given, if the definition of the DS depends on a set of → **system parameter**s.

continuous DS — A → **dynamical system (DS)** is called a *continuous DS*, if its → **system variable**s vary over time continuously. Often a continuous DS is defined as a set of → **ordinary differential equations (ODEs)**. Compare to: → **discrete DS**. See also: → **trajectory**.

discrete DS — A → **dynamical system (DS)** is called a *discrete DS*, if its → **system variable**s vary over time discretely. Usually a discrete DS is defined as a set of difference equations ($\Delta\mathbf{x}(t) = \mathbf{f_p}(\mathbf{x}(t), t),\ t \in \mathbf{Z}$). Compare to: → **continuous DS**. See also: → **orbit**.

dynamical system (DS) — A *dynamical system (DS)* is a set of → **system variable**s, usually called $\mathbf{x}[i]$, that vary over → **time**, which is usually denoted with t. A set of → **system parameter**s may span an entire → **class of dynamical systems**. See also: → **continuous DS**, → **discrete DS**, → **steady DS**, → **time-dependent DS**.

initial condition — A → **solution** to a → **dynamical system (DS)** is dependent on a specific *initial condition*, often called $\mathbf{s}$, and → **time** t. Usually the solution is called → **trajectory** in the case of a → **continuous DS**, and → **orbit** in the case of a → **discrete DS**.

orbit	A → **dynamical system (DS)** and a specific → **initial condition s** together yield an *orbit* through → **phase space** as a local → **solution** of the equations that define the DS. Often the term 'orbit' is used in the case of a → **discrete DS** only. In this case the orbit $\Sigma_{\mathbf{s}}$ of $\mathbf{s}$ is defined as the set of points $\Sigma_{\mathbf{s}}(t) = \mathbf{s} + \sum_{u=0}^{t-1} \mathbf{f_p}(\mathbf{x}(u), u)$, $\mathbf{N} \ni t \geq 0$. Compare to: → **trajectory**.
ordinary differential equations (ODEs)	A set of *ordinary differential equations (ODEs)* ($\dot{\mathbf{x}} = \mathbf{f_p}(\mathbf{x}, t)$, $t \in \mathbf{R}$) can be used to describe continuous dynamics over → **time** t. A → **solution** to equations of this kind is called → **trajectory** or → **orbit**. See also: → **continuous DS**.
phase space	A → **dynamical system (DS)** is a set of → **system variable**s, that vary over → **time**. The (often Cartesian) coordinate system, which is defined by assigning each system variable one coordinate axis is called *phase space*, often denoted by Ω. See also: → **orbit**, → **trajectory**.
solution	See: → **orbit**, → **trajectory**. See also: → **initial condition**.
state	The *state* of a → **dynamical system (DS)** is the set of values of the → **system variable**s at a certain → **time** t. Usually the state of a DS is denoted with $\mathbf{x}(t)$.
state variable	Equivalent to → **system variable**. See also: → **state**.
steady DS	Principally, the definition of a → **dynamical system (DS)** includes a dependence of $\mathbf{f}$ on → **time** t ($\dot{\mathbf{x}}(t) = \mathbf{f_p}(\mathbf{x}(t), t)$, $t \in \mathbf{R}$, or $\Delta\mathbf{x}(t) = \mathbf{f_p}(\mathbf{x}(t), t)$, $t \in \mathbf{Z}$). If $\mathbf{f_p}$ is assumed to be independent from t, i.e., the system definition can be written as $\dot{\mathbf{x}} = \mathbf{f_p}(\mathbf{x})$ or $\Delta\mathbf{x} = \mathbf{f_p}(\mathbf{x})$, the DS is called a *steady DS*. Compare to: → **time-dependent DS**. See also: → **time-independent DS**.

system parameter Instead of a single → **dynamical system (DS)**, an entire → **class of dynamical systems** is given, if the definition of the DS depends on a set of *system parameter*s. Compare to: → **system variable**, → **time**.

system variable A → **dynamical system (DS)** is a set of *system variable*s, usually called $\mathbf{x}[i]$, that vary over → **time** t. Compare to: → **system parameter**, → **time**. See also: → **phase space**, → **state variable**.

time A → **dynamical system (DS)** is a set of → **system variable**s, that vary over *time* t. Compare to: → **system parameter**.

time-dependent DS A → **dynamical system (DS)** is called a *time-dependent DS*, if its definition ($\dot{\mathbf{x}}(t) = \mathbf{f}_\mathbf{p}(\mathbf{x}(t), t)$, $t \in \mathbf{R}$, or $\Delta\mathbf{x}(t) = \mathbf{f}_\mathbf{p}(\mathbf{x}(t), t)$, $t \in \mathbf{Z}$) actually varies over → **time** t. Compare to: → **steady DS**.

time-independent DS Equivalent to: → **steady DS**.

trajectory A → **dynamical system (DS)** and a specific → **initial condition s** together yield an *trajectory* through → **phase space** as a local → **solution** of the equations that define the DS. Often the term 'trajectory' is used in the case of a → **continuous DS** only. In this case the trajectory $\mathcal{T}_\mathbf{s}$ of $\mathbf{s}$ is defined as the curve $\mathcal{T}_\mathbf{s}(t) = \mathbf{s} + \int_{u=0}^{t} \mathbf{f}_\mathbf{p}(\mathcal{T}(u), u)\,\mathrm{d}u$, $\mathbf{R} \supset t \geq 0$. Compare to: → **orbit**.

Curriculum vitae

Helwig K. Löffelmann,
born on April 16^{th}, 1971, in Vienna, Austria.

Education

- September 1977 - June 1981: Volksschule (primary school) in Mistelbach, Austria.
- September 1981 - June 1985: Bundesrealgymnasium (secondary school) in Floridsdorf (Wien), Austria.
- September 1985 - June 1989: Bundesoberstufenrealgymnasium (high school) in Mistelbach.
- October 1989 - April 1995: Studies of Informatik (computer science) at Vienna University of Technology. Diploma thesis "Extended Cameras for Ray Tracing" at the Institute of Computer Graphics, finished in February 1995. Graduation (with distinction) to "Diplom-Ingenieur der Informatik" (computer science) in April 1995.
- Since May 1995: Further studies of Informatik (computer science) at the Vienna University of Technology, i.e., work on this PhD thesis, at the Institute of Computer Graphics. Member of the scientific staff of the Institute of Computer Graphics, since March 1995.

Jobs

- March 1991 - June 1991: Tutor (teaching assistant) at the Institute of Computer Graphics, Vienna University of Technology.
- July 1991 - August 1991: Ferialpraxis (short job in summer) at IBM Austria, Vienna Software Development Laboratory, Austria.
- October 1992 - January 1993: Tutor (teaching assistant) at the Institute for Computer Engineering, Vienna University of Technology.
- June 1992 - Februar 1994: Free-lance programmer (databases and applications with Open Access III).
- March 1994 - February 1995: Studienassistent (assistant) at the Institute of Computer Graphics, Vienna University of Technology.
- Since October 1995: Universitätsassistent (assistant professor) at the Institute of Computer Graphics, Vienna University of Technology.

Publications and Talks

- See Sect. "Bibliography" in this thesis for a sub-set of my publications, or
- `http://www.cg.tuwien.ac.at/~helwig/publs.html` for an up-to-date list.

Contact Information

- E-Mail: `mailto:helwig@cg.tuwien.ac.at`,
- WWW: `http://www.cg.tuwien.ac.at/~helwig/`.
- Street address: Institute of Computer Graphics, Vienna University of Technology, Karlsplatz 13/186/2, A-1040 Wien, Austria.

Acknowledgements

The last lines of this text should be words of gratitude to all those who helped to make this thesis possible. First of all, I have to thank Meister **Edi Gröller** for being a patient superisor and for supporting this work with ideas, criticism, etc. I want to thank **Rainer Wegenkittl** for being a congenial colleague who worked also on the visualization of dynamical systems for quite a long time. I am very thankful to **Lukas Mroz**, **Thomas Kučera**, **Helmut Doleisch**, **Edgar Weippl**, and **Markus Götzinger**, who did great work in implementing the ideas presented in this thesis. **Zsolt Szalavári** helped me to learn about local analysis of dynamical systems, and **Robert F. Tobler** provided me with good hints for the design and set-up of the DynSys3D system. I also want to thank **Alexandra Milik** and **Peter Szmolyan** for the joined work on mixed-mode oscillations. For providing me with an environment and support for my work I want to thank **Werner Purgathofer**. Together with **Andreas König** and **Peter Stieglecker** I had numerous discussions, more or less related to this thesis. Finally, I really have to thank **Albrecht Kadlec**, **Andrea Wähner**, **Anja Deml**, **Elsbeth Gruss**, **Gabi Königshofer**, **Herbert Oppolzer**, **Karin Hummel**, **Maria Hauser**, **Martin Pottendorfer**, and **Stephan Grünfelder** for standing by me in good and bad times,

thank you all!